Folder

58 printable files

PrecompileCosts\CostsCsv.pas
PrecompileCosts\DataNeedsCsv.pas
PrecompileCosts\LCRCompiledCost.pas
PrecompileCosts\LCRCompiledCostTemplate.pas
PrecompileCosts\LCRPreCompile.pas
PrecompileCosts\PrecompileCosts.dpr
PrecompileCosts\frmMain.dfm
PrecompileCosts\frmMain.pas
SafeWaterRun.dpr
code\BenefitModules\PFASSerumCalc.pas
code\BenefitModules\SafeWaterBenefitBase.pas
code\BenefitModules\SafeWaterBenefitModules.pas
code\BenefitModules\SafeWaterCohortArray.pas
code\BenefitModules\bmPFASAbstractCancSC.pas
code\BenefitModules\bmPFASBase.pas
code\BenefitModules\bmPFASCVD.pas
code\BenefitModules\bmPFASDBP.pas
code\BenefitModules\bmPFASKidneyC.pas
code\BenefitModules\bmPFASLBW.pas
code\BenefitModules\bmPFASLiverC.pas
code\BenefitModules\bmPFASStats.pas
code\CalcDistSpecFun.pas
code\CostModules\CompiledCost.pas
code\CostModules\CompiledCostTemplate.pas
code\CostModules\SafeWaterCostBase.pas
code\CostModules\SafeWaterCostModules.pas
code\CostModules\SafeWaterTFObject.pas
code\CostModules\SafeWaterWBSCostCurves.pas
code\CostModules\cmPFASBase.pas
code\CostModules\cmPFASOccRun.pas
code\CostModules\cmPFASOption1.pas
code\CostModules\cmPFASOption2s.pas
code\CostModules\cmPFASOption2x.pas
code\CostModules\cmPFASOption3.pas
code\CostModules\cmPFASOption4.pas
code\CostModules\cmPFASOptionMCL1.pas
code\PBPK\PFASPBPK.pas
code\PBPK\Spliner.pas
code\PBPK\pfaspk.dpr
code\PBPK\wtest.dfm
code\PBPK\wtest.pas
code\SafeWaterCalcDist.pas
code\SafeWaterConfig.pas
code\SafeWaterGlobals.pas
code\SafeWaterMetrics.pas
code\SafeWaterModel.pas
code\SafeWaterModelData.pas
code\SafeWaterOccurrence.pas
code\SafeWaterOccurrence2.pas
code\SafeWaterPWSRecords.pas
code\SafeWaterRunner.pas
code\SafeWaterUncertBucket.pas
code\SafeWaterUtility.pas
swcl.dpr
uicode\wndTests.dfm
uicode\wndTests.pas
wndMain.dfm
wndMain.pas

PrecompileCosts\CostsCsv.pas

```
unit CostsCsv;

interface

uses Classes, SysUtils, Generics.Collections;

//const CSVHeader = '"CostNumber","CostName","CostDescription","CostApplies","TotalCost","Hours","Labor","O&M","Domain","OutputCostGroup","Frequency",
"Conditions","AggregateCostDescription","Tab","FlowChartPosition","ICR","IntermediateAgglomerator","Agglomerator","Year","Blank","ICRAgglomerator","
IncludeCost","VLSEp","Bin1","Bin2","Bin3","Bin4"';

```

```
9  type
10   TExpressionsRecord=record
11     IDNum : integer;
12     RowNum: integer;
13     CostName : string;
14     ApplicationLevel : string;
15     AgglomeratorCategories : string;
16     BriefCostDescription : string;
17     CostDescription : string;
18     CostTrigger : string;
19     StartYear_SmallSys : string;
20     Repeat_SmallSys : string;
21     StartYear_NonSmallSys : string;
22     Repeat_NonSmallSys : string;
23     Formula : string;
24     Notes : string;
25      //procedure LoadFromStream(AStream : TStream);
26      //procedure SaveToStream(AStream : TStream);
27   end;
28
29   TExpressionsRecords = class(TObject)
30   private
31     fExpressionsRecords : TList<TExpressionsRecord>;
32     fExpressionsCategoryCount : TDictionary<integer,integer>;
33     procedure ParseYears(id, sz: integer; sy, ry: string);
34     function getExpressionsRecord(I: Integer): TExpressionsRecord;
35     function getExpressionsRecordCount: Integer;
36   public
37     //mimic structure in CompiledCosts so we can dunp it into source
38     //public because the precompile unit will dump it.
39     fCCCalcYearStore : array[0..1,0..200,0..1000] of boolean;
40
41     constructor Create();
42     destructor Destroy();override;
43
44     procedure LoadFromFile(FName: string; ClearRecs:boolean = true);
45     procedure WriteToFile(FName: string);
46
47     property ExpressionsRecord[I: Integer]: TExpressionsRecord read getExpressionsRecord;
48     property Count: Integer read getExpressionsRecordCount;
49   end;
50
51 implementation
52
53 { TExpressionsRecords }
54
55 constructor TExpressionsRecords.Create;
56 begin
57   fExpressionsRecords := TList<TExpressionsRecord>.create;
58   fExpressionsCategoryCount := TDictionary<integer,integer>.Create;
59 end;
60
61 destructor TExpressionsRecords.Destroy;
62 begin
63   fExpressionsRecords.Free;
64   fExpressionsCategoryCount.Free;
65   inherited;
66 end;
67
68 function TExpressionsRecords.getExpressionsRecord(I: Integer): TExpressionsRecord;
69 begin
70   Result := fExpressionsRecords.items[i];
71 end;
72
73 function TExpressionsRecords.getExpressionsRecordCount: Integer;
74 begin
75   Result := fExpressionsRecords.count;
76 end;
77
78 procedure TExpressionsRecords.LoadFromFile(FName: string; ClearRecs:boolean = true);
79 begin
80   if ClearRecs then begin
81     fExpressionsRecords.Clear;
82     fExpressionsCategoryCount.Clear;
83   end;
```

```
  var T := TStreamReader.Create(FName,TEncoding.ANSI,false,4096);
  var ILine := TStringList.Create;
  ILine.Duplicates := dupIgnore;
  ILine.StrictDelimiter := true;
  ILine.CommaText := T.ReadLine;

  const iIDNum = ILine.IndexOf('ID Number');
  const iCostName = ILine.IndexOf('Short Cost Name');
  const iApplicationLevel = ILine.IndexOf('Application Level');
  const iAgglomeratorCategories = ILine.IndexOf('Agglomerator Categories');
  const iBriefCostDescription = ILine.IndexOf('Brief cost description');
  const iCostDescription = ILine.IndexOf('Cost Description');
  const iCostTrigger = ILine.IndexOf('Cost Trigger (if blank then 1)');
  const iStartYear_SmallSys = ILine.IndexOf('Start_Year_SmallSys');
  const iRepeat_SmallSys = ILine.IndexOf('Repeat_SmallSys');
  const iStartYear_NonSmallSys = ILine.IndexOf('Start_Year_NonSmallSys');
  const iRepeat_NonSmallSys = ILine.IndexOf('Repeat_NonSmallSys');
  const iFormula = ILine.IndexOf('Formula');
  const iNotes = ILine.IndexOf('Notes');

  var Expressions: TExpressionsRecord;
  var i := 1;
  while not T.EndOfStream do begin
    inc(i);
    ILine.CommaText := T.ReadLine;
    Expressions := Default(TExpressionsRecord);
    Expressions.IDNum := ILine[iIDNum].toInteger;
    Expressions.CostName := ILine[iCostName];
    Expressions.ApplicationLevel := ILine[iApplicationLevel];
    Expressions.AgglomeratorCategories := ILine[iAgglomeratorCategories];
    Expressions.BriefCostDescription := ILine[iBriefCostDescription];
    Expressions.CostDescription := ILine[iCostDescription];
    Expressions.CostTrigger := lowercase(ILine[iCostTrigger]);
    Expressions.StartYear_SmallSys := ILine[iStartYear_SmallSys];
    Expressions.Repeat_SmallSys := ILine[iRepeat_SmallSys];
    Expressions.StartYear_NonSmallSys := ILine[iStartYear_NonSmallSys];
    Expressions.Repeat_NonSmallSys := ILine[iRepeat_NonSmallSys];
    Expressions.Formula := lowercase(ILine[iFormula]);
    Expressions.Notes := ILine[iNotes];
    Expressions.RowNum := i;

    //skip blanks and zeroes
    if  not ((Expressions.Formula = '0') or (Expressions.Formula = '')) then begin
      var id := fExpressionsRecords.Add(Expressions);
      ParseYears(id, 0 , Expressions.StartYear_SmallSys, Expressions.Repeat_SmallSys);
      ParseYears(id, 1 , Expressions.StartYear_NonSmallSys, Expressions.Repeat_NonSmallSys);
    end;
  end;
  ILine.Free;
  T.Free;
end;

procedure TExpressionsRecords.ParseYears(id, sz: integer; sy, ry: string);
begin
  var s := pos('-',sy);
  var sty, eny, rpy: integer;
  if s>1 then begin
    sty := strtoint(copy(sy,1,s-1));
    eny := strtoint(copy(sy,s+1,1000));
  end else begin
    sty := strtoint(sy);
    eny := sty;
  end;
  for var i := sty to eny do begin
    //Note: Years are change to be zero based
   fCCCalcYearStore[sz,i-1,id] := true;
  end;
  rpy := 0;
  if not TryStrToInt(ry,rpy) then rpy:=0;
  if rpy = 0 then exit;

  for var i := eny + 1 to 201 do begin
    var m := (i - eny) mod rpy;
```

```
158     if m = 0  then
159       fCCCalcYearStore[sz,i-1,id] := true;
160   end;
161
162 end;
163
164 procedure TExpressionsRecords.WriteToFile(FName: string);
165 begin
166
167 end;
168
169
170 end.
171
```

PrecompileCosts\DataNeedsCsv.pas

```
1  unit DataNeedsCsv;
2
3  interface
4
5  uses Classes, SysUtils, Generics.Collections;
6
7  type
8    TDataRecord=record
9      Name : string;
10     Description : string;
11     Source : string;
12     ss0,ss101,ss501,ss1001,ss3301,ss10k,ss50k, ss100k, ss1m : double;
13     gwss0,gwss101,gwss501,gwss1001,gwss3301,gwss10k,gwss50k, gwss100k,gwss1m : double;
14      //procedure LoadFromStream(AStream : TStream);
15      //procedure SaveToStream(AStream : TStream);
16   end;
17
18   TSPFASRecord=record
19     PWSID : string;
20   end;
21
22   TDataRecords = class(TObject)
23   private
24     function getDataRecord(I: Integer): TDataRecord;
25     function getDataRecordCount: Integer;
26   public
27     fDataRecords : TList<TDataRecord>;
28     fStatePFASRecords : TList<TSPFASRecord>;
29
30
31     constructor Create();
32     destructor Destroy();override;
33
34     procedure LoadFromFile(FName: string; ClearRecs:boolean = true);
35     procedure WriteToFile(FName: string);
36     procedure LoadSPFAS(FName: string; ClearRecs:boolean = true);
37
38     property DataRecord[I: Integer]: TDataRecord read getDataRecord;
39     property Count: Integer read getDataRecordCount;
40   end;
41
42 implementation
43
44 { TDataRecords }
45
46 constructor TDataRecords.Create;
47 begin
48   fDataRecords := TList<TDataRecord>.create;
49 end;
50
51 destructor TDataRecords.Destroy;
52 begin
53   fDataRecords.Free;
54   inherited;
55 end;
56
```

```
57  function TDataRecords.getDataRecord(I: Integer): TDataRecord;
58  begin
59    Result := fDataRecords.items[i];
60  end;
61
62  function TDataRecords.getDataRecordCount: Integer;
63  begin
64    Result := fDataRecords.count;
65  end;
66
67  procedure TDataRecords.LoadFromFile(FName: string; ClearRecs:boolean = true);
68  var
69    T: TStreamReader;
70
71    function ifBlnk(aVal: string) : string;
72    begin
73      if aVal = '' then ifBlnk := '-9999'
74      else ifBlnk := aVal;
75      if (length(aVal)>0) and (aVal[1] = '$') then ifBlnk := copy(aVAL,2,length(aVal)-1)
76      else ifBlnk := aVal;
77    end;
78  begin
79    if ClearRecs then begin
80      fDataRecords.Clear;
81    end;
82
83    T := TStreamReader.Create(FName,TEncoding.ANSI,false,4096);
84    var ILine := TStringList.Create;
85    ILine.Duplicates := dupIgnore;
86    ILine.StrictDelimiter := true;
87    ILine.CommaText := T.ReadLine;
88    const iName = ILine.IndexOf('Data Name');
89    const iDescription = ILine.IndexOf('Description');
90    const iSource = ILine.IndexOf('Source');
91
92    const iss0 = ILine.IndexOf('sw_ss0_100');
93    const iss101 = ILine.IndexOf('sw_ss101_500');
94    const iss501 = ILine.IndexOf('sw_ss501_1000');
95    const iss1001 = ILine.IndexOf('sw_ss1001_3300');
96    const iss3301 = ILine.IndexOf('sw_ss3301_10k');
97    const iss10k = ILine.IndexOf('sw_ss10k_50k');
98    const iss50k = ILine.IndexOf('sw_ss50k_100k');
99    const iss100k = ILine.IndexOf('sw_ss100k_1m');
100   const iss1MP = ILine.IndexOf('sw_ss1Mp');
101
102   const igwss0 = ILine.IndexOf('gw_ss0_100');
103   const igwss101 = ILine.IndexOf('gw_ss101_500');
104   const igwss501 = ILine.IndexOf('gw_ss501_1000');
105   const igwss1001 = ILine.IndexOf('gw_ss1001_3300');
106   const igwss3301 = ILine.IndexOf('gw_ss3301_10k');
107   const igwss10k = ILine.IndexOf('gw_ss10k_50k');
108   const igwss50k = ILine.IndexOf('gw_ss50k_100k');
109   const igwss100k = ILine.IndexOf('gw_ss100k_1m');
110   const igwss1MP = ILine.IndexOf('gw_ss1Mp');
111
112   var DataRec: TDataRecord;
113   var i := 1;
114   while not T.EndOfStream do begin
115     inc(i);
116     ILine.CommaText := T.ReadLine;
117     if ILine.Count<2 then continue;
118     DataRec := Default(TDataRecord);
119
120     DataRec.Name := ILine[iName];
121     //DataRec.Description := ILine[iDescription];
122     //DataRec.Source := ILine[iSource];
123
124     DataRec.ss0 := ifBlnk(ILine[iss0]).ToDouble();
125     DataRec.ss101 := ifBlnk(ILine[iss101]).ToDouble();
126     DataRec.ss501 := ifBlnk(ILine[iss501]).ToDouble();
127     DataRec.ss1001 := ifBlnk(ILine[iss1001]).ToDouble();
128     DataRec.ss3301 := ifBlnk(ILine[iss3301]).ToDouble();
129     DataRec.ss10k := ifBlnk(ILine[iss10k]).ToDouble();
130     DataRec.ss50k := ifBlnk(ILine[iss50k]).ToDouble();
131     DataRec.ss100k := ifBlnk(ILine[iss100k]).ToDouble();
```

```
    DataRec.ss1m := ifBlnk(ILine[iss1mp]).ToDouble();

    DataRec.gwss0 := ifBlnk(ILine[igwss0]).ToDouble();
    DataRec.gwss101 := ifBlnk(ILine[igwss101]).ToDouble();
    DataRec.gwss501 := ifBlnk(ILine[igwss501]).ToDouble();
    DataRec.gwss1001 := ifBlnk(ILine[igwss1001]).ToDouble();
    DataRec.gwss3301 := ifBlnk(ILine[igwss3301]).ToDouble();
    DataRec.gwss10k := ifBlnk(ILine[igwss10k]).ToDouble();
    DataRec.gwss50k := ifBlnk(ILine[igwss50k]).ToDouble();
    DataRec.gwss100k := ifBlnk(ILine[igwss100k]).ToDouble();
    DataRec.gwss1m := ifBlnk(ILine[igwss1mp]).ToDouble();

    var id := fDataRecords.Add(DataRec);
  end;
  ILine.Free;
  T.Free;
end;

procedure TDataRecords.LoadSPFAS(FName: string; ClearRecs: boolean);
var
  T: TStreamReader;
begin
  if ClearRecs then begin
    fStatePFASRecords.Clear;
  end;

  T := TStreamReader.Create(FName,TEncoding.ANSI,false,4096);
  var ILine := TStringList.Create;
  ILine.Duplicates := dupIgnore;
  ILine.StrictDelimiter := true;
  ILine.CommaText := T.ReadLine;
  ILine.CommaText := T.ReadLine;
  const iPWSID = ILine.IndexOf('PWSID');

  var DataRec: TSPFASRecord;
  var i := 1;
  while not T.EndOfStream do begin
    inc(i);
    ILine.CommaText := T.ReadLine;
    DataRec := Default(TSPFASRecord);
    DataRec.PWSID := ILine[iPWSID];

    var id := fStatePFASRecords.Add(DataRec);
  end;
  ILine.Free;
  T.Free;
end;

procedure TDataRecords.WriteToFile(FName: string);
begin

end;

end.

```

PrecompileCosts\LCRCompiledCost.pas

```
unit LCRCompiledCost;

//Autogenerated unit -
//Generated - 3/22/2022 12:49:33 PM
//Baseline Workbook - K:\PrecompileCosts\data\Expressions_v3.csv
//Found variables in workbook - 40
//Found costing steps - 59
//Option Workbook - (*WORKBOOKOPTION*)
//Found variables in workbook - (*NUMVARSOPTION*)
//Found costing steps - 9998

interface

uses Math, SysUtils;

```

```
16 type
17   TLSRValueStore = array[0..59] of double;
18   TLSRValueStoreFlag = array[0..59] of boolean;
19   TLSRCalcYearStore = array[0..100,0..59] of boolean;
20
21   _VariablesRecBASELINE = record
22     b_exceed,b_nondetect,
23     b_nondetect_all_ep,b_treat_nondetect,epa533,epa533_cost,epa537,epa537_cost,
24     hrs_pa_adopt_rule,hrs_pa_initial_ta,hrs_pa_report_ep,hrs_pa_train,hrs_pa_treat,hrs_pa_waiver,
25     hrs_samp,hrs_sdwis,hrs_sys_adopt_rule,hrs_sys_initial_ta,hrs_sys_report_ep,hrs_sys_treat,
26     hrs_sys_waiver,hrs_tier2_approve,hrs_tier2_certify,hrs_tier2_distribute,hrs_tier2_template,hrs_tier3_approve,
27     hrs_tier3_certify,hrs_tier3_distribute,hrs_tier3_template,labor_pa_rate,labor_rate,labor_sys_rate,
28     numb_hh,numb_initial_samples,numb_nondetect_samples,statepfas,tier2_cost,tier3_cost,
29     ucmr5,wbs_model : double;
30
31     P_b_exceed,P_b_nondetect,
32     P_b_nondetect_all_ep,P_b_treat_nondetect,P_epa533,P_epa533_cost,P_epa537,P_epa537_cost,
33     P_hrs_pa_adopt_rule,P_hrs_pa_initial_ta,P_hrs_pa_report_ep,P_hrs_pa_train,P_hrs_pa_treat,P_hrs_pa_waiver,
34     P_hrs_samp,P_hrs_sdwis,P_hrs_sys_adopt_rule,P_hrs_sys_initial_ta,P_hrs_sys_report_ep,P_hrs_sys_treat,
35     P_hrs_sys_waiver,P_hrs_tier2_approve,P_hrs_tier2_certify,P_hrs_tier2_distribute,P_hrs_tier2_template,P_hrs_tier3_approve,
36     P_hrs_tier3_certify,P_hrs_tier3_distribute,P_hrs_tier3_template,P_labor_pa_rate,P_labor_rate,P_labor_sys_rate,
37     P_numb_hh,P_numb_initial_samples,P_numb_nondetect_samples,P_statepfas,P_tier2_cost,P_tier3_cost,
38     P_ucmr5,P_wbs_model : pdouble;
39   end;
40   _VariablesRecOPTION = record
41 (*VARIABLESOPTION*)
42   end;
43
44   TLSRCompiledCost = class
45     _CalcCost,_Cost,_Hours,_OM,_Labor  : TLSRValueStore;
46     _ImAState : TLSRValueStoreFlag;
47     _YearOK : TLSRCalcYearStore;
48     TotEval : int64;
49     constructor create();
50     procedure _Reset(aState : boolean);
51     procedure _LoadVars; virtual; abstract;
52     procedure _SetVarPointer(const name : string; const pd : PDouble); virtual; abstract;
53     procedure _Evaluate(const Yr : integer); virtual; abstract;
54     procedure _EvaluateState(const Yr: integer); virtual; abstract;
55     function _DumpVars : string; virtual; abstract;
56
57   end;
58
59   TLSRCompiledCostBaseline = class(TLSRCompiledCost)
60     _Variables : _VariablesRecBaseline;
61
62     constructor create();
63     procedure _LoadVars; override;
64     procedure _SetVarPointer(const name : string; const pd : PDouble); override;
65     procedure _Evaluate(const Yr : integer); override;
66     procedure _EvaluateState(const Yr: integer); override;
67
68     function _DumpVars : string; override;
69   end;
70
71   TLSRCompiledCostOption = class(TLSRCompiledCost)
72     _Variables : _VariablesRecOption;
73
74     constructor create();
75     procedure _LoadVars; override;
76     procedure _SetVarPointer(const name : string; const pd : PDouble); override;
77     procedure _Evaluate(const Yr : integer); override;
78     procedure _EvaluateState(const Yr: integer); override;
79
80     function _DumpVars : string; override;
81   end;
82
83
84 const
85   _CWorkBookBASELINE = 'K:\PrecompileCosts\data\Expressions_v3.csv';
86   _CWorkBookOPTION = '(*WORKBOOKOPTION*)';
87   _CWorkBookDate = '3/22/2022 12:49:33 PM';
88
89 var
90   _UseCompiled : boolean = false;
```

```
91
92 implementation
93
94 { TLSRCompiledCost }
95
96 constructor TLSRCompiledCost.create;
97 begin
98   TotEval:=0;
99 end;
100
101 procedure TLSRCompiledCost._Reset(aState : boolean);
102 var i : integer;
103 begin
104   for i:=0 to high(_CalcCost) do begin
105     if not (_ImAState[i] = aState) then continue;
106     _CalcCost[i]:=0;
107     _Cost[i]:=0;
108     _Hours[i]:=0;
109     _OM[i]:=0;
110     _Labor[i]:=0;
111   end;
112 end;
113
114 { TLSRCompiledCostBaseline }
115
116 procedure TLSRCompiledCostBaseline._LoadVars;
117 begin
118   if Assigned(_Variables.P_b_exceed) then _Variables.b_exceed:= _Variables.P_b_exceed^;
119   if Assigned(_Variables.P_b_nondetect) then _Variables.b_nondetect:= _Variables.P_b_nondetect^;
120   if Assigned(_Variables.P_b_nondetect_all_ep) then _Variables.b_nondetect_all_ep:= _Variables.P_b_nondetect_all_ep^;
121   if Assigned(_Variables.P_b_treat_nondetect) then _Variables.b_treat_nondetect:= _Variables.P_b_treat_nondetect^;
122   if Assigned(_Variables.P_epa533) then _Variables.epa533:= _Variables.P_epa533^;
123   if Assigned(_Variables.P_epa533_cost) then _Variables.epa533_cost:= _Variables.P_epa533_cost^;
124   if Assigned(_Variables.P_epa537) then _Variables.epa537:= _Variables.P_epa537^;
125   if Assigned(_Variables.P_epa537_cost) then _Variables.epa537_cost:= _Variables.P_epa537_cost^;
126   if Assigned(_Variables.P_hrs_pa_adopt_rule) then _Variables.hrs_pa_adopt_rule:= _Variables.P_hrs_pa_adopt_rule^;
127   if Assigned(_Variables.P_hrs_pa_initial_ta) then _Variables.hrs_pa_initial_ta:= _Variables.P_hrs_pa_initial_ta^;
128   if Assigned(_Variables.P_hrs_pa_report_ep) then _Variables.hrs_pa_report_ep:= _Variables.P_hrs_pa_report_ep^;
129   if Assigned(_Variables.P_hrs_pa_train) then _Variables.hrs_pa_train:= _Variables.P_hrs_pa_train^;
130   if Assigned(_Variables.P_hrs_pa_treat) then _Variables.hrs_pa_treat:= _Variables.P_hrs_pa_treat^;
131   if Assigned(_Variables.P_hrs_pa_waiver) then _Variables.hrs_pa_waiver:= _Variables.P_hrs_pa_waiver^;
132   if Assigned(_Variables.P_hrs_samp) then _Variables.hrs_samp:= _Variables.P_hrs_samp^;
133   if Assigned(_Variables.P_hrs_sdwis) then _Variables.hrs_sdwis:= _Variables.P_hrs_sdwis^;
134   if Assigned(_Variables.P_hrs_sys_adopt_rule) then _Variables.hrs_sys_adopt_rule:= _Variables.P_hrs_sys_adopt_rule^;
135   if Assigned(_Variables.P_hrs_sys_initial_ta) then _Variables.hrs_sys_initial_ta:= _Variables.P_hrs_sys_initial_ta^;
136   if Assigned(_Variables.P_hrs_sys_report_ep) then _Variables.hrs_sys_report_ep:= _Variables.P_hrs_sys_report_ep^;
137   if Assigned(_Variables.P_hrs_sys_treat) then _Variables.hrs_sys_treat:= _Variables.P_hrs_sys_treat^;
138   if Assigned(_Variables.P_hrs_sys_waiver) then _Variables.hrs_sys_waiver:= _Variables.P_hrs_sys_waiver^;
139   if Assigned(_Variables.P_hrs_tier2_approve) then _Variables.hrs_tier2_approve:= _Variables.P_hrs_tier2_approve^;
140   if Assigned(_Variables.P_hrs_tier2_certify) then _Variables.hrs_tier2_certify:= _Variables.P_hrs_tier2_certify^;
141   if Assigned(_Variables.P_hrs_tier2_distribute) then _Variables.hrs_tier2_distribute:= _Variables.P_hrs_tier2_distribute^;
142   if Assigned(_Variables.P_hrs_tier2_template) then _Variables.hrs_tier2_template:= _Variables.P_hrs_tier2_template^;
143   if Assigned(_Variables.P_hrs_tier3_approve) then _Variables.hrs_tier3_approve:= _Variables.P_hrs_tier3_approve^;
144   if Assigned(_Variables.P_hrs_tier3_certify) then _Variables.hrs_tier3_certify:= _Variables.P_hrs_tier3_certify^;
145   if Assigned(_Variables.P_hrs_tier3_distribute) then _Variables.hrs_tier3_distribute:= _Variables.P_hrs_tier3_distribute^;
146   if Assigned(_Variables.P_hrs_tier3_template) then _Variables.hrs_tier3_template:= _Variables.P_hrs_tier3_template^;
147   if Assigned(_Variables.P_labor_pa_rate) then _Variables.labor_pa_rate:= _Variables.P_labor_pa_rate^;
148   if Assigned(_Variables.P_labor_rate) then _Variables.labor_rate:= _Variables.P_labor_rate^;
149   if Assigned(_Variables.P_labor_sys_rate) then _Variables.labor_sys_rate:= _Variables.P_labor_sys_rate^;
150   if Assigned(_Variables.P_numb_hh) then _Variables.numb_hh:= _Variables.P_numb_hh^;
151   if Assigned(_Variables.P_numb_initial_samples) then _Variables.numb_initial_samples:= _Variables.P_numb_initial_samples^;
152   if Assigned(_Variables.P_numb_nondetect_samples) then _Variables.numb_nondetect_samples:= _Variables.P_numb_nondetect_samples^;
153   if Assigned(_Variables.P_statepfas) then _Variables.statepfas:= _Variables.P_statepfas^;
154   if Assigned(_Variables.P_tier2_cost) then _Variables.tier2_cost:= _Variables.P_tier2_cost^;
155   if Assigned(_Variables.P_tier3_cost) then _Variables.tier3_cost:= _Variables.P_tier3_cost^;
156   if Assigned(_Variables.P_ucmr5) then _Variables.ucmr5:= _Variables.P_ucmr5^;
157   if Assigned(_Variables.P_wbs_model) then _Variables.wbs_model:= _Variables.P_wbs_model^;
158
159 end;
160
161 procedure TLSRCompiledCostBaseline._Evaluate(const Yr : integer);
162 begin
163   _Reset(False);
164   _LoadVars();
```

```
165   with _Variables do begin
166
167     //system_read  row: 0
168     _Cost[0] := 1*(hrs_sys_adopt_rule*labor_sys_rate);
169       inc(TotEval);
170
171     //system_implement_train  row: 1
172     _Cost[1] := 1*(hrs_sys_initial_ta*labor_sys_rate);
173       inc(TotEval);
174
175     //pa_adopt  row: 2
176     _Cost[2] := 1*(hrs_pa_adopt_rule*labor_pa_rate);
177       inc(TotEval);
178
179     //pa_implement_train  row: 3
180     _Cost[3] := 1*(hrs_pa_initial_ta*labor_pa_rate);
181       inc(TotEval);
182
183     //pa_internal_train  row: 4
184     _Cost[4] := 1*(hrs_pa_train*labor_pa_rate);
185       inc(TotEval);
186
187     //pa_sdwis  row: 5
188     _Cost[5] := 1*(hrs_sdwis*labor_pa_rate);
189       inc(TotEval);
190
191     //sys_initial_monitor_EPA533  row: 6
192     _CalcCost[6] := EPA533*((1-UCMR5)+(1-STATEPFAS));
193     inc(TotEval);
194     if _CalcCost[6] > 0 then begin
195       _Cost[6] := 4*(hrs_samp*labor_sys_rate+EPA533_cost*numb_initial_samples);
196       inc(TotEval);
197       end;
198
199     //sys_initial_monitor_EPA537  row: 7
200     _CalcCost[7] := EPA537*((1-UCMR5)+(1-STATEPFAS));
201     inc(TotEval);
202     if _CalcCost[7] > 0 then begin
203       _Cost[7] := 4*(hrs_samp*labor_sys_rate+EPA537_cost*numb_initial_samples);
204       inc(TotEval);
205       end;
206
207     //sys_report_initial_monitor  row: 8
208     _CalcCost[8] := (1-UCMR5)+(1-STATEPFAS);
209     inc(TotEval);
210     if _CalcCost[8] > 0 then begin
211       _Cost[8] := 4*(hrs_sys_report_ep*labor_sys_rate);
212       inc(TotEval);
213       end;
214
215     //pa_review_initial_monitor  row: 9
216     _CalcCost[9] := (1-UCMR5)+(1-STATEPFAS);
217     inc(TotEval);
218     if _CalcCost[9] > 0 then begin
219       _Cost[9] := 4*(hrs_pa_report_ep*labor_pa_rate);
220       inc(TotEval);
221       end;
222
223     //sys_initial_confirm_EPA533  row: 10
224     _CalcCost[10] := EPA533*b_exceed*((1-UCMR5)+(1-STATEPFAS));
225     inc(TotEval);
226     if _CalcCost[10] > 0 then begin
227       _Cost[10] := 1*(hrs_samp*labor_sys_rate+EPA533_cost);
228       inc(TotEval);
229       end;
230
231     //sys_initial_confirm_EPA537  row: 11
232     _CalcCost[11] := EPA537*b_exceed*((1-UCMR5)+(1-STATEPFAS));
233     inc(TotEval);
234     if _CalcCost[11] > 0 then begin
235       _Cost[11] := 1*(hrs_samp*labor_sys_rate+EPA537_cost);
236       inc(TotEval);
237       end;
238
239     //sys_report_confirm_monitor  row: 12
```

```
240     _CalcCost[12] := b_exceed*(1-UCMR5)+(1-STATEPFAS);
241     inc(TotEval);
242     if _CalcCost[12] > 0 then begin
243       _Cost[12] := 1*(hrs_sys_report_ep*labor_sys_rate);
244       inc(TotEval);
245       end;
246
247     //pa_review_confirm_monitor  row: 13
248     _CalcCost[13] := b_exceed*((1-UCMR5)+(1-STATEPFAS));
249     inc(TotEval);
250     if _CalcCost[13] > 0 then begin
251       _Cost[13] := 1*(hrs_pa_report_ep*labor_pa_rate);
252       inc(TotEval);
253       end;
254
255     //sys_imp_monitor_nondetect_EPA533  row: 14
256     _CalcCost[14] := EPA533*b_nondetect*(UCMR5+STATEPFAS);
257     inc(TotEval);
258     if _CalcCost[14] > 0 then begin
259       _Cost[14] := 1*(hrs_samp*labor_sys_rate+EPA533_cost*numb_nondetect_samples);
260       inc(TotEval);
261       end;
262
263     //sys_imp_monitor_nondetect_EPA537  row: 15
264     _CalcCost[15] := EPA537*b_nondetect*(UCMR5+STATEPFAS);
265     inc(TotEval);
266     if _CalcCost[15] > 0 then begin
267       _Cost[15] := 1*(hrs_samp*labor_sys_rate+EPA537_cost*numb_nondetect_samples);
268       inc(TotEval);
269       end;
270
271     //sys_report_imp_monitor_nondetect  row: 16
272     _CalcCost[16] := b_nondetect*(UCMR5+STATEPFAS);
273     inc(TotEval);
274     if _CalcCost[16] > 0 then begin
275       _Cost[16] := 1*(hrs_sys_report_ep*labor_sys_rate);
276       inc(TotEval);
277       end;
278
279     //pa_review_imp_monitor_nondetect  row: 17
280     _CalcCost[17] := b_nondetect*(UCMR5+STATEPFAS);
281     inc(TotEval);
282     if _CalcCost[17] > 0 then begin
283       _Cost[17] := 1*(hrs_pa_report_ep*labor_pa_rate);
284       inc(TotEval);
285       end;
286
287     //sys_imp_monitor_detect_EPA533  row: 18
288     _CalcCost[18] := EPA533*(1-(b_exceed+b_nondetect));
289     inc(TotEval);
290     if _CalcCost[18] > 0 then begin
291       _Cost[18] := 1*(hrs_samp*labor_sys_rate+EPA533_cost);
292       inc(TotEval);
293       end;
294
295     //sys_imp_monitor_detect_EPA537  row: 19
296     _CalcCost[19] := EPA537*(1-(b_exceed+b_nondetect));
297     inc(TotEval);
298     if _CalcCost[19] > 0 then begin
299       _Cost[19] := 1*(hrs_samp*labor_sys_rate+EPA537_cost);
300       inc(TotEval);
301       end;
302
303     //sys_report_imp_monitor_detect  row: 20
304     _CalcCost[20] := 1-(b_exceed+b_nondetect);
305     inc(TotEval);
306     if _CalcCost[20] > 0 then begin
307       _Cost[20] := 1*(hrs_sys_report_ep*labor_sys_rate);
308       inc(TotEval);
309       end;
310
311     //pa_review_imp_monitor_detect  row: 21
312     _CalcCost[21] := 1-(b_exceed+b_nondetect);
313     inc(TotEval);
```

```
314     if _CalcCost[21] > 0 then begin
315       _Cost[21] := 1*(hrs_pa_report_ep*labor_pa_rate);
316       inc(TotEval);
317       end;
318
319     //sys_imp_monitor_exceed_EPA533  row: 22
320     _CalcCost[22] := EPA533*b_exceed;
321     inc(TotEval);
322     if _CalcCost[22] > 0 then begin
323       _Cost[22] := 4*(hrs_samp*labor_sys_rate+EPA533_cost);
324       inc(TotEval);
325       end;
326
327     //sys_imp_monitor_exceed_EPA537  row: 23
328     _CalcCost[23] := EPA537*b_exceed;
329     inc(TotEval);
330     if _CalcCost[23] > 0 then begin
331       _Cost[23] := 4*(hrs_samp*labor_sys_rate+EPA537_cost);
332       inc(TotEval);
333       end;
334
335     //sys_report_imp_monitor_exceed  row: 24
336     _CalcCost[24] := b_exceed;
337     inc(TotEval);
338     if _CalcCost[24] > 0 then begin
339       _Cost[24] := 4*(hrs_sys_report_ep*labor_sys_rate);
340       inc(TotEval);
341       end;
342
343     //pa_review_imp_monitor_exceed  row: 25
344     _CalcCost[25] := b_exceed;
345     inc(TotEval);
346     if _CalcCost[25] > 0 then begin
347       _Cost[25] := 4*(hrs_pa_report_ep*labor_pa_rate);
348       inc(TotEval);
349       end;
350
351     //sys_stand_monitor_nondetect_EPA533  row: 26
352     _CalcCost[26] := EPA533*(b_nondetect+b_treat_nondetect);
353     inc(TotEval);
354     if _CalcCost[26] > 0 then begin
355       _Cost[26] := 1*(hrs_samp*labor_sys_rate+EPA533_cost*numb_nondetect_samples);
356       inc(TotEval);
357       end;
358
359     //sys_stand_monitor_nondetect_EPA537  row: 27
360     _CalcCost[27] := EPA537*(b_nondetect+b_treat_nondetect);
361     inc(TotEval);
362     if _CalcCost[27] > 0 then begin
363       _Cost[27] := 1*(hrs_samp*labor_sys_rate+EPA537_cost*numb_nondetect_samples);
364       inc(TotEval);
365       end;
366
367     //sys_report_stand_monitor_nondetect  row: 28
368     _CalcCost[28] := b_nondetect+b_treat_nondetect;
369     inc(TotEval);
370     if _CalcCost[28] > 0 then begin
371       _Cost[28] := 1*(hrs_sys_report_ep*labor_sys_rate);
372       inc(TotEval);
373       end;
374
375     //pa_review_stand_monitor_nondetect  row: 29
376     _CalcCost[29] := b_nondetect+b_treat_nondetect;
377     inc(TotEval);
378     if _CalcCost[29] > 0 then begin
379       _Cost[29] := 1*(hrs_pa_report_ep*labor_pa_rate);
380       inc(TotEval);
381       end;
382
383     //sys_stand_monitor_detect_EPA533  row: 30
384     _CalcCost[30] := EPA533*(1-(b_treat_nondetect+b_nondetect));
385     inc(TotEval);
386     if _CalcCost[30] > 0 then begin
387       _Cost[30] := 1*(hrs_samp*labor_sys_rate+EPA533_cost);
388       inc(TotEval);
```

```
      end;

    //sys_stand_monitor_detect_EPA537  row: 31
    _CalcCost[31] := EPA537*(1-(b_treat_nondetect+b_nondetect));
    inc(TotEval);
    if _CalcCost[31] > 0 then begin
      _Cost[31] := 1*(hrs_samp*labor_sys_rate+EPA537_cost);
      inc(TotEval);
      end;

    //sys_report_stand_monitor_detect  row: 32
    _CalcCost[32] := 1-(b_treat_nondetect+b_nondetect);
    inc(TotEval);
    if _CalcCost[32] > 0 then begin
      _Cost[32] := 1*(hrs_sys_report_ep*labor_sys_rate);
      inc(TotEval);
      end;

    //pa_review_stand_monitor_detect  row: 33
    _CalcCost[33] := 1-(b_treat_nondetect+b_nondetect);
    inc(TotEval);
    if _CalcCost[33] > 0 then begin
      _Cost[33] := 1*(hrs_pa_report_ep*labor_pa_rate);
      inc(TotEval);
      end;

    //sys_stand_monitor_exceed_EPA533  row: 34
    _CalcCost[34] := EPA533*b_exceed;
    inc(TotEval);
    if _CalcCost[34] > 0 then begin
      _Cost[34] := 4*(hrs_samp*labor_sys_rate+EPA533_cost);
      inc(TotEval);
      end;

    //sys_stand_monitor_exceed_EPA537  row: 35
    _CalcCost[35] := EPA537*b_exceed;
    inc(TotEval);
    if _CalcCost[35] > 0 then begin
      _Cost[35] := 4*(hrs_samp*labor_sys_rate+EPA537_cost);
      inc(TotEval);
      end;

    //sys_report_stand_monitor_exceed  row: 36
    _CalcCost[36] := b_exceed;
    inc(TotEval);
    if _CalcCost[36] > 0 then begin
      _Cost[36] := 4*(hrs_sys_report_ep*labor_sys_rate);
      inc(TotEval);
      end;

    //pa_review_stand_monitor_exceed  row: 37
    _CalcCost[37] := b_exceed;
    inc(TotEval);
    if _CalcCost[37] > 0 then begin
      _Cost[37] := 4*(hrs_pa_report_ep*labor_pa_rate);
      inc(TotEval);
      end;

    //sys_monitor_waiver  row: 38
    _CalcCost[38] := b_nondetect_all_ep;
    inc(TotEval);
    if _CalcCost[38] > 0 then begin
      _Cost[38] := 1*(hrs_sys_waiver*labor_rate);
      inc(TotEval);
      end;

    //pa_monitor_waiver  row: 39
    _CalcCost[39] := b_nondetect_all_ep;
    inc(TotEval);
    if _CalcCost[39] > 0 then begin
      _Cost[39] := 1*(hrs_pa_waiver*labor_pa_rate);
      inc(TotEval);
      end;
```

```
463     //sys_waiver_sample_EPA533  row: 40
464     _CalcCost[40] := EPA533*b_nondetect_all_ep;
465     inc(TotEval);
466     if _CalcCost[40] > 0 then begin
467       _Cost[40] := 1*(hrs_samp*labor_sys_rate+EPA533_cost);
468       inc(TotEval);
469       end;
470
471     //sys_waiver_sample_EPA537  row: 41
472     _CalcCost[41] := EPA537*b_nondetect_all_ep;
473     inc(TotEval);
474     if _CalcCost[41] > 0 then begin
475       _Cost[41] := 1*(hrs_samp*labor_sys_rate+EPA537_cost);
476       inc(TotEval);
477       end;
478
479     //sys_waiver_sample_report  row: 42
480     _CalcCost[42] := b_nondetect_all_ep;
481     inc(TotEval);
482     if _CalcCost[42] > 0 then begin
483       _Cost[42] := 1*(hrs_sys_report_ep*labor_sys_rate);
484       inc(TotEval);
485       end;
486
487     //pa_waiver_sample_report  row: 43
488     _CalcCost[43] := b_nondetect_all_ep;
489     inc(TotEval);
490     if _CalcCost[43] > 0 then begin
491       _Cost[43] := 1*(hrs_pa_report_ep*labor_pa_rate);
492       inc(TotEval);
493       end;
494
495     //sys_permit_application  row: 44
496     _CalcCost[44] := b_exceed;
497     inc(TotEval);
498     if _CalcCost[44] > 0 then begin
499       _Cost[44] := 1*(hrs_sys_treat*labor_sys_rate);
500       inc(TotEval);
501       end;
502
503     //pa_permit_application  row: 45
504     _CalcCost[45] := b_exceed;
505     inc(TotEval);
506     if _CalcCost[45] > 0 then begin
507       _Cost[45] := 1*(hrs_pa_treat*labor_pa_rate);
508       inc(TotEval);
509       end;
510
511     //sys_treatment_install  row: 46
512     _CalcCost[46] := b_exceed;
513     inc(TotEval);
514     if _CalcCost[46] > 0 then begin
515       _Cost[46] := 1*(WBS_model);
516       inc(TotEval);
517       end;
518
519     //pa_template_tier2  row: 47
520     _CalcCost[47] := b_exceed;
521     inc(TotEval);
522     if _CalcCost[47] > 0 then begin
523       _Cost[47] := 1*(hrs_tier2_template*labor_pa_rate);
524       inc(TotEval);
525       end;
526
527     //sys_template_tier2  row: 48
528     _CalcCost[48] := b_exceed;
529     inc(TotEval);
530     if _CalcCost[48] > 0 then begin
531       _Cost[48] := 1*(hrs_tier2_template*labor_sys_rate);
532       inc(TotEval);
533       end;
534
535     //pa_approve_tier2  row: 49
536     _CalcCost[49] := b_exceed;
537     inc(TotEval);
```

```
538     if _CalcCost[49] > 0 then begin
539       _Cost[49] := 1*(hrs_tier2_approve*labor_pa_rate);
540       inc(TotEval);
541       end;
542
543     //pws_distribute_tier2  row: 50
544     _CalcCost[50] := b_exceed;
545     inc(TotEval);
546     if _CalcCost[50] > 0 then begin
547       _Cost[50] := 4*(numb_hh*hrs_tier2_distribute*labor_sys_rate+numb_hh*tier2_cost);
548       inc(TotEval);
549       end;
550
551     //pws_report_tier2  row: 51
552     _CalcCost[51] := b_exceed;
553     inc(TotEval);
554     if _CalcCost[51] > 0 then begin
555       _Cost[51] := 1*(hrs_tier2_certify*labor_sys_rate);
556       inc(TotEval);
557       end;
558
559     //pa_review_tier2  row: 52
560     _CalcCost[52] := b_exceed;
561     inc(TotEval);
562     if _CalcCost[52] > 0 then begin
563       _Cost[52] := 1*(hrs_tier2_certify*labor_pa_rate);
564       inc(TotEval);
565       end;
566
567     //pa_template_tier3  row: 53
568     _Cost[53] := 1*(hrs_tier3_template*labor_pa_rate);
569       inc(TotEval);
570
571     //sys_template_tier3  row: 54
572     _Cost[54] := 1*(hrs_tier3_template*labor_sys_rate);
573       inc(TotEval);
574
575     //pa_approve_tier3  row: 55
576     _Cost[55] := 1*(hrs_tier3_approve*labor_pa_rate);
577       inc(TotEval);
578
579     //pws_distribute_tier3  row: 56
580     _Cost[56] := 1*(hrs_tier3_distribute*labor_sys_rate+tier3_cost);
581       inc(TotEval);
582
583     //pws_report_tier3  row: 57
584     _Cost[57] := 1*(hrs_tier3_certify*labor_sys_rate);
585       inc(TotEval);
586
587     //pa_review_tier3  row: 58
588     _Cost[58] := 1*(hrs_tier3_certify*labor_pa_rate);
589       inc(TotEval);
590
591   end;
592 end;
593
594 procedure TLSRCompiledCostBaseline._EvaluateState(const Yr : integer);
595 begin
596   _Reset(True);
597   _LoadVars();
598   with _Variables do begin
599
600   end;
601 end;
602
603
604 procedure TLSRCompiledCostBaseline._SetVarPointer(const name : string; const pd : PDouble);
605 var s : string;
606 begin
607   s := lowercase(name);
608 (*_SetVarPointerBASELINE*)
609 end;
610
611 function TLSRCompiledCostBaseline._DumpVars: string;
```

```
612 var s : string;
613 begin
614   s:='';
615   s := s + 'b_exceed: ' + _Variables.b_exceed.ToString + #13#10;
616   s := s + 'b_nondetect: ' + _Variables.b_nondetect.ToString + #13#10;
617   s := s + 'b_nondetect_all_ep: ' + _Variables.b_nondetect_all_ep.ToString + #13#10;
618   s := s + 'b_treat_nondetect: ' + _Variables.b_treat_nondetect.ToString + #13#10;
619   s := s + 'epa533: ' + _Variables.epa533.ToString + #13#10;
620   s := s + 'epa533_cost: ' + _Variables.epa533_cost.ToString + #13#10;
621   s := s + 'epa537: ' + _Variables.epa537.ToString + #13#10;
622   s := s + 'epa537_cost: ' + _Variables.epa537_cost.ToString + #13#10;
623   s := s + 'hrs_pa_adopt_rule: ' + _Variables.hrs_pa_adopt_rule.ToString + #13#10;
624   s := s + 'hrs_pa_initial_ta: ' + _Variables.hrs_pa_initial_ta.ToString + #13#10;
625   s := s + 'hrs_pa_report_ep: ' + _Variables.hrs_pa_report_ep.ToString + #13#10;
626   s := s + 'hrs_pa_train: ' + _Variables.hrs_pa_train.ToString + #13#10;
627   s := s + 'hrs_pa_treat: ' + _Variables.hrs_pa_treat.ToString + #13#10;
628   s := s + 'hrs_pa_waiver: ' + _Variables.hrs_pa_waiver.ToString + #13#10;
629   s := s + 'hrs_samp: ' + _Variables.hrs_samp.ToString + #13#10;
630   s := s + 'hrs_sdwis: ' + _Variables.hrs_sdwis.ToString + #13#10;
631   s := s + 'hrs_sys_adopt_rule: ' + _Variables.hrs_sys_adopt_rule.ToString + #13#10;
632   s := s + 'hrs_sys_initial_ta: ' + _Variables.hrs_sys_initial_ta.ToString + #13#10;
633   s := s + 'hrs_sys_report_ep: ' + _Variables.hrs_sys_report_ep.ToString + #13#10;
634   s := s + 'hrs_sys_treat: ' + _Variables.hrs_sys_treat.ToString + #13#10;
635   s := s + 'hrs_sys_waiver: ' + _Variables.hrs_sys_waiver.ToString + #13#10;
636   s := s + 'hrs_tier2_approve: ' + _Variables.hrs_tier2_approve.ToString + #13#10;
637   s := s + 'hrs_tier2_certify: ' + _Variables.hrs_tier2_certify.ToString + #13#10;
638   s := s + 'hrs_tier2_distribute: ' + _Variables.hrs_tier2_distribute.ToString + #13#10;
639   s := s + 'hrs_tier2_template: ' + _Variables.hrs_tier2_template.ToString + #13#10;
640   s := s + 'hrs_tier3_approve: ' + _Variables.hrs_tier3_approve.ToString + #13#10;
641   s := s + 'hrs_tier3_certify: ' + _Variables.hrs_tier3_certify.ToString + #13#10;
642   s := s + 'hrs_tier3_distribute: ' + _Variables.hrs_tier3_distribute.ToString + #13#10;
643   s := s + 'hrs_tier3_template: ' + _Variables.hrs_tier3_template.ToString + #13#10;
644   s := s + 'labor_pa_rate: ' + _Variables.labor_pa_rate.ToString + #13#10;
645   s := s + 'labor_rate: ' + _Variables.labor_rate.ToString + #13#10;
646   s := s + 'labor_sys_rate: ' + _Variables.labor_sys_rate.ToString + #13#10;
647   s := s + 'numb_hh: ' + _Variables.numb_hh.ToString + #13#10;
648   s := s + 'numb_initial_samples: ' + _Variables.numb_initial_samples.ToString + #13#10;
649   s := s + 'numb_nondetect_samples: ' + _Variables.numb_nondetect_samples.ToString + #13#10;
650   s := s + 'statepfas: ' + _Variables.statepfas.ToString + #13#10;
651   s := s + 'tier2_cost: ' + _Variables.tier2_cost.ToString + #13#10;
652   s := s + 'tier3_cost: ' + _Variables.tier3_cost.ToString + #13#10;
653   s := s + 'ucmr5: ' + _Variables.ucmr5.ToString + #13#10;
654   s := s + 'wbs_model: ' + _Variables.wbs_model.ToString + #13#10;
655
656   result:=s;
657 end;
658
659 constructor TLSRCompiledCostBaseline.create;
660 begin
661   inherited create;
662
663 end;
664
665 { TLSRCompiledCostOption }
666
667 procedure TLSRCompiledCostOption._LoadVars;
668 begin
669 (*LoadVarsOPTION*)
670 end;
671
672 procedure TLSRCompiledCostOption._Evaluate(const Yr : integer);
673 begin
674   _Reset(False);
675   _LoadVars();
676   with _Variables do begin
677 (*EVALUATEOPTION*)
678   end;
679 end;
680
681 procedure TLSRCompiledCostOption._EvaluateState(const Yr : integer);
682 begin
683   _Reset(True);
684   _LoadVars();
685   with _Variables do begin
686 (*EVALUATESTATEOPTION*)
```

```
687   end;
688 end;
689
690
691 procedure TLSRCompiledCostOption._SetVarPointer(const name : string; const pd : PDouble);
692 var s : string;
693 begin
694   s := lowercase(name);
695 (*_SetVarPointerOPTION*)
696 end;
697
698 function TLSRCompiledCostOption._DumpVars: string;
699 var s : string;
700 begin
701   s:='';
702 (*DumpVarsOPTION*)
703   result:=s;
704 end;
705
706 constructor TLSRCompiledCostOption.create;
707 begin
708   inherited create;
709 (*SetStateOPTION*)
710 end;
711
712
713
714 end.
715
```

PrecompileCosts\LCRCompiledCostTemplate.pas

```
1  unit LCRCompiledCostTemplate;
2
3  //Autogenerated unit -
4  //Generated - (*DATE*)
5  //Baseline Workbook - (*WORKBOOKBASELINE*)
6  //Found variables in workbook - (*NUMVARSBASELINE*)
7  //Found costing steps - 9999
8  //Option Workbook - (*WORKBOOKOPTION*)
9  //Found variables in workbook - (*NUMVARSOPTION*)
10 //Found costing steps - 9998
11
12 interface
13
14 uses Math, SysUtils;
15
16 type
17   TLSRValueStore = array[0..9997] of double;
18   TLSRValueStoreFlag = array[0..9997] of boolean;
19   TLSRCalcYearStore = array[0..100,0..9997] of boolean;
20
21   _VariablesRecBASELINE = record
22 (*VARIABLESBASELINE*)
23   end;
24   _VariablesRecOPTION = record
25 (*VARIABLESOPTION*)
26   end;
27
28   TLSRCompiledCost = class
29     _CalcCost,_Cost,_Hours,_OM,_Labor  : TLSRValueStore;
30     _ImAState : TLSRValueStoreFlag;
31     _YearOK : TLSRCalcYearStore;
32     TotEval : int64;
33     constructor create();
34     procedure _Reset(aState : boolean);
35     procedure _LoadVars; virtual; abstract;
36     procedure _SetVarPointer(const name : string; const pd : PDouble); virtual; abstract;
37     procedure _Evaluate(const Yr : integer); virtual; abstract;
38     procedure _EvaluateState(const Yr: integer); virtual; abstract;
39     function _DumpVars : string; virtual; abstract;
40
41   end;
```

```
42
43    TLSRCompiledCostBaseline = class(TLSRCompiledCost)
44      _Variables : _VariablesRecBaseline;
45
46      constructor create();
47      procedure _LoadVars; override;
48      procedure _SetVarPointer(const name : string; const pd : PDouble); override;
49      procedure _Evaluate(const Yr : integer); override;
50      procedure _EvaluateState(const Yr: integer); override;
51
52      function _DumpVars : string; override;
53    end;
54
55    TLSRCompiledCostOption = class(TLSRCompiledCost)
56      _Variables : _VariablesRecOption;
57
58      constructor create();
59      procedure _LoadVars; override;
60      procedure _SetVarPointer(const name : string; const pd : PDouble); override;
61      procedure _Evaluate(const Yr : integer); override;
62      procedure _EvaluateState(const Yr: integer); override;
63
64      function _DumpVars : string; override;
65    end;
66
67
68  const
69    _CWorkBookBASELINE = '(*WORKBOOKBASELINE*)';
70    _CWorkBookOPTION = '(*WORKBOOKOPTION*)';
71    _CWorkBookDate = '(*DATE*)';
72
73  var
74    _UseCompiled : boolean = false;
75
76  implementation
77
78  { TLSRCompiledCost }
79
80  constructor TLSRCompiledCost.create;
81  begin
82    TotEval:=0;
83  end;
84
85  procedure TLSRCompiledCost._Reset(aState : boolean);
86  var i : integer;
87  begin
88    for i:=0 to high(_CalcCost) do begin
89      if not (_ImAState[i] = aState) then continue;
90      _CalcCost[i]:=0;
91      _Cost[i]:=0;
92      _Hours[i]:=0;
93      _OM[i]:=0;
94      _Labor[i]:=0;
95    end;
96  end;
97
98  { TLSRCompiledCostBaseline }
99
100 procedure TLSRCompiledCostBaseline._LoadVars;
101 begin
102 (*LoadVarsBASELINE*)
103 end;
104
105 procedure TLSRCompiledCostBaseline._Evaluate(const Yr : integer);
106 begin
107   _Reset(False);
108   _LoadVars();
109   with _Variables do begin
110 (*EVALUATEBASELINE*)
111   end;
112 end;
113
114 procedure TLSRCompiledCostBaseline._EvaluateState(const Yr : integer);
115 begin
```

```
116   _Reset(True);
117   _LoadVars();
118   with _Variables do begin
119 (*EVALUATESTATEBASELINE*)
120   end;
121 end;
122
123
124 procedure TLSRCompiledCostBaseline._SetVarPointer(const name : string; const pd : PDouble);
125 var s : string;
126 begin
127   s := lowercase(name);
128 (*_SetVarPointerBASELINE*)
129 end;
130
131 function TLSRCompiledCostBaseline._DumpVars: string;
132 var s : string;
133 begin
134   s:='';
135 (*DumpVarsBASELINE*)
136   result:=s;
137 end;
138
139 constructor TLSRCompiledCostBaseline.create;
140 begin
141   inherited create;
142 (*SetStateBASELINE*)
143 end;
144
145 { TLSRCompiledCostOption }
146
147 procedure TLSRCompiledCostOption._LoadVars;
148 begin
149 (*LoadVarsOPTION*)
150 end;
151
152 procedure TLSRCompiledCostOption._Evaluate(const Yr : integer);
153 begin
154   _Reset(False);
155   _LoadVars();
156   with _Variables do begin
157 (*EVALUATEOPTION*)
158   end;
159 end;
160
161 procedure TLSRCompiledCostOption._EvaluateState(const Yr : integer);
162 begin
163   _Reset(True);
164   _LoadVars();
165   with _Variables do begin
166 (*EVALUATESTATEOPTION*)
167   end;
168 end;
169
170
171 procedure TLSRCompiledCostOption._SetVarPointer(const name : string; const pd : PDouble);
172 var s : string;
173 begin
174   s := lowercase(name);
175 (*_SetVarPointerOPTION*)
176 end;
177
178 function TLSRCompiledCostOption._DumpVars: string;
179 var s : string;
180 begin
181   s:='';
182 (*DumpVarsOPTION*)
183   result:=s;
184 end;
185
186 constructor TLSRCompiledCostOption.create;
187 begin
188   inherited create;
189 (*SetStateOPTION*)
190 end;
```

```
end.
```

PrecompileCosts\LCRPreCompile.pas

```
unit LCRPreCompile;

interface

uses System.SysUtils, Classes, DateUtils, CostsCsv, DataNeedsCSV;

type
  TLCRPreCompile = class
  private
    fSourcePath, fCosts, fData, fOCosts, tmpString : string;
    fOnSection: string;
    fAllVars, fConstVars : TStringList;
    MaxR, CurR : integer;
    procedure ReadDataNeeds(fDataIn: string);
    procedure ReadSteps(fDataIn: string);
    procedure ExtractNames(s : string);
    procedure AddVars;
    procedure AddLookup;
    procedure LoadVars;
    procedure LoadVarsOk;
    procedure DumpVars;
    procedure DecorateConstants(var t: string);
    procedure DecorateVars(var t: string);
    procedure Go2(acFile: string);
  public
    constructor create(aSourcePth : string; aCostsFile, aDataFile,
                       aCostsOption : string);
    destructor Destroy; override;

    procedure Go;
  end;

implementation

const NL = #13#10;

{ TLCRPreCompile }

procedure TLCRPreCompile.AddLookup;
var i : integer;
    tmps : string;
const indent = '  ';
begin
  tmps:=indent;
  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+'if s = '+ QuotedStr(fAllVars[i])+' then _Variables.'+fAllVars[i]+':=pd';
    if i<fAllVars.Count -1 then
      tmps:=tmps+' else '+NL+indent
    else
      tmps:=tmps+' ;'+NL;
  end;
  tmpString := StringReplace(tmpString,'(*_SetVarPointer'+FOnSection+'*)',tmps, [rfIgnoreCase,rfReplaceAll]);
end;

procedure TLCRPreCompile.AddVars;
var i : integer;
    tmps : string;
const indent = '    ';
begin
  tmps:=indent;
  if fAllVars.Count=0 then begin
    tmps := tmps + 'd: pdouble;';
    tmpString := StringReplace(tmpString,'(*VARIABLES'+FOnSection+'*)',tmps, [rfIgnoreCase,rfReplaceAll]);
    exit;
  end;
```

```
  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+fAllVars[i];
    if i<fAllVars.Count -1 then tmps:=tmps+',';
    if i mod 6 = 1 then tmps:=tmps+NL+indent;
  end;
  tmps:=tmps + ' : pdouble;'+NL+NL+indent;

  (*
  for i:=0 to fAllVars.Count -1 do begin
    tmps:=tmps+'P_'+fAllVars[i];
    if i<fAllVars.Count -1 then tmps:=tmps+',';
    if i mod 6 = 1 then tmps:=tmps+NL+indent;
  end;
  tmps:=tmps + ' : pdouble;';
  *)
  tmpString := StringReplace(tmpString,'(*VARIABLES'+FOnSection+'*)',tmps, [rfIgnoreCase,rfReplaceAll]);
end;

procedure TLCRPreCompile.LoadVars;
var i : integer;
    tmps : string;
const indent = '  ';
begin
//  tmps:=indent;
//  for i:=0 to fAllVars.Count -1 do begin
//    tmps:=tmps+'if Assigned(_Variables.P_'+fAllVars[i]+') then _Variables.'+fAllVars[i] + ':= _Variables.P_'+fAllVars[i]+'^;'+NL+indent;
//  end;
//  tmpString := StringReplace(tmpString,'(*LoadVars'+FOnSection+'*)',tmps, [rfIgnoreCase,rfReplaceAll]);' +
end;

procedure TLCRPreCompile.LoadVarsOk;
var i : integer;
    tmps : string;
const indent = '  ';
begin
  tmps:=indent + 'result := ('+NL+indent;
  if fAllVars.Count = 0 then
    tmps:=tmps +'true'
  else
    for i:=0 to fAllVars.Count -1 do begin
      if i  <  fAllVars.Count -1 then
       tmps:=tmps+'Assigned(_Variables.'+fAllVars[i]+') and '+NL+indent
      else
       tmps:=tmps+'Assigned(_Variables.'+fAllVars[i]+')'+NL+indent;
  end;
  tmps:=tmps+');' +NL;
  tmpString := StringReplace(tmpString,'(*LoadVarsOk'+FOnSection+'*)',tmps, [rfIgnoreCase,rfReplaceAll]);
end;

constructor TLCRPreCompile.create(aSourcePth, aCostsFile, aDataFile,aCostsOption : string);
begin
  fSourcePath:=aSourcePth;
  fCosts:=aCostsFile;
  fData:=aDataFile;
  fOCosts:=aCostsOption;
end;

procedure TLCRPreCompile.DecorateConstants(var t: string);
begin
  for var i := 0 to fConstVars.Count-1 do begin
    var sr := FConstVars[i];
    var idx := pos(sr,t);
    if idx>0 then begin
      t := StringReplace(t, sr, sr+'[P.SourceWater,P.SystemSize]',[rfReplaceAll]);
    end;
  end;
end;

procedure TLCRPreCompile.DecorateVars(var t: string);
begin
  for var i := 0 to fAllVars.Count-1 do begin
    var sr := FAllVars[i];
    var idx := pos(sr,t);
```

```
140     var len := length(sr);
141     var ok := true;
142     if idx>0 then begin
143       for var j := 0 to fConstVars.Count-1 do begin
144         var sr2 := FConstVars[j];
145         var idx2 := pos(sr2,t);
146         if idx2=idx then begin
147           ok:=false;
148           break;
149         end;
150       end;
151       if not ok then continue;
152       if (idx+len<length(t)) and (not (t[idx+len] in [' ','*','+','/','-',')'])) then ok := false;
153
154       if ok then begin
155         t := StringReplace(t, sr, sr+'^',[rfReplaceAll]);
156       end;
157     end;
158   end;
159 end;
160
161 destructor TLCRPreCompile.Destroy;
162 begin
163   inherited;
164 end;
165
166 procedure TLCRPreCompile.DumpVars;
167 var i : integer;
168     tmps : string;
169 const indent = '  ';
170 begin
171   tmps:=indent;
172   for i:=0 to fAllVars.Count -1 do begin
173     tmps:=tmps+'s := s + ' +QuotedStr(fAllVars[i] + ': ') +' + _Variables.'+fAllVars[i]+'^.ToString + #13#10;'+NL+indent;
174   end;
175   tmpString := StringReplace(tmpString,'(*DumpVars'+FOnSection+'*)',tmps, [rfIgnoreCase,rfReplaceAll]);
176 end;
177
178 procedure TLCRPreCompile.ExtractNames(s: string);
179 var i,j : integer;
180     v : string;
181 const
182     OKC = ['a'..'z','A'..'Z','0'..'9','_'];
183     OKS = ['a'..'z','A'..'Z','_'];
184 begin
185   i:=1;
186   while i<=length(s) do begin
187     v:='';
188     while (not (s[i] in OKC)) and (I<length(s)) do inc(i);
189     repeat
190       if s[i] in OKC then
191         v:=v+s[i];
192       inc(i);
193       if i > length(s) then break;
194     until (not (s[i] in OKC)) and (I<=length(s));
195     if length(v)>0 then
196       if v[1] in OKS then
197         if fConstVArs.IndexOf(lowercase(V))<0 then
198           fAllVars.Add(lowercase(V));
199   end;
200 end;
201
202 procedure TLCRPreCompile.Go2(acFile: string);
203 begin
204   tmpString := StringReplace(tmpString,'(*WORKBOOK'+FOnSection+'*)',acFile, [rfIgnoreCase,rfReplaceAll]);
205   tmpString := StringReplace(tmpString,'(*DATE*)',DateTimeToStr(Now()), [rfIgnoreCase,rfReplaceAll]);
206
207   fAllVars := TStringList.Create;
208   fAllVars.Sorted := true;
209   fAllVars.Duplicates := dupIgnore;
210
211   ReadSteps(acFile);
212   AddVars();
213   AddLookup();
214   LoadVars();
```

```
  LoadVarsOk();
  DumpVars();
  tmpString := StringReplace(tmpString,'(*NUMVARS'+FOnSection+'*)',fAllVars.Count.ToString, [rfIgnoreCase,rfReplaceAll]);
  fAllVars.Free;

  if FOnSection = 'OPTION' then
    tmpString := StringReplace(tmpString,'9998', CurR.ToString, [rfIgnoreCase,rfReplaceAll])
  else
    tmpString := StringReplace(tmpString,'9999', CurR.ToString, [rfIgnoreCase,rfReplaceAll]);

end;

procedure TLCRPreCompile.Go;
var tSL : TStringList;
    v : integer;
begin
  tSL := TSTringList.Create;
  tSL.LoadFromFile(fSourcePath+'CompiledCostTemplate.pas');
  tSL.Strings[0]:='unit CompiledCost;' ;
  tmpString:=tSL.Text;

  MaxR := 0;

  fConstVars := TStringList.Create;
  fConstVars.Sorted := true;
  fConstVars.Duplicates := dupIgnore;

  ReadDataNeeds(fData);
  FOnSection := 'OPTION';
  Go2(fOCosts);
  FOnSection := 'BASELINE';
  Go2(fCosts);
  fConstVars.Free;

  tmpString := StringReplace(tmpString,'9997', MaxR.ToString, [rfIgnoreCase,rfReplaceAll]);
  tSL.Text := tmpString;
  tSL.SaveToFile(fSourcePath+'CompiledCost.pas');
  tSL.Free;
end;

procedure TLCRPreCompile.ReadDataNeeds(fDataIn: string);
begin
  const T =TDataRecords.Create;
  const indent = '  ';
  T.LoadFromFile(fDataIn);
  var tmps: string := '';
  for var i := 0 to T.Count-1 do begin
    var r := T.DataRecord[i];
    fConstVars.Add(lowercase(r.Name));
    var fs := format('%s: TSSArray = ((%g,%g,%g,%g,%g,%g,%g,%g,%g),(%g,%g,%g,%g,%g,%g,%g,%g,%g));',[lowercase(r.Name),
      r.gwss0, r.gwss101, r.gwss501, r.gwss1001, r.gwss3301, r.gwss10k, r.gwss50k, r.gwss100k, r.gwss1m,
      r.ss0, r.ss101, r.ss501, r.ss1001, r.ss3301, r.ss10k, r.ss50k, r.ss100k,r.ss1m]);
    tmps := tmps + indent + fs + NL;
  end;
  tmpString := StringReplace(tmpString,'(*CONSTANTS*)', tmps, [rfIgnoreCase,rfReplaceAll]);
  T.Free;
end;

procedure TLCRPreCompile.ReadSteps(fDataIn: string);
var
  Costs: TExpressionsRecords;
  NumRows: integer;
  CostRecord: TExpressionsRecord;

  r, ci: integer;
  evString,t,evStringSt,SetState, tmps : string;
  usecc,astate : boolean;
const indent = '  ';
begin
  CurR:=0;
  if not FileExists(fDataIn) then begin
    tmpString := StringReplace(tmpString,'(*SetState'+FOnSection+'*)', '', [rfIgnoreCase,rfReplaceAll]);
    tmpString := StringReplace(tmpString,'(*EVALUATE'+FOnSection+'*)','', [rfIgnoreCase,rfReplaceAll]);
    tmpString := StringReplace(tmpString,'(*EVALUATESTATE'+FOnSection+'*)','', [rfIgnoreCase,rfReplaceAll]);
```

```
    exit;
  end;

  Costs := TExpressionsRecords.Create;
  Costs.LoadFromFile(fDataIn);
  NumRows := Costs.Count;

  {
    IDNum : integer;
    CostName : ansistring;
    ApplicationLevel : ansistring;
    BriefCostDescription : ansistring;
    CostDescription : ansistring;
    CostOptionTrigger : ansistring;
    Occurrence : ansistring;
    AdditionalTriggers : ansistring;
    Year_SmallSys : ansistring;
    Year_NonSmallSys : ansistring;
    Formula : ansistring;
    Notes : ansistring;
  }

  ci:=0;
  evString := indent;
  var evStringSys := indent;
  SetState := indent;
  var nmString := indent;
  var epString := indent;
  var Syscost: boolean;
  for r := 0 to NumRows-1 do begin
     CostRecord := Costs.ExpressionsRecord[r];

     t := trim(CostRecord.CostName);
     if t[1]='#' then continue;

     if pos('ENTRY POINT', uppercase(CostRecord.ApplicationLevel))<1 then begin
       epString := epstring + '_EPCost[' +ci.ToString+ '] := false;'+ NL + indent;
       SysCost := true;
     end else
       SysCost := false;

     if SysCost then
       tmps := evstringsys
     else
       tmps := evstring;
     tmps := tmps + NL + indent;
     tmps := tmps + '//' + t +'  row: ' +CostRecord.RowNum.ToString + NL + indent;
     usecc:=false;

     nmString := nmString + '_Names[' +ci.ToString+ '] := ' + QuotedStr(t) + ';' + NL + indent+
                 '_Cat[' +ci.ToString+ '] := ' + QuotedStr(CostRecord.AgglomeratorCategories) + ';' + NL + indent;

     t:=trim(CostRecord.CostTrigger);
     ExtractNames(t);
     DecorateConstants(t);
     DecorateVars(t);
     tmps := tmps + 'if (_YearOK[sz,Yr,' +ci.ToString+ ']) then begin'+NL + indent+indent;
     if t<>'' then begin
       tmps := tmps + '_CalcCost['+ci.ToString+'] := '+ t + ';'+NL + indent + indent +
              'inc(TotEval);'+NL + indent + indent ;
       usecc:=true;
       tmps := tmps + 'if _CalcCost['+ci.ToString+'] > 0 then begin'+NL + indent+indent+'  ';
     end;


     t:=trim(CostRecord.Formula);
     ExtractNames(t);
     DecorateConstants(t);
     DecorateVars(t);
     if t<>'' then begin
       tmps := tmps + '_Cost['+ci.ToString+'] := '+ t + ';'+NL + indent ;
       if usecc  then
         tmps := tmps +  indent + indent
       else
         tmps := tmps +  indent ;
```

```
364
365         tmps := tmps + 'for DRIdx := 0 to _HighDRIdx do begin' +NL + indent + indent;
366         tmps := tmps + 'v := _Cost['+ci.ToString+'] / Data^.PreCalcDiscRate[DRIdx,Yr+PeriodAdjust];'+NL + indent + indent;
367         tmps :=  tmps + '_i(_PCost[DRIdx,'+ci.ToString+']^,v);'+NL + indent + indent;
368         tmps :=  tmps + '_i(_PCostY[Yr,DRIdx,'+ci.ToString+']^,v);'+NL + indent + indent;
369
370         tmps := tmps + 'v := _Cost['+ci.ToString+'] / Data^.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,DRIdx,Yr+PeriodAdjust];'+NL + indent +
 indent;
371         tmps :=  tmps + '_i(_PCost_pws[DRIdx,'+ci.ToString+']^,v);'+NL + indent + indent;
372         tmps :=  tmps + '_i(_PCostY_pws[Yr,DRIdx,'+ci.ToString+']^,v);'+NL + indent + indent;
373         tmps := tmps + 'end;'+NL;
374
375
376         (*
377         tmps := tmps + 'v := CDiscount(Yr,_Cost['+ci.ToString+']);'+NL + indent + indent;
378         tmps := tmps + 'if _PCost['+ci.ToString+'] <> nil then _i(_PCost['+ci.ToString+']^,v);'+NL + indent + indent;
379         tmps := tmps + 'v := _Cost['+ci.ToString+'] / Data^.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,Yr];'+NL + indent + indent;
380         tmps := tmps + 'if _PCost_pws['+ci.ToString+'] <> nil then _i(_PCost_pws['+ci.ToString+']^,v);'+NL + indent + indent;
381         *)
382
383         tmps := tmps +'inc(TotEval);'+NL + indent;
384       end else
385         tmps := tmps + '_Cost['+ci.ToString+'] := 0;'+NL + indent + indent + indent +
386                'inc(TotEval);'+NL + indent
387         ;
388
389
390       if usecc then tmps := tmps + '  ';
391       if usecc then tmps := tmps + NL + indent + 'end;' + NL + indent;
392
393       tmps := tmps + 'end;';
394
395       inc(ci);
396
397       if SysCost then
398         evstringsys := tmps
399       else
400         evstring:=tmps;
401   end;
402
403   //dump out the year arrays...
404   tmps := indent;
405   for var c := 0 to Costs.Count-1 do begin
406     for var y := 0 to 100 do begin
407       for var sz := 0 to 1 do begin
408         if Costs.fCCCalcYearStore[sz,y,c] then begin
409           tmps := tmps +
410                Format('_YearOK[%d,%d,%d] := true;',[sz,y,c]) +
411                NL + indent;
412
413         end;
414       end;
415     end;
416   end;
417   tmpString := StringReplace(tmpString,'(*SetYears'+FOnSection+'*)', tmps, [rfIgnoreCase,rfReplaceAll]);
418
419   CurR := ci;
420   ci:=ci-1;
421   Costs.Free;
422   if ci>MaxR then MaxR:=ci;
423 //  tmpString := StringReplace(tmpString,'9999', ci.ToString, [rfIgnoreCase,rfReplaceAll]);
424   tmpString := StringReplace(tmpString,'(*SetNames'+FOnSection+'*)', nmString, [rfIgnoreCase,rfReplaceAll]);
425   tmpString := StringReplace(tmpString,'(*SetEPCosts'+FOnSection+'*)', epString, [rfIgnoreCase,rfReplaceAll]);
426   tmpString := StringReplace(tmpString,'(*SetState'+FOnSection+'*)', SetState, [rfIgnoreCase,rfReplaceAll]);
427   tmpString := StringReplace(tmpString,'(*EVALUATE'+FOnSection+'*)',evString, [rfIgnoreCase,rfReplaceAll]);
428   tmpString := StringReplace(tmpString,'(*EVALUATESYSTEM'+FOnSection+'*)',evStringSys, [rfIgnoreCase,rfReplaceAll]);
429 end;
430
431
432 end.
433
```

PrecompileCosts\PrecompileCosts.dpr

```
program PrecompileCosts;

uses
  Vcl.Forms,
  frmMain in 'frmMain.pas' {Form1},
  LCRPreCompile in 'LCRPreCompile.pas',
  CostsCsv in 'CostsCsv.pas',
  SafeWaterUtility in 'J:\safewater2022\code\SafeWaterUtility.pas',
  SafeWaterGlobals in 'J:\safewater2022\code\SafeWaterGlobals.pas',
  DataNeedsCsv in 'DataNeedsCsv.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.

```

PrecompileCosts\frmMain.dfm

```
object Form1: TForm1
  Left = 0
  Top = 0
  Caption = 'Form1'
  ClientHeight = 433
  ClientWidth = 745
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -12
  Font.Name = 'Segoe UI'
  Font.Style = []
  OnCreate = FormCreate
  TextHeight = 15
  object Label1: TLabel
    Left = 32
    Top = 40
    Width = 88
    Height = 15
    Caption = 'Option Cost File:'
  end
  object edtFilename: TEdit
    Left = 126
    Top = 37
    Width = 444
    Height = 23
    TabOrder = 0
    Text =
      'K:\Safewater2022\data\CostWorkbook\PFAS_AdminCosts_v17_Expressio' +
      'ns.csv'
  end
  object btnReadCosts: TButton
    Left = 40
    Top = 88
    Width = 75
    Height = 25
    Caption = 'Go'
    TabOrder = 1
    OnClick = btnReadCostsClick
  end
  object Memo1: TMemo
    Left = 40
    Top = 224
    Width = 545
    Height = 121
    Lines.Strings = (
      'Memo1')
    ScrollBars = ssBoth
    TabOrder = 2
```

```
    WordWrap = False
  end
  object btnOpenFile: TButton
    Left = 576
    Top = 36
    Width = 33
    Height = 25
    Caption = '...'
    TabOrder = 3
    OnClick = btnOpenFileClick
  end
  object btnDataNeeds: TButton
    Left = 136
    Top = 88
    Width = 75
    Height = 25
    Caption = 'Data Needs'
    TabOrder = 4
    OnClick = btnDataNeedsClick
  end
  object edSrcPath: TLabeledEdit
    Left = 126
    Top = 136
    Width = 355
    Height = 23
    EditLabel.Width = 82
    EditLabel.Height = 23
    EditLabel.Caption = 'Source file path'
    LabelPosition = lpLeft
    TabOrder = 5
    Text = 'J:\safewater2022\code\costmodules\'
  end
  object edDataNeeds: TLabeledEdit
    Left = 126
    Top = 165
    Width = 459
    Height = 23
    EditLabel.Width = 64
    EditLabel.Height = 23
    EditLabel.Caption = 'Data neeeds'
    LabelPosition = lpLeft
    TabOrder = 6
    Text =
      'K:\Safewater2022\data\CostWorkbook\PFAS_AdminCosts_v17_DataNeeds' +
      '.csv'
  end
  object OpenDialog1: TOpenDialog
    Left = 696
    Top = 56
  end
end

```

PrecompileCosts\frmMain.pas

```
unit frmMain;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Generics.Collections,
  SafeWaterGlobals, SafeWaterUtility, LCRPreCompile, DataNeedsCsv, Vcl.Mask,
  Vcl.ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    edtFilename: TEdit;
    OpenDialog1: TOpenDialog;
    btnReadCosts: TButton;
    Memo1: TMemo;
    btnOpenFile: TButton;
```

```
19      btnDataNeeds: TButton;
20      edSrcPath: TLabeledEdit;
21      edDataNeeds: TLabeledEdit;
22      procedure btnReadCostsClick(Sender: TObject);
23      procedure btnOpenFileClick(Sender: TObject);
24      procedure FormCreate(Sender: TObject);
25      procedure btnDataNeedsClick(Sender: TObject);
26    private
27      { Private declarations }
28    public
29      { Public declarations }
30    end;
31
32  var
33    Form1: TForm1;
34
35  implementation
36
37  {$R *.dfm}
38
39  procedure TForm1.btnDataNeedsClick(Sender: TObject);
40  var
41    DataRecords: TDataRecords;
42    FDataRecords : TList<TDataRecord>;
43    i: integer;
44    cnt: integer;
45  begin
46    if edtFilename.text <> '' then
47    begin
48      DataRecords := TDataRecords.Create;
49      DataRecords.LoadFromFile(edtFilename.Text);
50      FDataRecords := DataRecords.fDataRecords;
51      cnt := DataRecords.Count;
52
53      for i := 0 to cnt -1  do
54      begin
55         Memo1.Lines.Add(FDataRecords[i].Name + ' ' + FDataRecords[i].ss0.ToString);
56      end;
57
58      FDataRecords.Free;
59    end;
60  end;
61
62  procedure TForm1.btnOpenFileClick(Sender: TObject);
63  begin
64    OpenDialog1.InitialDir := DataPath;
65    OpenDialog1.Filter:='Comma-delimited (*.csv) |*.csv';
66    OpenDialog1.Title := 'Select Option Costs';
67
68    if OpenDialog1.Execute() then
69      edtFilename.Text := OpenDialog1.FileName;
70  end;
71
72  procedure TForm1.btnReadCostsClick(Sender: TObject);
73    var T : TLCRPreCompile;
74    var srcPath: string;
75    var sFIle : string;
76  begin
77    if edtFilename.text <> '' then
78    begin
79      srcPath := edSrcPAth.Text;
80      //sFile := OpenDialog1.FileName;
81      sFile := edtFilename.Text;
82      T:=TLCRPreCompile.create(srcPath,'', edDataNeeds.Text, sFile);
83      T.Go;
84      T.Free;
85      showmessage('done');
86    end;
87  end;
88
89  procedure TForm1.FormCreate(Sender: TObject);
90  begin
91    SetPaths;
92  end;
93
```

```
end.

```

SafeWaterRun.dpr

```
program SafeWaterRun;

uses
  FastMM5,
  Vcl.Forms,
  wndMain in 'wndMain.pas' {wMain},
  Vcl.Themes,
  Vcl.Styles,
  SafeWaterGlobals in 'code\SafeWaterGlobals.pas',
  SafeWaterOccurrence in 'code\SafeWaterOccurrence.pas',
  wndTests in 'uicode\wndTests.pas' {wTests},
  SafeWaterUtility in 'code\SafeWaterUtility.pas',
  SafeWaterModelData in 'code\SafeWaterModelData.pas',
  SafeWaterPWSRecords in 'code\SafeWaterPWSRecords.pas',
  SafeWaterModel in 'code\SafeWaterModel.pas',
  SafeWaterConfig in 'code\SafeWaterConfig.pas',
  SafeWaterUncertBucket in 'code\SafeWaterUncertBucket.pas',
  SafeWaterCalcDist in 'code\SafeWaterCalcDist.pas',
  SafeWaterCostBase in 'code\CostModules\SafeWaterCostBase.pas',
  cmPFASOption1 in 'code\CostModules\cmPFASOption1.pas',
  SafeWaterCostModules in 'code\CostModules\SafeWaterCostModules.pas',
  SafeWaterRunner in 'code\SafeWaterRunner.pas',
  SafeWaterMetrics in 'code\SafeWaterMetrics.pas',
  SafeWaterBenefitBase in 'code\BenefitModules\SafeWaterBenefitBase.pas',
  CalcDistSpecFun in 'code\CalcDistSpecFun.pas',
  SafeWaterBenefitModules in 'code\BenefitModules\SafeWaterBenefitModules.pas',
  bmPFASCVD in 'code\BenefitModules\bmPFASCVD.pas',
  PFASPBPK in 'code\PBPK\PFASPBPK.pas',
  Spliner in 'code\PBPK\Spliner.pas',
  bmPFASBase in 'code\BenefitModules\bmPFASBase.pas',
  PFASSerumCalc in 'code\BenefitModules\PFASSerumCalc.pas',
  bmPFASStats in 'code\BenefitModules\bmPFASStats.pas',
  SafeWaterCohortArray in 'code\BenefitModules\SafeWaterCohortArray.pas',
  cmPFASBase in 'code\CostModules\cmPFASBase.pas',
  SafeWaterTFObject in 'code\CostModules\SafeWaterTFObject.pas',
  SafeWaterWBSCostCurves in 'code\CostModules\SafeWaterWBSCostCurves.pas',
  CompiledCost in 'code\CostModules\CompiledCost.pas',
  bmPFASKidneyC in 'code\BenefitModules\bmPFASKidneyC.pas',
  cmPFASOption3 in 'code\CostModules\cmPFASOption3.pas',
  bmPFASLBW in 'code\BenefitModules\bmPFASLBW.pas',
  cmPFASOption4 in 'code\CostModules\cmPFASOption4.pas',
  bmPFASDBP in 'code\BenefitModules\bmPFASDBP.pas',
  cmPFASOption2s in 'code\CostModules\cmPFASOption2s.pas',
  bmPFASLiverC in 'code\BenefitModules\bmPFASLiverC.pas',
  bmPFASAbstractCancSC in 'code\BenefitModules\bmPFASAbstractCancSC.pas',
  cmPFASOption2x in 'code\CostModules\cmPFASOption2x.pas',
  cmPFASOptionMCL1 in 'code\CostModules\cmPFASOptionMCL1.pas',
  cmPFASOccRun in 'code\CostModules\cmPFASOccRun.pas';

{$R *.res}

begin
  ReportMemoryleaksOnShutdown := true;
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TwMain, wMain);
  Application.CreateForm(TwTests, wTests);
  Application.Run;
end.

```

code\BenefitModules\PFASSerumCalc.pas

```
unit PFASSerumCalc;

interface
```

```
4
5  uses Math, Classes, PFASPBPK, SafeWaterGlobals, SafewaterConfig, Generics.Collections,
6       System.Threading, SysUtils, System.Diagnostics, SafewaterUtility;
7
8  type
9    TPFASCode = (pPFOS,pPFOA);
10   TYearlySubPopArray = array of TSubPopArray;
11
12   TSerumCalc = class(TObject)
13   private
14     PBK: TPFASPBPKBase;
15     TimeStep: double;
16     MaxAge: integer;
17
18     StartYear, EndYear, ChangeYear: integer;
19     DoParallel: boolean;
20     fReportProgress: TProgressProc;
21     FInfo: TStringList;
22     BW, Cons: PSubpopArray;
23     DYears: integer;
24     FOutConv: double;
25
26     procedure RunDecayChange(const a: integer);
27   public
28     //serum levels for unit doses...
29     UnitDose: TSubPopArray;
30     //serum decay helper after a change
31     Decays: TYearlySubPopArray;
32
33     constructor create(aParamSelect : TPFASCode; Cfg : TSafewaterConfig; aBW, aCons: PSubpopArray);
34     destructor Destroy; override;
35
36     procedure RunUnitDoses;
37     procedure SetCustomBodyWeights(P : PSubPopArray);
38     procedure ScaleUnitArray(var Dest: TSubPopArray; const aConc: double);
39     procedure GetBeforeAfterSerumDeprecate(var Base, Alt : TSubPopArray; const OnYr: integer; const BConc,AConc : double; const dw_rsc: double;
aOutConv: double=0);
40     procedure GetBeforeAfterSerum(var Base, Alt : TSubPopArray; const OnYr: integer; const BConc,AConc : double; const dw_rsc: double; aOutConv:
double=0);
41     procedure GetBeforeAfterSerumFromDose(var Base, Alt : TSubPopArray; const OnYr: integer; const BDose,ADose : double);
42     procedure GetBeforeAfterSerumFromDoseInd(var Base, Alt : TSubPopArray; const ChgYr: integer; const BDose,ADose : double);
43     procedure ConvertToLifetimeAvg(var Base, Alt : TYearlySubPopArray; const NYears: integer);
44
45     property OutConv: double read FOutConv write FOutConv;
46     property ReportProgress: TProgressProc read fReportProgress write fReportProgress;
47     property Info: TStringList read FInfo;
48     //convenience for debugging
49     property Parallel: boolean read DoParallel write DoParallel;
50   end;
51
52   TPFASSerum = class(TObjectList<TSerumCalc>)
53   private
54     fReportProgress: TProgressProc;
55     FInfo: TStringList;
56   published
57     constructor create(aContamList : TStringList; Cfg : TSafewaterConfig; aBW, aCons: PSubpopArray);
58     destructor Destroy; override;
59     procedure RunUnitDoses;
60     function GetSerumCalc(aName: string) : TSerumCalc;
61     function GetAllInfo: TstringLIst;
62     property ReportProgress: TProgressProc read fReportProgress write fReportProgress;
63     property Info: TStringList read FInfo;
64   end;
65
66
67
68 implementation
69
70 { TSerumCalc }
71
72 procedure TSerumCalc.ConvertToLifetimeAvg(var Base, Alt: TYearlySubPopArray;
73   const NYears: integer);
74 begin
75   var s,r,a,cy,l,ly : integer;
76   var SSA, SSB: double;
```

```
  var _A,_B: TYearlySubpopArray;
  //copy alt array - could get stepped on otherwise
  setlength(_A,NYears);
  setlength(_B,NYears);
  for r := 0 to NYears-1 do begin
    _A[r] := CopySubpopData(Alt[r]);
    _B[r] := CopySubpopData(Base[r]);
  end;

  for cy := 0 to NYears-1 do begin
    for s := gMale to gFemale do begin
      for r := reWhite to reOther do begin
        SSB:=0;
        for a := 0 to CMaxAge do begin
          SSB := SSB + Base[cy][s,r,a];
          Base[cy][s,r,a] := SSB / (a+1);
          l := a - cy;
          SSA:=0;
          for ly := 0 to a do begin
             if l<0 then
               SSA:=SSA + _A[cy][s,r,ly]
             else if ly < l then
               SSA:=SSA + _B[cy][s,r,ly]
             else
               SSA:=SSA + _A[ly-l][s,r,ly];
          end;
          Alt[cy][s,r,a] := SSA / (a+1);
        end;
      end;
    end;
  end;
end;

constructor TSerumCalc.create(aParamSelect: TPFASCode; Cfg : TSafewaterConfig; aBW, aCons: PSubpopArray);
begin
  case aPAramSelect of
    pPFOS: PBK := TPFOSPBPK.create;
    pPFOA: PBK := TPFOAPBPK.create;
  end;
  Cons := aCons;
  BW := aBW;
//  PBK.SetCustomBodyWeights(BW);
  MaxAge := Cfg.MaxAge;
  TimeStep := Cfg.PBPKTimeStep;
  StartYear := Cfg.StartYear;
  EndYear := Cfg.EndYear;
  ChangeYear := Cfg.PFASChangeYear;
  DoParallel := True;

  FInfo := TStringList.Create;
  DYears := 50;
  FOutConv := cmgPerLtongPerml
end;

destructor TSerumCalc.Destroy;
begin
  PBK.Free;
  FInfo.Free;
  inherited;
end;

procedure TSerumCalc.GetBeforeAfterSerumDeprecate(var Base, Alt : TSubPopArray; const OnYr: integer; const BConc,AConc : double; const dw_rsc: double;
aOutConv: double=0);
begin
  var D, SB, SA, P, B: double;
  var s,r,a,y : integer;
  var ac := aConc * PPTTomgl;
  var bc := bConc * PPTTomgl;
  var c := aOutConv;
  if c=0 then c := FOutConv;

  for s := gMale to gFemale do begin
    for r := reWhite to reOther do begin
      for a := 0 to CMaxAge do begin
        y := min(CMaxAge, a+OnYr);
```

```
151         //TODO check this out with predictable decay test
152         D := Cons[s,r,y] / BW[s,r,y];
153 D:=0.013;
154         b := bc * D * (1-dw_rsc) / dw_rsc ;
155         SB := UnitDose[s,r,y] * (bc * D + b) * c;
156         Base[s,r,a] := SB;
157
158         SA := UnitDose[s,r,y] * (ac * D + b) * c;
159         if OnYr < DYears  then begin
160           P := Decays[OnYr,s,r,y];
161           SA := -((1 - P) * (SB - SA)) + SB;
162         end;
163         Alt[s,r,a] := SA;
164       end;
165     end;
166   end;
167 end;
168
169 procedure TSerumCalc.GetBeforeAfterSerum(var Base, Alt : TSubPopArray; const OnYr: integer; const BConc,AConc : double; const dw_rsc: double;
    aOutConv: double=0);
170 begin
171   var D, SB, SA, P, B: double;
172   var s,r,a : integer;
173   var ac := aConc * PPTTomgl;
174   var bc := bConc * PPTTomgl;
175   var c := aOutConv;
176   if c=0 then c := FOutConv;
177
178   for s := gMale to gFemale do begin
179     for r := reWhite to reOther do begin
180       for a := 0 to CMaxAge do begin
181         //TODO check this out with predictable decay test
182         D := Cons[s,r,a] / BW[s,r,a];
183 D:=0.013;
184         b := bc * D * (1-dw_rsc) / dw_rsc ;
185         SB := UnitDose[s,r,a] * (bc * D + b) * c;
186         Base[s,r,a] := SB;
187
188         SA := UnitDose[s,r,a] * (ac * D + b) * c;
189         if OnYr < DYears  then begin
190           P := Decays[OnYr,s,r,a];
191           SA := -((1 - P) * (SB - SA)) + SB;
192         end;
193         Alt[s,r,a] := SA;
194       end;
195     end;
196   end;
197 end;
198
199
200 procedure TSerumCalc.GetBeforeAfterSerumFromDose(var Base, Alt : TSubPopArray; const OnYr: integer; const BDose,ADose : double);
201 begin
202   var SB, SA, P: double;
203   var s,r,a,y : integer;
204   for s := gMale to gFemale do begin
205     for r := reWhite to reOther do begin
206       for a := 0 to CMaxAge do begin
207         y := min(CMaxAge, a+OnYr);
208         SB := BDose * UnitDose[s,r,y] * FOutConv;
209         Base[s,r,a] := SB;
210         SA := ADose * UnitDose[s,r,y] * FOutConv;
211         if OnYr >= DYears  then
212           Alt[s,r,a] := SA
213         else begin
214           P := Decays[OnYr,s,r,a];
215           Alt[s,r,a] := -((1 - P) * (SB - SA)) + SB;
216         end;
217       end;
218     end;
219   end;
220 end;
221
222 procedure TSerumCalc.GetBeforeAfterSerumFromDoseInd(var Base, Alt: TSubPopArray;
223   const ChgYr: integer; const BDose, ADose: double);
224 begin
```

```
225   var SB, SA, P: double;
226   var s,r,a : integer;
227   for s := gMale to gFemale do begin
228     for r := reWhite to reOther do begin
229       for a := 0 to CMaxAge do begin
230         SB := BDose * UnitDose[s,r,a] * FOutConv;
231         Base[s,r,a] := SB;
232         SA := ADose * UnitDose[s,r,a] * FOutConv;
233
234         if (ChgYr>a)  then
235           Alt[s,r,a] := SB
236         else
237         if (a - ChgYr >= DYears)  then
238           Alt[s,r,a] := SA
239         else begin
240           P := Decays[a-ChgYr,s,r,a];
241           Alt[s,r,a] := -((1 - P) * (SB - SA)) + SB;
242         end;
243       end;
244     end;
245   end;
246 end;
247
248 procedure TSerumCalc.RunDecayChange(const a: integer);
249 begin
250   var S:= TPFASPBKRun.create;
251   S.Defaults;
252   S.DoseChange.Clear;
253   S.TimeStep := TimeStep;
254   var T: TValueChange;
255   T.Year := 1960;
256   T.Value := 1;
257   S.DoseChange.Add(T);
258
259   T.Year := 2025;
260   T.Value := 0.5;
261   S.DoseChange.Add(T);
262
263   var bs,ns, cs: double;
264   for var sx := gMale to gFemale do begin
265     for var r := reWhite to reOther do begin
266       S.Sex := sx;
267       S.Race := r;
268       for var y := 0 to DYears-1 do begin
269         T.Year := 2025 - y;
270         S.BirthYear := 2025 - a;
271         S.EndYear := 2025 + 1;
272         S.DoseChange.Items[1] := T;
273         if y>a then begin
274          Decays[y,sx,r,a]:=0;
275          continue;
276         end;
277         var Res := PBK.Run(S);
278         for var RR in Res do begin
279           if RR.Age<a then continue;
280           bs := UnitDose[sx,r,a] ;
281           cs := bs * 0.5;
282           ns := RR.Data.C_m;
283           Decays[y,sx,r,a] := 1 - ((bs - ns) /  (bs - cs));
284           if ABS(Decays[y,sx,r,a]) < 5e-5 then Decays[y,sx,r,a] := 0;
285           //only need 1 age
286           break;
287         end;
288         Res.Free;
289       end;
290     end;
291   end;
292   S.Free;
293 end;
294
295
296 procedure TSerumCalc.RunUnitDoses;
297 begin
298   var S:= TPFASPBKRun.create;
299   S.Defaults;
```

```
300   S.TimeStep := TimeStep;
301   var T: TValueChange;
302   T.Year := 1960;
303   T.Value := 1;
304   S.DoseChange.Add(T);
305
306   for var sx := gMale to gFemale do begin
307     for var r := reWhite to reOther do begin
308       S.Sex := sx;
309       S.Race := r;
310       S.BirthYear := 1960;
311       S.EndYear := 1960 + 86;
312       PBK.RunSafeWaterStyle(UnitDose[sx,r],s);
313     end;
314   end;
315
316   Setlength(Decays, DYears); //go out 50 years - that should be ebough to reach equality with alt dose
317   if DoParallel then begin
318     TParallel.For(0,  cMaxAge, procedure(a:Integer)
319       begin
320         RunDecayChange(a);
321         if Assigned(FReportProgress) then FReportProgress(0,0,PBK.Name+' task:'+inttostr(a));
322       end);
323       FInfo.Add(PBK.Name+': total size:' + inttostr(sizeof(UnitDose) * (25 + 1)));
324   end else begin
325     for var a := 0 to cMaxAge do begin
326       RunDecayChange(a);
327       if Assigned(FReportProgress) then FReportProgress(0,0,PBK.Name+' task:'+inttostr(a));
328     end;
329     FInfo.Add(PBK.Name+': total size:' + inttostr(sizeof(UnitDose) * (25 + 1)));
330   end;
331   S.Free;
332
333 end;
334
335 procedure TSerumCalc.ScaleUnitArray(var Dest: TSubPopArray; const aConc: double);
336 begin
337   var aDose: double;
338   for var s := gMale to gFemale do begin
339     for var r := reWhite to reOther do begin
340       for var a := 0 to MaxAge - 1 do begin
341         aDose := aConc * PPTTomgl * Cons[s,r,a] / BW[s,r,a];
342         Dest[s,r,a]  := UnitDose[s,r,a] * aDose * FOutConv;
343       end;
344     end;
345   end;
346
347 end;
348
349
350 procedure TSerumCalc.SetCustomBodyWeights(P: PSubPopArray);
351 begin
352   PBK.SetCustomBodyWeights(P);
353 end;
354
355 { TPFASSerum }
356
357 constructor TPFASSerum.create(aContamList: TStringList; Cfg : TSafewaterConfig; aBW, aCons: PSubpopArray);
358 begin
359   inherited create(true);
360   for var s in aContamList do begin
361     if s='PFOA' then Add(TSerumCalc.create(pPFOA, Cfg, aBW, aCons)) else
362     if s='PFOS' then Add(TSerumCalc.create(pPFOS, Cfg, aBW, aCons));
363   end;
364   FInfo:=TStringList.Create;
365 end;
366
367 destructor TPFASSerum.destroy;
368 begin
369   FInfo.Free;
370   inherited;
371 end;
372
373 function TPFASSerum.GetAllInfo: TstringLIst;
```

```
begin
  FInfo.Clear;
  for var T in self do begin
    FInfo.AddStrings(T.Info);
  end;
  Result:=FInfo;
end;

function TPFASSerum.GetSerumCalc(aName: string): TSerumCalc;
begin
  //doesn't have to be quick - called very infrequently
  Result := nil;
  for var T in self do begin
    if T.PBK.Name = aName then begin
      Result := T;
      exit;
    end;
  end;
end;

procedure TPFASSerum.RunUnitDoses;
begin
  for var T in self do begin
    if Assigned(FReportProgress) then T.ReportProgress := FReportProgress;
    T.RunUnitDoses;
  end;
end;

end.

```

code\BenefitModules\SafeWaterBenefitBase.pas

```
unit SafeWaterBenefitBase;

interface

uses Classes, SafeWaterModelData, StrUtils, SafewaterPWSRecords, SafeWaterOccurrence,
     SafeWaterUncertBucket, SafewaterMetrics, SysUtils, SafewaterUtility, SafewaterCalcDist,
     Math, SafewaterGlobals, SafewaterCohortArray, System.Diagnostics;


type
  TBenefitOutputs = record
    Dollars :double;  //maybe this is the only common output
                      //descendants must add case types
  end;
  TMDRBenefitOutputs = array of TBenefitOutputs;
  TMDRYearlyBenefitOutputs = array of array of TBenefitOutputs;

  TSafeWaterBenefitBase=class(TObject)
  protected
    FDebugList: TstringLIst;
    FName, FGroup: string;
    FActive, FInitialized: boolean;
    FNeededContams: string;
    FData: PSWModelData;
    FGlb: TGlobalUncertainty;

    O: TMDRBenefitOutputs;
    OYearly: TMDRYearlyBenefitOutputs;
    CurConc, AfterConc : TContamArray;
    CurPop : double;
    NumContams: integer;
    FIteration : integer;  //in case modules need to know iteration
    FUseCF: boolean;
    FYears: integer;
    {$ifdef DEBUG}
    _d: boolean; //debug output indicator
    {$endif}

    CohortPop, CohortPopBase: PSubpopArray;  //"eligible" for option total for baseline
    CohortPopTotal: PSubpopArray;  // option total
```

```
41
42      ICohortPop, ICohortPopBase : TSubpopArray;  //used for copying pws pops for Cohort Following
43      ICohortPopTotal : TSubpopArray;
44      ICohortChange, ICohortChangeBase: TSubpopArray;  //used to change pops from cohorts for "eligibility"
45      //ICohortDeaths: TSubpopArray;  //total deaths in year under option (general + endpoint specific)
46
47      Q_Gen: TCohortData;
48      G85MR: double;  //mortality rate for age group 85+;
49      PostAnnualize: boolean;
50      TotalTime: int64;
51      aTimer: TStopWatch;
52
53      procedure StartTimer;
54      procedure TickTimer;
55      procedure SetupContamIndexes; virtual;
56      procedure ParseConc; virtual;
57      procedure SetupModule; virtual;
58      procedure BreakdownModule; virtual;
59      procedure AdvanceCohort;
60      //procedure AdvanceCohortBaseline;
61
62    public
63      constructor Create(aData : PSWModelData; GlbUnc: TGlobalUncertainty;  aNeededContams: string; aName, aGroup: string; aIteration: integer);
virtual;
64      destructor Destroy; override;
65
66      function Init: boolean;
67      procedure SetupUncertainty(Unc: TUncertaintyStudy); virtual;
68      procedure SetupOutputs(Metrics: TCategoryMetrics); virtual;
69
70      procedure ResetBenefits(); virtual;
71
72      procedure SetPWSWideEPValues(const P: TPWSRecord);  virtual;
73      procedure CollectPWSWideEPValues(const P: TPWSRecord); virtual;
74      procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit); virtual;
75      procedure AddEntryPointBenefitsPostProcess(const P: TPWSRecord; const Conc, FinalConc : TContamArray); virtual;
76
77      property Name: string read FName write FName;
78      property Group: string read FGroup write FGroup;
79      property Active: boolean read FActive write FActive;
80      property NeededContams: string read FNeededContams write FNeededContams;
81      property Initialized: boolean read FInitialized;
82      property Output: TMDRBenefitOutputs read O;
83    end;
84
85  implementation
86
87  { TSafeWaterBenefitBase }
88
89  procedure TSafeWaterBenefitBase.AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit);
90  begin
91    CurPop := P.Population / P.EPCount;
92    for var i := 0 to length(CurConc)-1 do begin
93      CurConc[i] := Conc[i];
94      AfterConc[i] := FInalConc[i];
95    end;
96    ParseConc();
97
98    if FUseCF then begin
99      ICohortPop := CopySubpopData(P.CohortPop);
100     ICohortPopBase := CopySubpopData(P.CohortPop);
101     ICohortPopTotal := CopySubpopData(P.CohortPop);
102     CohortPop := @ICohortPop;
103     CohortPopTotal := @ICohortPopTotal;
104     CohortPopBase := @ICohortPopBase;
105   end else begin
106     CohortPop := @P.CohortPop;
107     CohortPopTotal := @P.CohortPop;
108     CohortPopBase := @P.CohortPop;
109   end;
110   DefaultSubpopData(ICohortChange);
111   DefaultSubpopData(ICohortChangeBase);
112 end;
113
114 procedure TSafeWaterBenefitBase.AddEntryPointBenefitsPostProcess(const P: TPWSRecord; const Conc, FinalConc : TContamArray);
```

```
115 begin
116 end;
117
118 procedure TSafeWaterBenefitBase.AdvanceCohort;
119 begin
120   if not FUseCF then exit;
121   var tn,n,c:double;
122   var a,s,r: integer;
123   for s := gMale to gFemale do begin
124     for r := reWhite to reOther do begin
125       tn:=0;
126       for a := CMaxAge downto 0 do begin
127         if a = cMaxAge then begin
128           n := CohortPopTotal[s,r,a] * G85MR;
129           c := CohortPop[s,r,a] * G85MR;
130           //b := CohortPopBase[s,r,a] * G85MR;
131         end else begin
132           n := CohortPopTotal[s,r,a] * Q_Gen.V[s,r,a];
133           c := CohortPop[s,r,a] * Q_Gen.V[s,r,a];
134           //b := CohortPopBase[s,r,a] * Q_Gen.V[s,r,a];
135         end;
136         tn := tn+n;
137         if a=CMaxAge then begin
138           CohortPopTotal[s,r,a] := max(CohortPopTotal[s,r,a] - n, 0);
139           CohortPop[s,r,a] := max(CohortPop[s,r,a] - c - ICohortChange[s,r,a], 0);
140           //CohortPopBase[s,r,a] := max(CohortPopBase[s,r,a] - b - ICohortChangeBase[s,r,a], 0);
141         end else
142         if a=CMaxAge-1 then begin
143           CohortPopTotal[s,r,a+1] := max(CohortPopTotal[s,r,a+1] + CohortPop[s,r,a] - n, 0);
144           CohortPop[s,r,a+1] := max(CohortPop[s,r,a+1] + CohortPop[s,r,a] - c - ICohortChange[s,r,a], 0);
145           //CohortPopBase[s,r,a+1] := max(CohortPopBase[s,r,a+1] + CohortPopBase[s,r,a] - b - ICohortChangeBase[s,r,a], 0);
146         end else begin
147           CohortPopTotal[s,r,a+1] := max(CohortPopTotal[s,r,a] - n, 0);
148           CohortPop[s,r,a+1] := max(CohortPop[s,r,a] - c - ICohortChange[s,r,a], 0);
149           //CohortPopBase[s,r,a+1] := max(CohortPopBase[s,r,a] - b - ICohortChangeBase[s,r,a], 0);
150         end;
151       end;
152       CohortPop[s,r,0] := tn;
153       //CohortPopBase[s,r,0] := tn;
154       CohortPopTotal[s,r,0] := tn;
155     end;
156   end;
157   DefaultSubpopData(ICohortChange);
158   //DefaultSubpopData(ICohortChangeBase);
159 end;
160
161 (*
162 procedure TSafeWaterBenefitBase.AdvanceCohortBaseline;
163 begin
164   if not FUseCF then exit;
165   var tn,n,c:double;
166   var a,s,r: integer;
167   for s := gMale to gFemale do begin
168     for r := reWhite to reOther do begin
169       tn:=0;
170       for a := CMaxAge downto 1 do begin
171         n := CohortPopBase[s,r,a] * Q_Gen.V[s,r,a];
172         tn := tn+n;
173         if a=CMaxAge then begin
174           CohortPopBase[s,r,a] := max(CohortPopBase[s,r,a] - n - ICohortChangeBase[s,r,a], 0)
175         end else
176         if a=CMaxAge-1 then begin
177           CohortPopBase[s,r,a+1] := max(CohortPopBase[s,r,a+1] + CohortPopBase[s,r,a] - n - ICohortChangeBase[s,r,a], 0)
178         end else begin
179           CohortPopBase[s,r,a+1] := max(CohortPopBase[s,r,a] - n - ICohortChangeBase[s,r,a], 0);
180         end;
181       end;
182       CohortPopBase[s,r,0] := tn;
183     end;
184   end;
185   DefaultSubpopData(ICohortChangeBase);
186 end;
187 *)
188
189 procedure TSafeWaterBenefitBase.BreakdownModule;
```

```
190 begin
191   {$IfDef DEBUG}
192   fData.DebugLog.LD('Timers',fName,totaltime)
193   {$endif}
194 end;
195
196 procedure TSafeWaterBenefitBase.CollectPWSWideEPValues(const P: TPWSRecord);
197 begin
198   if PostAnnualize then begin
199     for var i := 0 to FData.Inp.RateCount-1 do
200       O[i].Dollars := O[i].Dollars * FData.Inp.PreCalcAnnRate[i, FYears];
201   end;
202 end;
203
204 constructor TSafeWaterBenefitBase.Create(aData: PSWModelData;
205   GlbUnc: TGlobalUncertainty; aNeededContams, aName, aGroup: string; aIteration: integer);
206 begin
207   FData := aData;
208   FIteration := aIteration;
209   FGlb := GlbUnc;
210   FName := aName;
211   FGroup := aGroup;
212   FActive := False;
213   FNeededContams := uppercase(aNeededContams);
214   FInitialized := False;
215   FDebugList := TStringList.Create;
216   if FData.Cfg.DebugOut then begin
217     FDebugList.CommaText := FData.Cfg.DebugList;
218     FDebugList.Sorted := True;
219   end;
220   G85MR:=0.231;  //average of all 85+ cohorts
221   DefaultSubpopData(ICohortChange);
222   DefaultSubpopData(ICohortChangeBase);
223   //DefaultSubpopData(ICohortDeaths);
224   TotalTime:=0;
225   PostAnnualize:=True;
226   setlength(O, FData.Inp.RateCount);
227   FYears := FData.Cfg.EndYear - FData.Cfg.StartYear + 1;
228   setlength(OYearly,FYears);
229   for var i := 0 to High(OYearly) do
230     setlength(OYearly[i],fData.Inp.RateCount);
231 end;
232
233 destructor TSafeWaterBenefitBase.Destroy;
234 begin
235   BreakdownModule;
236   FDebugList.Free;
237   inherited;
238 end;
239
240 function TSafeWaterBenefitBase.Init: boolean;
241 begin
242   result := true;
243   NumContams := FData.Occ.ContamList.Count;
244   setlength(CurConc,NumContams);
245   setlength(AfterConc,NumContams);
246   var S := SplitString(FNeededContams,',');
247   result := FData.Occ.DataAvailable(FNeededContams);
248   if Result then begin
249     SetupContamIndexes;
250     SetupModule;
251   end else
252     if FIteration < 0 then
253       FData.ErrLog.L(FName +' missing needed contaminants');
254   FInitialized := result;
255 end;
256
257 procedure TSafeWaterBenefitBase.ParseConc;
258 begin
259
260 end;
261
262 procedure TSafeWaterBenefitBase.ResetBenefits;
263 begin
```

```
264   var i,y : integer;
265   for y := 0 to high(OYearly) do
266     for i := 0 to fData.Inp.RateCount-1 do begin
267       OYearly[y,i] := Default(TBenefitOutputs);
268     end;
269
270   for i := 0 to fData.Inp.RateCount-1 do begin
271     o[i] := Default(TBenefitOutputs);
272   end;
273 end;
274
275 procedure TSafeWaterBenefitBase.SetPWSWideEPValues(const P: TPWSRecord);
276 begin
277   {$ifdef DEBUG}
278   if FData.Cfg.DebugOut then
279     _d := FDebugList.IndexOf(P.PWSId)>=0
280   else
281     _d := false;
282   {$endif}
283   ResetBenefits;
284 end;
285
286 procedure TSafeWaterBenefitBase.SetupContamIndexes;
287 begin
288   //do nothing in the abstract - convenience function for direct access to conc data
289 end;
290
291 procedure TSafeWaterBenefitBase.SetupModule;
292 begin
293   Q_Gen.LoadFromFile(DataPath+'CohortData\q_gen.csv');
294   FUseCF := FData.Cfg.ICFCF;
295
296 end;
297
298 procedure TSafeWaterBenefitBase.SetupOutputs(Metrics: TCategoryMetrics);
299 begin
300   //Note: Wedge to remove regoption name in the description per Jerry's request
301   var s:=fName;
302   var i:=pos('.',s);
303   if i>0 then Delete(s,1,i);
304
305   var mc := mcSumOnly;
306   var mt := mtBenefitYearly;
307   for i := 0 to FData.Inp.RateCount-1 do begin
308     var tg := floattostr(FData.Inp.Rates[i]);
309     Metrics.AddMetric(fName+'.Dollars',
310                       s + ' Dollars', @o[i].Dollars, mtBenefitDollars,mcSumOnly, False, tg);
311
312     if Fdata.Cfg.YearlyOut then
313       if (not FData.Cfg.ZeroDiscYearly) or (i=0) then
314         for var Yr := 0 to High(OYearly) do begin
315           var yt := inttostr(yr + FData.Cfg.StartYear);
316           var ayt := 'YYYY'+yt;
317           var dyt := ' Y('+yt+')';
318           Metrics.AddMetric(fName+'.Dollars'+ayt,
319                             s + ' Dollars'+dyt, @oYearly[Yr,i].Dollars, mt,mc, False, tg);
320         end;
321   end;
322 end;
323
324 procedure TSafeWaterBenefitBase.SetupUncertainty(Unc: TUncertaintyStudy);
325 begin
326
327 end;
328
329 procedure TSafeWaterBenefitBase.StartTimer;
330 begin
331   aTimer:=TStopWatch.StartNew;
332 end;
333
334 procedure TSafeWaterBenefitBase.TickTimer;
335 begin
336   aTimer.Stop;
337   TotalTime := TotalTime + aTimer.ElapsedMilliseconds;
338 end;
```

```
end.
```

code\BenefitModules\SafeWaterBenefitModules.pas

```
unit SafeWaterBenefitModules;

interface

uses Classes, Types, SafeWaterBenefitBase, Generics.Collections, StrUtils, SysUtils,
     SafeWaterUncertBucket, SafewaterMetrics, SafewaterPWSRecords, SafeWaterOccurrence,
     SafewaterModelData,
     // add all available benefit
     bmPFASCVD, bmPFASStats, bmPFASKidneyC, bmPFASLBW, bmPFASDBP, bmPFASLiverC
      ;

type
  TSafeWaterBenefitModulesList = TObjectList<TSafeWaterBenefitBase>;

  TSafeWaterBenefitModules = class(TSafeWaterBenefitModulesList)
  private
    FActiveCount: integer;
    FIteration : integer;  //in case modules need to know iteration
    FOptionName: string;
  public
    constructor create(aData: PSWModelData; GlbUnc: TGlobalUncertainty; aIteration: integer; aOptionName: string);

    procedure SetActive(ActiveList: TStringDynArray);
    procedure SetupUncertainty(Unc: TUncertaintyStudy);
    procedure SetupOutputs(Metrics: TCategoryMetrics);
    function InitOk : boolean;

    function GetTotalBenefit(const DRIdx: integer) : double;
    procedure SetPWSWideEPValues(const P: TPWSRecord);
    procedure CollectPWSWideEPValues(const P: TPWSRecord); virtual;
    procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit);
    procedure AddEntryPointBenefitsPostProcess(const P: TPWSRecord; const Conc, FinalConc : TContamArray);
    procedure ResetBenefits;
    property ActiveCount: integer read FActiveCount write FActiveCount;
  end;


implementation

{ TSafeWaterBenefitModules }

procedure TSafeWaterBenefitModules.AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit)
;
begin
  for var T in self do begin
    if T.Active then
      T.AddEntryPointBenefits(P,Conc,FinalConc, CostInfo);
  end;
end;

procedure TSafeWaterBenefitModules.AddEntryPointBenefitsPostProcess(const P: TPWSRecord; const Conc, FinalConc : TContamArray);
begin
  for var T in self do begin
    if T.Active then
      T.AddEntryPointBenefitsPostProcess(P, Conc, FinalConc);
  end;
end;

procedure TSafeWaterBenefitModules.CollectPWSWideEPValues(const P: TPWSRecord);
begin
  for var T in self do begin
    if T.Active then
      T.CollectPWSWideEPValues(P);
  end;
end;

constructor TSafeWaterBenefitModules.create(aData: PSWModelData;
```

```
  GlbUnc: TGlobalUncertainty; aIteration: integer; aOptionName: string);
begin
  inherited create(true);
  FIteration := aIteration;
  FOptionName := aOptionName;
  Add(TbmPFASCVD.Create(aData, GlbUnc, 'PFOS,PFOA', aOptionName+'.PFASCVD', 'PFAS',aIteration));
  Add(TbmPFASKidneyC.Create(aData, GlbUnc, 'PFOS,PFOA', aOptionName+'.PFASKidneyCanc', 'PFAS',aIteration,
                            'PFOA','PFASKidneyCanc',False));
  Add(TbmPFASLiverC.Create(aData, GlbUnc, 'PFOS,PFOA', aOptionName+'.PFASLiverCanc', 'PFAS',aIteration,
                           'PFOS','PFASLiverCanc',True));
  Add(TbmPFASLBW.Create(aData, GlbUnc, 'PFOS,PFOA', aOptionName+'.PFASLBW', 'PFAS',aIteration));
  Add(TbmPFASStats.Create(aData, GlbUnc, 'PFOS,PFOA', aOptionName+'.PFASSTATS', 'PFAS',aIteration));
  Add(TbmPFASDBP.Create(aData, GlbUnc, 'PFOS,PFOA', aOptionName+'.PFASDBP', 'PFAS',aIteration));
end;

function TSafeWaterBenefitModules.GetTotalBenefit(const DRIdx: integer): double;
begin
  Result := 0;
  for var T in self do begin
    if T.Active then
      Result:= Result + T.Output[DRIdx].Dollars;
  end;
end;

function TSafeWaterBenefitModules.InitOk: boolean;
begin
  Result:=True;
  for var T in self do begin
    if not T.Active then continue;
    if not T.Init then begin
      Result:=False;
      T.Active := false;
      dec(fActiveCount);
    end;
  end;
end;

procedure TSafeWaterBenefitModules.ResetBenefits;
begin
  for var T in self do begin
    if T.Active then
      T.ResetBenefits;
  end;
end;

procedure TSafeWaterBenefitModules.SetActive(ActiveList: TStringDynArray);
begin
  ActiveCount:=0;
  var L := TSTringList.Create;
  var G := TSTringList.Create;
  for var s in ActiveList  do L.Add(FOptionName+'.'+s);
  for var T in self do begin
    if L.IndexOf(T.Name)>=0 then begin
      T.Active := True;
      inc(fActiveCount);
    end;
  end;
  L.Free;
  G.Free;
end;

procedure TSafeWaterBenefitModules.SetPWSWideEPValues(const P: TPWSRecord);
begin
  for var T in self do begin
    if T.Active then
      T.SetPWSWideEPValues(P);
  end;
end;

procedure TSafeWaterBenefitModules.SetupOutputs(Metrics: TCategoryMetrics);
begin
  for var T in self do begin
    if T.Active then
      T.SetupOutputs(Metrics);
  end;
```

```
end;

procedure TSafeWaterBenefitModules.SetupUncertainty(Unc: TUncertaintyStudy);
begin
  for var T in self do begin
    if T.Active then
      T.SetupUncertainty(Unc);
  end;
end;

end.

```

code\BenefitModules\SafeWaterCohortArray.pas

```
unit SafeWaterCohortArray;

interface

uses Sysutils, Classes, SafewaterGlobals, SafewaterUtility, Generics.Collections,
     System.StrUtils, System.IOUtils ;

type
  TCohortData = record
    FromFile: string;
    V: TSubpopArray;
    procedure LoadFromFile(aFilename: string; Column: integer = 3; RaiseE: boolean=false);
    procedure SaveToFile(aFilename: string);
  end;
  PCohortData = ^TCohortData;

  TCohortIdLookup = TDictionary<string,integer>;
  TCohortDataById = TDictionary<string,TSubpopArray>;
  TCohortDataByIds = class(TObject)
  private
    Lookup: TCohortIdLookup;
    DataB: array of TSubpopArray;
    procedure LoadBinFile(aFilename: string);
    procedure SaveBinFile(aFilename: string; const aData : TCohortDataById);
  public
    constructor create;
    destructor Destroy; override;
    procedure LoadFromFile(aFilename: string;  CheckSumOne: boolean = false);

    function GetCohortData(const id: string): PSubpopArray;
  end;


  TEJProportions = class
    fValues : array of double;
  end;
  TEJProportionsByID = TObjectDictionary<string,TEJProportions>;
  TEJProportionsByIDs = class(TObject)
  private
    fNames: TArray<string>;
    Lookup: TEJProportionsByID;
  public
    constructor create;
    destructor Destroy; override;
    procedure LoadFromFile(aFilename: string);

    function GetAllProportionData(const id: string): TEJProportions;
    function GetProportionData(const id: string; const Name: string): double; overload;

    property Names: TArray<string> read fNames;
  end;



implementation

{ TCohortData }

```

```
procedure TCohortData.LoadFromFile(aFilename: string; Column: integer = 3; RaiseE: boolean=false);
begin
  FromFile := aFilename;
  var ok:boolean;
  V := ReadSubpopData(aFilename,ok,Column, RaiseE);
end;

procedure TCohortData.SaveToFile(aFilename: string);
begin
  const T = TStreamWriter.Create(aFilename,False, TEncoding.ASCII,4096);
  for var s := gMale to gFemale do begin
    for var r := reWhite to reOther do begin
      for var a := 0 to cMaxAge do begin
        var os := '%s,%s,%s,%g';
        var sa,ss,sr : string;
         //convert to strings

        T.WriteLine(Format(os,[sa,ss,sr,V[s,r,a]]));
      end;
    end;
  end;

  T.Free;
end;

{ TCohortDataByIds }

constructor TCohortDataByIds.create;
begin
  Lookup := TCohortIdLookup.Create;
end;

destructor TCohortDataByIds.Destroy;
begin
  Lookup.Free;
  inherited;
end;

function TCohortDataByIds.GetCohortData(const id: string): PSubpopArray;
begin
  Result := nil;
  var idx:integer;
  if Lookup.TryGetValue(id,idx) then begin
    Result := @DataB[idx];
  end;
end;

procedure TCohortDataByIds.LoadBinFile(aFilename: string);
begin
  var T := TBufferedFileStream.Create(aFilename, fmOpenRead + fmShareDenyNone);
  var i: integer;
  Lookup.Clear;
  T.Read(i,sizeof(i));
  setlength(DataB,i);
  var oni: integer;
  var s: string;
  var sx := sizeof(TSubpopArray);
  for oni :=0 to i-1 do begin
    s := ReadStreamStr(T);
    Lookup.Add(s,oni);
    T.ReadBuffer(dataB[oni],sx);
  end;
  T.Free;
end;

procedure TCohortDataByIds.LoadFromFile(aFilename: string;  CheckSumOne: boolean = false);
begin
  var bfile := ChangeFileExt(aFilename,'.bin');

  if TFile.Exists(bfile) then begin
    if not TFile.Exists(aFileName) then begin
      LoadBinFile(bfile);
      exit;
    end;
```

```
133     if Fileage(bfile) > FileAge(aFilename) then begin
134       LoadBinFile(bfile);
135       exit;
136     end;
137   end;
138   //File doesn't exist - so exit with no data
139   if not TFile.Exists(aFileName) then exit;
140
141   var aData := TCohortDataById.Create;
142   var s,r,a, six, rix,aix : integer;
143   var id, IL: string;
144   var T := TStreamReader.Create(aFilename);
145   var F: TArray<string>;
146   var CD: TSubpopArray;
147   var ReadOk: boolean := True;
148   var v: double;
149
150   //TODO fix this hacky mess - enforce opinion on column names
151   var hdr := T.ReadLine;
152   hdr := lowercase(StringReplace(hdr,'"','',[rfReplaceAll]));
153   var TSL := TStringList.Create;
154   TSL.CommaText := hdr;
155   six := TSL.IndexOf('sex');
156   aix := TSL.IndexOf('age');
157   rix := TSL.IndexOf('race_eth');
158   if rix<0 then  rix := TSL.IndexOf('race/ethicity');
159   TSL.Free;
160
161   while not T.EndOfStream do begin
162     IL := T.ReadLine;
163     IL := StringReplace(IL,'"','',[rfReplaceAll]);
164     F := SplitString(IL,',');
165     id := F[0];
166     //remove leading zeroes
167     if id[1] = '0' then delete(id,1,1);
168     if not aData.TryGetValue(id,CD) then begin
169       DefaultSubpopData(CD);
170     end;
171
172     s := -1;
173     if lowercase(F[six]) = 'male' then s := gMale else
174     if lowercase(F[six]) = 'female' then s := gFemale;
175     r := -1;
176     var cc := lowercase(F[rix]);
177     if cc = 'white' then r := reWhite else
178     if cc = 'other' then r := reOther else
179     if cc = 'black' then r := reBlack else
180     if cc = 'hispanic' then r := reHispanic;
181     try
182       a := strtoint(F[aix]);
183     except
184       a := -1;
185     end;
186
187     if (s>-1) and (r>-1) and (a>=0) and (a<=100) then
188       try
189         v := strtofloat(F[4]);
190         if a>=85 then
191           CD[s,r,85] := CD[s,r,85] + v
192         else
193           CD[s,r,a] := v;
194       except
195         ReadOK := false;
196       end
197     else
198       ReadOK := false;
199     aData.AddOrSetValue(id,CD)
200   end;
201   T.Free;
202
203   //find errors and normalize
204
205   if CheckSumOne then begin
206     for var SPD in aData.Keys do begin
207       CD := aData.Items[sPD];
```

```
      var sum := SumSubpopData(CD);
      if (sum<0.98) or (sum>1.02) then
        CSL(Format('Bad sum %s,  %g',[SPD, sum]));
      if (sum<>0) then
        NormalizeSubpopdata(CD);
      aData.AddOrSetValue(SPD,CD);
    end;
  end;

  SaveBinFile(bFile, aData);
  aData.Free;
  LoadBinFile(bfile);
end;

procedure TCohortDataByIds.SaveBinFile(aFilename: string; const aData : TCohortDataById);
begin
  var T := TBufferedFileStream.Create(aFilename, fmCReate);
  var i: integer := aData.Count;
  T.Write(i,sizeof(i));
  i:=0;
  var CD: TSubPopArray;
  var sx := sizeof(TSubpopArray);
  var s: string;
  for S in aData.Keys do begin
    WriteStreamStr(T,s);
    CD := aData.Items[s];
    T.WriteBuffer(CD, sx);
  end;
  T.Free;
end;

{ TEJProportionsByIDs }

constructor TEJProportionsByIDs.create;
begin
  Lookup:=TEJProportionsByID.Create([doOwnsValues]);
end;

destructor TEJProportionsByIDs.Destroy;
begin
  Lookup.Free;
  inherited;
end;


function TEJProportionsByIDs.GetAllProportionData(const id: string): TEJProportions;
begin
  Lookup.TryGetValue(id,result);
end;

function TEJProportionsByIDs.GetProportionData(const id: string; const Name: string): double;
begin
//  TODO if necc
end;

procedure TEJProportionsByIDs.LoadFromFile(aFilename: string);
begin
  var inf := TStreamReader.Create(aFilename,TEncoding.ASCII,false,16*4096);
  var s := inf.ReadLine;
  fNames := SplitString(s,',');
  var inl : TArray<string>;
  var v : TEJProportions;
  while not inf.EndOfStream do begin
    s := inf.ReadLine;
    inl := SplitString(s,',');
    v := TEJproportions.Create;
    setlength(v.fValues, length(fNames));
    for var i := 1 to length(inl) -1 do
      v.fValues[i] := inl[i].ToDouble();
    Lookup.AddOrSetValue(inl[0],v);
  end;
  inf.Free;
end;

```

```
end.

```

code\BenefitModules\bmPFASAbstractCancSC.pas

```
unit bmPFASAbstractCancSC;

interface

uses SysUtils, Classes, Math, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
     SafeWaterUtility, SafeWaterUncertBucket, SafewaterMetrics, SafeWaterGlobals,
     SafeWaterCalcDist, PFASSerumCalc, SafewaterCohortArray,
     bmPFASBase ;


//sets length of data in files for follow periods
const FollowYears = 11;

type
  TbmCancOutputs=record
    Fatal, Nonfatal: double;
  end;
  TbmCancDollarOutputs=record
    FatalDollars, NonfatalDollars: double;
    NonfatalDollarsWTP: double;
  end;

  TYrCohortData =  array[0..FollowYears-1] of TCohortData;

  TbmAbstractCancSC = class(TbmPFASBase)
  protected
    //these are only necc for SetBAseProbs - could be read from a preprocesssed file
    //in the future
    ps_dist, ps_reg, ps_uns, ps_local: TYrCohortData;
    dist,reg,uns,local : TCohortData;

    _KC : TbmCancOutputs;
    _KCD : array of TbmCancDollarOutputs;
    _KCC : array[reWhite..reOther] of TbmCancOutputs;
    _KCI : array of TbmCancOutputs;

    //get all years of changes at once;
    ybase, yscen: TYearlySubPopArray;

    AllMortDollars, AllNFDollars, AllNFDollarsWTP: array of TYearlySubPopArray;
    C_T, CumCases : TSubpopArray;
    BaseProbDone: boolean;

    StartAge, TotYears: integer;  //age to start calc - just a time saver...
    C_Share, Slope, COI_year_1,COI_year_2,COI_year_3 : TDistributed;

    SumProbSurvAge, GenProb, ProbSurvAge, MortShare : TCohortData;
    ProbSurvYear, ProbMortYear, ProbMortInYear, SumMortYear : TYearlySubPopArray;
    fLabel, fContam: string;
    fUseWTP: boolean;
    WTP : array[2024..2150] of double;  //yearly values - ungeneral
    bWTP: double;
    sP : TSerumCalc;
    fCIX: integer;  //index of cotaminant this applies to

    procedure LoadParms; virtual;
    procedure SetupModule; override;
    procedure SetDollarStream(); virtual;
    procedure SetBaseProbs();
    procedure CalcCases(const P: TPWSRecord; const OnYr, OnChangeYear: integer);
    procedure UpdateWTPYearly;
  public
    constructor Create(aData: PSWModelData; GlbUnc: TGlobalUncertainty;
      aNeededContams, aName, aGroup: string; aIteration: integer;
      aContam,aLabel: string; aUseWTP: boolean); overload;

    procedure ResetBenefits(); override;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
```

```
69
70      procedure CollectPWSWideEPValues(const P: TPWSRecord); override;
71      procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit); override;
72    end;
73
74  implementation
75
76  { TbmAbstractCancSC }
77
78  constructor TbmAbstractCancSC.Create(aData: PSWModelData;
79    GlbUnc: TGlobalUncertainty; aNeededContams, aName, aGroup: string; aIteration: integer;
80    aContam,aLabel: string; aUseWTP: boolean);
81  begin
82    inherited Create(aData, GlbUnc, aNeededContams, aName, aGroup, aIteration);
83    fLabel := aLabel; // 'PFASKidneyCanc';
84    fContam := acontam; //'PFOA';
85    fCIX := FData.Occ.ContamList.IndexOf(fContam);
86    //TODO - in config
87    fUseWTP:=aUseWtp;
88  end;
89
90  procedure TbmAbstractCancSC.UpdateWTPYearly;
91  begin
92    for var y := low(WTP) to high(WTP) do begin
93      WTP[Y]:=bWTP * power(1.00705,Y-2024);
94    end;
95  end;
96
97  procedure TbmAbstractCancSC.AddEntryPointBenefits(const P: TPWSRecord;
98    const Conc, FinalConc: TContamArray; const CostInfo: TCostInfo2Benefit);
99  begin
100   //always call inherited
101   inherited;
102   //no change...
103   if CurConc[fCIX] = AfterConc[fCIX] then exit;
104
105   {$IfDef DEBUG}
106   StartTimer;
107   {$endif}
108
109   if not BaseProbDone then SetBaseProbs();
110
111   var Yr, OnYr, YearsSinceChange : integer;
112   for Yr := FData.Cfg.PFASChangeYear to FData.Cfg.EndYear do begin
113     YearsSinceChange := Yr - FData.Cfg.PFASChangeYear;
114     sP.GetBeforeAfterSerum(ybase[YearsSinceChange], yscen[YearsSinceChange], YearsSinceChange, CurConc[fCIX], AfterConc[fCIX], FData.Cfg.DW_RSC, 0);
115   end;
116   sP.ConvertToLifetimeAvg(ybase, yscen, FData.Cfg.EndYear - FData.Cfg.PFASChangeYear + 1);
117
118   for Yr := FData.Cfg.PFASChangeYear to FData.Cfg.EndYear do begin
119     OnYr := Yr - FData.Cfg.StartYear;
120     YearsSinceChange := Yr - FData.Cfg.PFASChangeYear;
121     CalcCases(P, OnYr, YearsSinceChange);
122     AdvanceCohort;
123   end;
124
125   {$IfDef DEBUG}
126   TickTimer;
127   {$endif}
128 end;
129
130 procedure TbmAbstractCancSC.CalcCases(const P: TPWSRecord; const OnYr, OnChangeYear: integer);
131 begin
132   var a,s,r,ix : integer;
133   var d,dnf,dnfwtp,df,rr,cc,lcc,aoc,arr, dprob,mcc : double;
134   var nf,f, sf, snf, sac, lrr,allcases  : double;
135
136   d := 0; df := 0; dnf := 0; dnfwtp := 0;
137   sf := 0; snf:=0; sac:=0;
138   const YrsRemain = FData.Cfg.EndYear -  FData.Cfg.PFASChangeYear - OnChangeYear;
139   const YrsTot = FData.Cfg.EndYear -  FData.Cfg.PFASChangeYear + 1;
140   ix := min(10, YrsRemain);
141
142   for s := gMale to gFemale do begin
143     for r := reWhite to reOther do begin
```

```
144       mcc := 0;
145       for a := StartAge to CMaxAge do begin
146         if CohortPop[s,r,a] =  0 then continue;
147
148         rr := 1 - Slope.fValue *
149                  (ybase[OnChangeYear][s,r,a] - yscen[OnChangeYear][s,r,a]);
150
151         //Change PAF to Global 9/18 -
152         rr := max(1 - fGlb.PAF,rr);
153         cc := max(rr * CumCases[s,r,a],0);
154         mcc := max(cc, mcc);
155
156         if (OnChangeYear>0) and (a>0) then begin
157           lrr  := 1 -  Slope.fValue
158                  * (ybase[OnChangeYear-1][s,r,a-1] - yscen[OnChangeYear-1][s,r,a-1]);
159           lrr := max(1 - fGlb.PAF,lrr);
160           lcc := max(lrr * CumCases[s,r,a-1],0);
161         end else begin
162           if a=0 then
163             lcc:=0
164           else
165             lcc := CumCases[s,r,a-1];
166           lrr := 1;
167         end;
168
169         aoc :=  max(mcc - lcc, 0);
170
171         if C_T[s,r,a]>0 then
172           arr := min(aoc / C_T[s,r,a],1)
173         else
174           arr := 1;
175
176         dprob := ProbSurvAge.V[s,r,a] * (1 - arr);
177
178         allcases := (CohortPop[s,r,a]
179                     - CohortPop[s,r,a] * GenProb.V[s,r,a]
180                     + CohortPop[s,r,a] * GenProb.V[s,r,a] * MortShare.V[s,r,a]
181                     ) * dprob;
182
183         ICohortChange[s,r,a] := allcases;
184
185         f := allcases * SumMortYear[ix][s,r,a];
186         nf := allcases -  f;
187
188 sf:=sf+f;
189 snf:=snf+nf;
190 sac:=sac+allcases;
191         for var i := 0 to FData.Inp.RateCount-1 do begin
192           (*
193           df := df + allcases * AllMortDollars[OnYr][s,r,a];
194           dnf := dnf + allcases * AllNFDollars[OnYr][s,r,a];
195           dnfwtp := dnfwtp + allcases * AllNFDollarsWTP[OnYr][s,r,a];
196           d := d + allcases * (AllMortDollars[OnYr][s,r,a] + AllNFDollars[OnYr][s,r,a]);
197           *)
198           //accumulate
199           _i(O[i].Dollars,  allcases * (AllMortDollars[i][OnYr][s,r,a] + AllNFDollars[i][OnYr][s,r,a]));
200           _i(OYearly[OnYr,i].Dollars,  allcases * (AllMortDollars[i][OnYr][s,r,a] + AllNFDollars[i][OnYr][s,r,a]));
201           _i(_KCD[i].FatalDollars,allcases * AllMortDollars[i][OnYr][s,r,a]);
202           _i(_KCD[i].NonFatalDollars,allcases * AllNFDollars[i][OnYr][s,r,a]);
203           _i(_KCD[i].NonFatalDollarsWTP,allcases * AllNFDollarsWTP[i][OnYr][s,r,a]);
204
205         end;
206
207         _i(_KC.Fatal,f);
208         _i(_KC.NonFatal,nf);
209         _i(_KCC[r].NonFatal,nf);
210         _i(_KCC[r].Fatal,f);
211         for var ii := 1 to length(EJProps)-1 do begin
212           _i(_KCI[ii].NonFatal, nf * EJProps[ii]);
213           _i(_KCI[ii].Fatal, f * EJProps[ii]);
214         end;
215
216        {$ifdef DEBUG}
217        if (_d) and (fIteration=0) then begin
```

```
218          var aage: string;
219          var bc,ac: double;
220          if fContam='PFOS' then begin
221            bc := bcPFOS; ac:= acPFOS;
222          end else begin
223            bc := bcPFOA; ac:= acPFOA;
224          end;
225
226          if (OnYr)>a then
227            aage:='A'+inttostr(OnYr-a)
228          else
229            aage := inttostr(A-OnYr);
230
231          var chs := s.ToString+'_'+r.ToString+'_'+aage;
232          FData.DebugLog.L(fLabel+'_icf',Format('%d,%s,%g,%g,'+
233                            '"%s",%d,%d,%d,%g,'+
234                            '%g,%g,%g,'+
235                            '%g,%g,%g,'+
236                            '%g,%g,%g,%g,%g,%g,%g,'+
237                            '%g,%g,%g,'+
238                            '%g,%g,%d,%d,'+
239                            '%g,%g,%g,%g,%g,%g,%g',
240                           [OnChangeYear,P.PWSId,bc, ac,
241                           chs,s,r,a,CohortPop[s,r,a],
242                           yscen[OnChangeYear][s,r,a],ybase[OnChangeYear][s,r,a],(yscen[OnChangeYear][s,r,a]-ybase[OnChangeYear][s,r,a]),
243                           SumProbSurvAge.V[s,r,a], SumProbSurvAge.V[s,r,max(a-1,0)],rr,
244                           lrr, cc, aoc, arr, dprob, f, nf,
245                           allcases, SumMortYear[ix][s,r,a], ProbSurvYear[ix][s,r,a],
246                           ProbSurvAge.V[s,r,a],ProbSurvAge.V[s,r,max(a-1,0)], ix, YrsRemain,
247                           Cumcases[s,r,a],Cumcases[s,r,max(a-1,0)], mcc, C_T[s,r,a],
248                           GenProb.V[s,r,a],MortShare.V[s,r,a],CohortPopTotal[s,r,a]
249                           ]));
250
251        end;
252       {$endif}
253
254       end;
255     end;
256   end;
257   {$ifdef DEBUG}
258     FData.DebugLog.L(fLabel+'_pws',Format('%d,%s,%g,%g,%g,%g,%g,%g',
259                       [OnYr,P.Pwsid,sac,snf,sf,dnf,df]));
260   {$endif}
261
262 end;
263
264 procedure TbmAbstractCancSC.CollectPWSWideEPValues(const P: TPWSRecord);
265 begin
266   inherited;
267   //annualizing
268   for var i := 0 to FData.Inp.RateCount-1 do begin
269     _KCD[i].FatalDollars := _KCD[i].FatalDollars *  FData.Inp.PreCalcAnnRate[i, FData.Cfg.EndYear - FData.Cfg.StartYear + 1];
270     _KCD[i].NonFatalDollars := _KCD[i].NonFatalDollars * FData.Inp.PreCalcAnnRate[i, FData.Cfg.EndYear - FData.Cfg.StartYear + 1];
271     _KCD[i].NonFatalDollarsWTP := _KCD[i].NonFatalDollarsWTP * FData.Inp.PreCalcAnnRate[i, FData.Cfg.EndYear - FData.Cfg.StartYear + 1];
272   end;
273 end;
274
275 procedure TbmAbstractCancSC.LoadParms;
276 begin
277 end;
278
279 procedure TbmAbstractCancSC.ResetBenefits;
280 begin
281   inherited;
282   _KC := Default(TbmCancOutputs);
283   for var r  := reWhite  to reOther do
284     _KCC[r] := Default(TbmCancOutputs);
285   for var r  := 0 to high(_KCI) do
286     _KCI[r] := Default(TbmCancOutputs);
287   for var i := 0 to FData.Inp.RateCount-1 do
288     _KCD[i] := Default(TbmCancDollarOutputs);
289 end;
290
291 procedure TbmAbstractCancSC.SetBaseProbs();
292 begin
```

```
293   //precalc rates....
294   var AllYrs := FData.Cfg.EndYear - FData.Cfg.StartYear + 1;
295   var s,r,a,y, nix, i: integer;
296   var dt, od, sd, sl, su, sr, spd, spu, spr, spl, ap, apa, cd : double;
297
298   //init
299   for y := 0 to FollowYears - 1  do begin
300     DefaultSubPopData(ProbSurvYear[y]);
301     DefaultSubPopData(ProbMortYear[y]);
302     DefaultSubPopData(SumMortYear[y]);
303     DefaultSubPopData(ProbMortInYear[y]);
304   end;
305   for i := 0 to FData.Inp.RateCount-1 do begin
306     for y := 0 to AllYrs - 1  do begin
307       DefaultSubPopData(AllMortDollars[i][y]);
308       DefaultSubPopData(AllNFDollars[i][y]);
309       DefaultSubPopData(AllNFDollarsWTP[i][y]);
310     end;
311   end;
312   DefaultSubPopData(C_T);
313   DefaultSubPopData(CumCases);
314
315   for s := gMale to gFemale do begin
316     for r := reWhite to reOther do begin
317       apa:=1;
318       for a := StartAge to CMaxAge do begin
319         //calculate expected propotion of cummulative cancer cases - baseline
320         dt := apa * (GenProb.V[s,r,a] * ( 1 - MortShare.V[s,r,a]));
321         C_T[s,r,a] := ProbSurvAge.V[s,r,a] * (apa -dt) ;
322         apa := apa - C_T[s,r,a] -  dt;
323
324         spd:=Dist.V[s,r,a];
325         spu:=Uns.V[s,r,a];
326         spl:=Local.V[s,r,a];
327         spr:=Reg.V[s,r,a];
328         ap := 1;
329         for y := 0 to FollowYears -1  do begin
330           nix := min(a+y+1,CMaxAge);
331           od := GenProb.V[s,r,nix] * ( 1 - MortShare.V[s,r,nix]);
332           ap := ap - (ap * GenProb.V[s,r,nix]);
333           spl := spl - (spl * od);
334           spr := spr - (spr * od);
335           spu := spu - (spu * od);
336           spd := spd - (spd * od);
337
338           sd := ps_dist[y].V[s,r,a];
339           su := ps_uns[y].V[s,r,a];
340           sr := ps_reg[y].V[s,r,a];
341           sl := ps_local[y].V[s,r,a];
342
343           cd := (spd * (1-sd )) + (spu * (1-su )) +
344                 (spr * (1-sr )) + (spl * (1-sl ));
345
346           ProbMortYear[y][s,r,a] := max(cd, 0)  ;
347
348           ProbSurvYear[y][s,r,a] := (spd * sd) +
349                                     (spu * su) +
350                                     (spr * sr) +
351                                     (spl * sl) ;
352
353           if (y>0) then begin
354             ProbMortInYear[y][s,r,a] := max(ProbMortYear[y][s,r,a] -
355                                          ProbMortYear[y-1][s,r,a],0) ;
356
357             SumMortYear[y][s,r,a] := SumMortYear[y-1][s,r,a] + ProbMortInYear[y][s,r,a];
358           end;
359
360  //         spl := spl - (spl * (1-sl ));
361 //          spr := spr - (spr * (1-sr ));
362 //          spu := spu - (spu * (1-su ));
363 //          spd := spd - (spd * (1-sd ));
364
365           if (a+y+1) > 100 then Break;
366         end;
```

```
367       end;
368     end;
369   end;
370
371   for s := gMale to gFemale do begin
372     for r := reWhite to reOther do begin
373       for a := StartAge to CMaxAge do begin
374         if a=StartAge then
375          CumCases[s,r,a]:=C_T[s,r,a]
376        else
377          CumCases[s,r,a]:=C_T[s,r,a] + CumCases[s,r,a-1]
378       end;
379     end;
380   end;
381
382
383   {$ifdef DEBUG}
384   var str,str2, str3,str4, str5, str6:string;
385   str:='sex,race,age';
386   for y := 0 to FollowYears -1  do begin
387     str:=str+',y'+y.tostring;
388   end;
389   FData.DebugLog.L(fLabel+'_mr',str);
390   FData.DebugLog.L(fLabel+'_mdpy',str);
391   FData.DebugLog.L(fLabel+'_sr',str);
392   FData.DebugLog.L(fLabel+'_miy',str);
393   FData.DebugLog.L(fLabel+'_basecumcases','sex,race,age, cumcases');
394   FData.DebugLog.L(fLabel+'_base_c_t','sex,race,age, c_t');
395   str:='sex,race,age';
396   for y := 0 to AllYrs -1  do begin
397     str:=str+',y'+y.tostring;
398   end;
399   FData.DebugLog.L(fLabel+'_md',str);
400   FData.DebugLog.L(fLabel+'_sd',str);
401   {$endif}
402
403   SetDollarStream();
404   BaseProbDone := True;
405 end;
406
407 procedure TbmAbstractCancSC.SetDollarStream;
408 begin
409   var AllYrs := FData.Cfg.EndYear - FData.Cfg.StartYear + 1;
410   var OnYr, CalYear,s,r,a,y :integer;
411   var nfd, fd, ad, nfdwtp: double;
412   for s := gMale to gFemale do begin
413     for r := reWhite to reOther do begin
414       for a := StartAge to CMaxAge do begin
415         for OnYr := 0 to AllYrs -1  do begin
416           CalYear := FData.Cfg.StartYear + OnYr;
417           for y := 0 to Min(a+AllYrs,95)  do begin
418             if OnYr+y > AllYrs then break;
419
420             nfdwtp := 0;
421             if fUseWTP then begin
422               if y=0 then
423                 nfd := WTP[CalYear]
424               else
425                 nfd:=0;
426               nfdwtp := nfd;
427             end else begin
428               if y=0 then begin
429                 nfd := COI_Year_1.fValue;
430                 nfdwtp := WTP[CalYear];
431               end else
432               if y=1 then
433                 nfd := COI_Year_2.fValue
434               else
435                 nfd := COI_Year_3.fValue;
436             end;
437
438             if y > 10 then begin
439               ad := ad * (1 - GenProb.V[s,r,min(a+y,CMaxAge)]);
440             end else
441               ad := ProbSurvYear[y][s,r,a];
```

```
            nfd := nfd * ad;
            nfdwtp := nfdwtp * ad;

            for var i:=0 to FData.Inp.RateCount-1 do begin
              AllNFDollars[i][OnYr][s,r,a] := AllNFDollars[i][OnYr][s,r,a] + nfd / FData.Inp.PreCalcDiscRate[i,OnYr + y+FData.Cfg.PeriodAdjust];
              AllNFDollarsWTP[i][OnYr][s,r,a] := AllNFDollarsWTP[i][OnYr][s,r,a] + nfdwtp / FData.Inp.PreCalcDiscRate[i,OnYr + y+FData.Cfg.PeriodAdjust];
              //no deaths in year 0
              if (y = 0) or (y>10) then continue;
              fd := (ProbMortInYear[y][s,r,a]) * FGlb.VSLY[CalYear] / FData.Inp.PreCalcDiscRate[i,OnYr + y+FData.Cfg.PeriodAdjust];
              AllMortDollars[i][OnYr][s,r,a] := AllMortDollars[i][OnYr][s,r,a] + fd;
            end;
          end;
        end;
      end;
    end;
  end;

  //Note:  moved 9/23 - check output
  //TODO needs updated with multi discount rate stuff
  {$ifdef DEBUG}
  (*
  var str,str2, str3,str4, str5, str6:string;
  for s := gMale to gFemale do begin
    for r := reWhite to reOther do begin
      for a := StartAge to CMaxAge do begin
        str:=s.ToString+','+r.ToString+','+a.ToString;
        str2:=s.ToString+','+r.ToString+','+a.ToString;
        str3:=s.ToString+','+r.ToString+','+a.ToString;
        str4:=s.ToString+','+r.ToString+','+a.ToString;
        str5:=s.ToString+','+r.ToString+','+a.ToString;
        str6:=s.ToString+','+r.ToString+','+a.ToString;
        FData.DebugLog.L(fLabel+ '_base_c_t',str+','+floattostr(C_T[s,r,a]));
        FData.DebugLog.L(fLabel+'_basecumcases',str+','+floattostr(CumCases[s,r,a]));
        for y := 0 to AllYrs -1  do begin
          if y<FollowYears then begin
            str := str+','+floattostr(SumMortYear[y][s,r,a]);
            str2 := str2+','+floattostr(ProbSurvYear[y][s,r,a]);
            str5 := str5+','+floattostr(ProbMortInYear[y][s,r,a]);
            str6 := str6+','+floattostr(ProbMortYear[y][s,r,a]);
          end;
          str3 := str3+','+floattostr(AllNFDollars[y][s,r,a]);
          str4 := str4+','+floattostr(AllMortDollars[y][s,r,a]);
        end;
        FData.DebugLog.L(fLabel+'_mr',str);
        FData.DebugLog.L(fLabel+'_sr',str2);
        FData.DebugLog.L(fLabel+'_sd',str3);
        FData.DebugLog.L(fLabel+'_md',str4);
        FData.DebugLog.L(fLabel+'_miy',str5);
        FData.DebugLog.L(fLabel+'_mdpy',str6);
      end;
    end;
  end;
  *)
  {$endif}

end;

procedure TbmAbstractCancSC.SetupModule;
begin
  inherited;
  StartAge := 0;
  TotYears := FData.Cfg.EndYear - FData.Cfg.StartYear;
  //TODO guard against not finding calculators...
  sP := FData.SerumCalc.GetSerumCalc(fContam);
  setlength(ybase,FData.Cfg.EndYear - FData.Cfg.PFASChangeYear + 1);
  setlength(yscen,FData.Cfg.EndYear - FData.Cfg.PFASChangeYear + 1);

  setlength(ProbSurvYear,FollowYears);
  setlength(ProbMortInYear,FollowYears);
  setlength(ProbMortYear,FollowYears);
  setlength(SumMortYear,FollowYears);
  setlength(AllNFDollars,FData.Inp.RateCount);
  setlength(AllNFDollarsWTP,FData.Inp.RateCount);
  setlength(AllMortDollars,FData.Inp.RateCount);
```

```
516   for var i := 0 to Fdata.Inp.RateCount-1 do begin
517     setlength(AllNFDollars[i],FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
518     setlength(AllNFDollarsWTP[i],FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
519     setlength(AllMortDollars[i],FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
520   end;
521
522   LoadParms;
523 end;
524
525 procedure TbmAbstractCancSC.SetupOutputs(Metrics: TCategoryMetrics);
526 begin
527   inherited;
528   Metrics.AddMetric(fName+'.Fatal',
529                     fLabel+' Fatal Cases', @_KC.Fatal, mtBenefitCases,mcSumOnly, False);
530   Metrics.AddMetric(fName+'.NonFatal',
531                     fLabel+' Non Fatal Cases', @_KC.NonFatal, mtBenefitCases,mcSumOnly, False);
532
533   setlength(_KCD,FData.Inp.RateCount);
534   for var i := 0 to FData.Inp.RateCount-1 do begin
535     var tg := floattostr(FData.Inp.Rates[i]);
536     Metrics.AddMetric(fName+'.FatalDollars',
537                       fLabel+' Fatal Dollars', @_KCD[i].FatalDollars, mtBenefitDollars,mcSumOnly, False, tg);
538     Metrics.AddMetric(fName+'.NonFatalDollars',
539                       fLabel+' Non Fatal Dollars', @_KCD[i].NonFatalDollars, mtBenefitDollars,mcSumOnly, False, tg);
540     Metrics.AddMetric(fName+'.NonFatalDollarsWTP',
541                       fLabel+' Non Fatal Dollars WTP', @_KCD[i].NonFatalDollarsWTP, mtBenefitDollars,mcSumOnly, False, tg);
542   end;
543
544   setlength(_KCI,length(fData.EJProps.Names));
545   if FData.Cfg.PFASEJOutput then begin
546     for var i := 1 to length(fData.EJProps.Names) - 1 do begin
547       var s := fData.EJProps.Names[i];
548       Metrics.AddMetric(fName+'.Fatal'+s,
549                         fLabel+ ' Fatal Cases '+s, @_KCI[i].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
550       Metrics.AddMetric(fName+'.NonFatal'+s,
551                         fLabel+' Non Fatal Cases '+s, @_KCI[i].NonFatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
552     end;
553
554
555     Metrics.AddMetric(fName+'.FatalWhite',
556                       fLabel+' Fatal Cases White', @_KCC[reWhite].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
557     Metrics.AddMetric(fName+'.FatalBlack',
558                       fLabel+' Fatal Cases Black', @_KCC[reBlack].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
559     Metrics.AddMetric(fName+'.FatalOther',
560                       fLabel+' Fatal Cases Other', @_KCC[reOther].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
561     Metrics.AddMetric(fName+'.FatalHispanic',
562                       fLabel+' Fatal Cases Hispanic', @_KCC[reHispanic].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
563     Metrics.AddMetric(fName+'.NonFatalWhite',
564                       fLabel+' Non Fatal Cases White', @_KCC[reWhite].NonFatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
565     Metrics.AddMetric(fName+'.NonFatalBlack',
566                       fLabel+' Non Fatal Cases Black', @_KCC[reBlack].NonFatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
567     Metrics.AddMetric(fName+'.NonFatalOther',
568                       fLabel+' Non Fatal Cases Other', @_KCC[reOther].NonFatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
569     Metrics.AddMetric(fName+'.NonFatalHispanic',
570                       fLabel+' Non Fatal Cases Hispanic', @_KCC[reHispanic].NonFatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
571   end;
572
573 end;
574
575
576 end.
577
578
579
```

code\BenefitModules\bmPFASBase.pas

```
1 unit bmPFASBase;
2
3 interface
4
5 uses SysUtils, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
6      SafeWaterUtility, SafeWaterGlobals, Generics.Collections, SafewaterConfig,
```

```
7       SafeWaterUncertBucket, SafewaterMetrics,
8       PFASSerumCalc;
9
10
11  type
12
13    TbmPFASBase = class(TSafeWaterBenefitBase)
14      EJProps: array of double;  // for free form EJ output
15      EJInd: double;  //set to include PWS in EJ output
16
17      //contam indexes
18      iPFOS,iPFOA : integer;
19      //conc, before and after
20      bcPFOS,bcPFOA : double;
21      acPFOS,acPFOA : double;
22      //serum calculators....
23      sPFOA, sPFOS: TSerumCalc;
24      //indicates no change in conc - descendants can skip calculation if desired
25      PFOAChange, PFOSChange: boolean;
26
27      basePFOA, basePFOS: TSubPopArray;
28      scenPFOA, scenPFOS: TSubPopArray;
29
30      destructor Destroy; override;
31
32      procedure ResetBenefits(); override;
33      procedure SetupUncertainty(Unc: TUncertaintyStudy); override;
34      procedure SetupOutputs(Metrics: TCategoryMetrics); override;
35
36      procedure SetupContamIndexes; override;
37      procedure ParseConc; override;
38
39      procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit); override;
40    end;
41
42
43  implementation
44
45  { TbmPFASBase }
46
47  procedure TbmPFASBase.AddEntryPointBenefits(const P: TPWSRecord; const Conc,
48    FinalConc: TContamArray; const CostInfo: TCostInfo2Benefit);
49  begin
50    inherited;
51    EJInd := P.EJInd;
52    // retrieve EJ props
53    var E := FData.EJProps.GetAllProportionData(P.PWSId);
54    var i: integer;
55    if Assigned(E) then
56      for i:=0 to length(EJProps) - 1 do  EJProps[i] := E.FValues[i]
57    else
58      for i:=0 to length(EJProps) - 1 do  EJProps[i] := 0;
59
60    for i := FData.Cfg.StartYear to FData.Cfg.PFASChangeYear-1 do begin
61      AdvanceCohort;
62    end;
63  end;
64
65  destructor TbmPFASBase.Destroy;
66  begin
67    inherited;
68  end;
69
70  procedure TbmPFASBase.ParseConc;
71  begin
72    inherited;
73    if iPFOS>-1 then begin
74      bcPFOS := CurConc[iPFOS];
75      acPFOS := AfterConc[iPFOS];
76    end;
77    if iPFOA>-1 then begin
78      bcPFOA := CurConc[iPFOA];
79      acPFOA := AfterConc[iPFOA];
80    end;
81
```

```
  PFOACHange := acPFOA <> bcPFOA;
  PFOSCHange := acPFOS <> bcPFOS;

  {$ifdef DEBUG}
  if _d then begin
    FData.DebugLog.L('pfasbase',format('pfoa(b,a): %g, %g   pfos (b,a): %g, %g',[bcPFOA, acPFOA, bcPFOS, acPFOS]));
  end;
  {$endif}

  //if not (PFOACHange or PFOSCHange) then exit;


end;

procedure TbmPFASBase.ResetBenefits;
begin
  inherited;

end;

procedure TbmPFASBase.SetupContamIndexes;
begin
  inherited;
  //TODO maybe make these calls a little neater
  iPFOS := FData.Occ.ContamList.IndexOf('PFOS');
  iPFOA := FData.Occ.ContamList.IndexOf('PFOA');

  sPFOS := FData.SerumCalc.GetSerumCalc('PFOS');
  sPFOA := FData.SerumCalc.GetSerumCalc('PFOA');
end;



procedure TbmPFASBase.SetupOutputs(Metrics: TCategoryMetrics);
begin
  inherited;
  setlength(EJProps, length(fData.EJProps.Names));
end;

procedure TbmPFASBase.SetupUncertainty(Unc: TUncertaintyStudy);
begin
  inherited;

end;

end.

```

code\BenefitModules\bmPFASCVD.pas

```
unit bmPFASCVD;

interface

uses SysUtils, Classes,
     Math,
     SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
     SafeWaterUtility, SafeWaterUncertBucket, SafewaterMetrics, SafeWaterGlobals,
     SafeWaterCalcDist, PFASSerumCalc, SafewaterCohortArray,
     bmPFASBase ;

const htT = 0; htU = 1; htNon = 2;


type
  TbmCVDOutputs=record
    Fatal, MI, ISt, PostAcute: double;
  end;

  TbmPFASCVD = class(TbmPFASBase)
  protected
    _CVD : TbmCVDOutputs;
    _CVDC : array[reWhite..reOther] of TbmCVDOutputs;
```

```
24      _CVDI : array of TbmCVDOutputs;
25
26      OptionProb, BaseLineProb: TSubpopArray;
27      //Base Probs - can calculate once.
28      HazBaseProb: array [htT..htNon] of TCohortData;
29      BaseProbDone: boolean;
30      DeltaTC, DeltaSBP, DeltaHDLC: TCohortData;
31
32      //data read from files
33      PA_MI, PA_IS : array [1..6] of TCohortData;
34      HTR: array [htT..htNon] of TCohortData; //hypertensive rates
35
36      Tau: array [htT..htNon] of TCohortData; //cholesterol
37      SBP: array [htT..htNon] of TCohortData;
38      HDLC: TCohortData;
39      XPrime: array [htT..htNon] of TCohortData; //other factors precalulated
40
41      //betas from Equation 1
42      cBetaSBP,cBetaTC,cBetaHDLC : array [htT..htNon] of TCohortData;
43      BetaSBP, BetaSBPA : array [htT..htNon] of TCohortData;
44      BetaTC, BetaTCA,
45      BetaHDLC, BetaHDLCA,
46      Mean,Surv, AcuteDeath, MI, Ist,
47      //Q_Gen,
48      Q_Oth, Q_CVD,
49      Prev_CVD : TCohortData;
50
51      StartAge: integer;  //age to start calc - just a time saver...
52      NumYears: integer;  //set to 10 in ICF code.
53
54      TCBetaPFOA, TCBetaPFOS : TDistributed; // coeff for TC
55      BPBetaPFOA, BPBetaPFOS : TDistributed; // coeff for BP
56      HDLBetaPFOA, HDLBetaPFOS : TDistributed; // coeff for HDL
57      ciMI, ciIst: TDistributed;  //cost of illness values
58      ciMI65u, ciIst65u: TDistributed;  //cost of illness values 65+
59
60      procedure LoadParms;
61      procedure OutputTCCurves(DTC : double; aFilenameRoot: string);
62      procedure SetupModule; override;
63      procedure BreakdownModule; override;
64      procedure CalcBaseCalibration;
65      procedure SetProbs();
66      procedure CalcCases(const P: TPWSRecord; const OnYr, OnChangeYear: integer);
67    public
68      constructor Create(aData : PSWModelData; GlbUnc: TGlobalUncertainty;  aNeededContams: string; aName, aGroup: string; aIteration: integer);
69
70      procedure ResetBenefits(); override;
71      procedure SetupUncertainty(Unc: TUncertaintyStudy); override;
72      procedure SetupOutputs(Metrics: TCategoryMetrics); override;
73
74      procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit); override;
75    end;
76
77  implementation
78
79  { TbmPFASCVD }
80
81  procedure TbmPFASCVD.AddEntryPointBenefits(const P: TPWSRecord;
82    const Conc, FinalConc: TContamArray; const CostInfo: TCostInfo2Benefit);
83  begin
84    //always call inherited
85    inherited;
86    if not (PFOACHange or PFOSCHange) then exit;
87
88    {$IfDef DEBUG}
89    StartTimer;
90    {$endif}
91
92    if not BaseProbDone then CalcBaseCalibration;
93    var OnYr, YearsSinceChange : integer;
94    for var Yr := FData.Cfg.PFASChangeYear to FData.Cfg.EndYear do begin
95      OnYr := Yr - FData.Cfg.StartYear;
96      YearsSinceChange := Yr - FData.Cfg.PFASChangeYear;
97      {$ifdef DEBUG}
98      if _d then begin
```

```
99        FData.DebugLog.L('pfascvd',Format('Name:%s,  OnYr:%d,  YearsChange:%d',[fName, OnYr, YearsSinceChange]));
100     end;
101     {$endif}
102
103     sPFOA.GetBeforeAfterSerum(basePFOA, scenPFOA, YearsSinceChange, bcPFOA, acPFOA, FData.Cfg.DW_RSC);
104     sPFOS.GetBeforeAfterSerum(basePFOS, scenPFOS, YearsSinceChange, bcPFOS, acPFOS, FData.Cfg.DW_RSC);
105
106     SetProbs();
107     CalcCases(P, OnYr, YearsSinceChange);
108     AdvanceCohort;
109   end;
110
111   {$IfDef DEBUG}
112   TickTimer;
113   {$endif}
114 end;
115
116 procedure TbmPFASCVD.BreakdownModule;
117 begin
118   inherited;
119 end;
120
121 procedure TbmPFASCVD.CalcBaseCalibration;
122 begin
123   var x, baseprob, hazbase, lna: double;
124   var a,s,r,h: integer;
125   for a := StartAge to CMaxAge do begin
126     lna := ln(a);
127     for s := gMale to gFemale do begin
128       for r := reWhite to reOther do begin
129         BaselineProb[s,r,a] := 0;
130         for h := htT to htNon do begin
131            if HTR[h].V[s,r,a] = 0 then continue;
132           //precalc beta....
133           cBetaTC[h].V[s,r,a] := (BetaTC.V[s,r,a] + BetaTCA.V[s,r,a] * lna) ;
134           cBetaSBP[h].V[s,r,a] := (BetaSBP[h].V[s,r,a] + BetaSBPA[h].V[s,r,a] * lna);
135           cBetaHDLC[h].V[s,r,a] := (BetaHDLC.V[s,r,a] + BetaHDLCA.V[s,r,a] * lna);
136           //TODO this can be saved as well...
137           x := XPrime[h].V[s,r,a] - Mean.V[s,r,a];
138           baseprob := 1 - power(Surv.V[s,r,a], exp(cBetaTC[h].V[s,r,a] * ln(Tau[h].V[s,r,a])
139                                                   +cBetaSBP[h].V[s,r,a] * ln(SBP[h].V[s,r,a])
140                                                   +cBetaHDLC[h].V[s,r,a] * ln(HDLC.V[s,r,a])
141                                                   + x));
142           hazbase := 1 - power(1 - baseprob , 0.1);
143           HazBaseProb[h].V[s,r,a] :=  hazbase;
144           BaselineProb[s,r,a] := BaselineProb[s,r,a] + hazbase * HTR[h].V[s,r,a];;
145         end;
146       end;
147     end;
148   end;
149   BaseProbDone := True;
150 end;
151
152 procedure TbmPFASCVD.CalcCases(const P: TPWSRecord; const OnYr, OnChangeYear: integer);
153 var pa,pami, sami, paist, saist  : array[1..6] of double;
154     bpami, bsami, bpaist, bsaist  : array[1..6] of double;
155 begin
156   var a,s,r,y,i,ii: integer;
157   var d,cfb,cfo,sumpa,_dd,
158       cvdpopb, noncvdpopb, cvdpopsurvb,noncvdpopsurvb,
159       calbsurv, calbmi, calbist, calbd,
160       cvdincb, cdr, calib,
161       bd, bsurv, bmi, bist, basecvddeaths, basecvdpopdeaths,
162       calosurv, calomi, caloist, calod,
163       cvdinco,
164       od, osurv, omi, oist, optcvdpopdeaths
165       : double;
166
167   const CalYear = FData.Cfg.StartYear + OnYr;
168   for a := StartAge to CMaxAge do begin
169     for s := gMale to gFemale do begin
170       for r := reWhite to reOther do begin
171         if CohortPop[s,r,a]=0 then continue;
172         d:=0;
```

```
173         cvdpopb := CohortPop[s,r,a] * Prev_CVD.V[s,r,a];
174         noncvdpopb := CohortPop[s,r,a] * ( 1 - Prev_CVD.V[s,r,a]);
175
176         cvdpopsurvb := cvdpopb - cvdpopb * Q_Oth.V[s,r,a];
177         noncvdpopsurvb := noncvdpopb - noncvdpopb * Q_Oth.V[s,r,a];
178
179         cfB := BaselineProb[s,r,a] * noncvdpopsurvb;
180         bd := (PA_IS[1].V[s,r,a] * Ist.V[s,r,a] + PA_MI[1].V[s,r,a] * MI.V[s,r,a] + AcuteDeath.V[s,r,a]) * cfB;
181         bsurv := cFb - bd;
182         bmi := cfb * MI.V[s,r,a] - (cfb * MI.V[s,r,a] * PA_MI[1].V[s,r,a]) ;
183         bist := cfb * Ist.V[s,r,a] - (cfb * Ist.V[s,r,a] * PA_IS[1].V[s,r,a]) ;
184
185         basecvddeaths := (cvdpopsurvb + noncvdpopsurvb) * (Q_Gen.V[s,r,a] - Q_Oth.V[s,r,a]);
186         basecvdpopdeaths := (cvdpopsurvb) * (Q_Gen.V[s,r,a] - Q_Oth.V[s,r,a]);
187
188         cvdincb := (CohortPop[s,r,a] * Prev_CVD.V[s,r,min(a+1,CMaxAge)])
189                    - (CohortPop[s,r,a] * Prev_CVD.V[s,r,a])
190                    + basecvdpopdeaths + cvdpopb * Q_Oth.V[s,r,a];
191
192         calib := cvdincb / (cfb + cfb * MI.V[s,r,a] * PA_MI[1].V[s,r,a]
193                             + cfb * Ist.V[s,r,a] * PA_IS[1].V[s,r,a]
194                            );
195
196         if calib>1 then calib:=1;
197         cfB := cfB * calib;
198         calbd := bd * calib;
199         calbsurv := bsurv * calib;
200         calbmi := bmi * calib;
201         calbist := bist * calib;
202         cdr := (basecvddeaths - calbd) / cvdpopb;
203
204         //option
205         cfO := OptionProb[s,r,a] * noncvdpopsurvb;
206         od := (PA_IS[1].V[s,r,a] * Ist.V[s,r,a]  + PA_MI[1].V[s,r,a] * MI.V[s,r,a] + AcuteDeath.V[s,r,a]) * cfO;
207         osurv := cFO - od;
208         omi := cfo * MI.V[s,r,a] - (cfo * MI.V[s,r,a] * PA_MI[1].V[s,r,a]) ;
209         oist := cfo * Ist.V[s,r,a] - (cfo * Ist.V[s,r,a] * PA_IS[1].V[s,r,a]) ;
210
211         calod := od * calib;
212         calosurv := osurv * calib;
213         calomi := omi * calib;
214         caloist := oist * calib;
215
216         optcvdpopdeaths := (calod + cvdpopb * cdr ) ;
217         cvdinco := Calib *
218                    (cfo
219                     + (cfo * MI.V[s,r,a] * PA_MI[1].V[s,r,a])
220                     + (cfo * Ist.V[s,r,a] * PA_IS[1].V[s,r,a])
221                    );
222
223         //Note: Guard against negative cases.  From 7/29/22 email
224         //Note: Remove guard against negative cases.  From 9/12/22 email
225         //if (calbd - calod)<0 then continue;
226
227         //moved after check - 8/23/22
228         ICohortChange[s,r,a] := cfO;
229         //ICohortChangeBase[s,r,a] := cfB;
230
231         _i(_CVD.MI, calbmi - calomi);
232         _i(_CVD.ISt, calbist - caloist);
233         _i(_CVDC[r].MI, calbmi - calomi);
234         _i(_CVDC[r].ISt, calbist - caloist);
235         _i(_CVD.Fatal, calbd - calod);
236         _i(_CVDC[r].Fatal, calbd - calod);
237
238         for ii := 1 to length(EJProps)-1 do begin
239           _i(_CVDI[ii].MI, (calbmi - calomi) * EJProps[ii]);
240           _i(_CVDI[ii].ISt, (calbist - caloist) * EJProps[ii]);
241           _i(_CVDI[ii].Fatal, (calbd - calod) * EJProps[ii]);
242         end;
243
244         //dollars for all MI,IS
245         if a>64 then
246           d := (calbmi - calomi) * cimi.fValue + (calbist - caloist) * ciist.fValue
247         else
```

```
248           d := (calbmi - calomi) * cimi65u.fValue + (calbist - caloist) * ciist65u.fValue;
249         //discount and accumulate
250         for i := 0 to Fdata.Inp.RateCount-1 do begin
251           _dd := d / FData.Inp.PreCalcDiscRate[i,OnYr+FData.Cfg.PeriodAdjust];
252           _i(O[i].Dollars, _dd);
253           _i(OYearly[OnYr,i].Dollars, _dd);
254         end;
255
256         PAMI[1] := calomi;
257         SAMI[1]:= calomi;
258         PAIST[1] := caloist;
259         SAIST[1]:= caloist;
260         bPAMI[1] := calbmi;
261         bSAMI[1]:= calbmi;
262         bPAIST[1] := calbist;
263         bSAIST[1]:= calbist;
264         PA[1] := 0;
265         sumpa:=0;
266         for y := 2 to 6 do begin
267           PAMI[y] :=  (SAMI[y-1] - SAMI[y-1] * Q_Oth.V[s,r,a]) * PA_MI[y].V[s,r,a];
268           SAMI[y] :=  (SAMI[y-1] - SAMI[y-1] * Q_Oth.V[s,r,a]) * (1- PA_MI[y].V[s,r,a]);
269           PAIST[y] :=  (SAIST[y-1] - SAIST[y-1] * Q_Oth.V[s,r,a]) * PA_IS[y].V[s,r,a];
270           SAIST[y] :=  (SAIST[y-1] - SAIST[y-1] * Q_Oth.V[s,r,a]) * (1- PA_IS[y].V[s,r,a]);
271           bPAMI[y] :=  (bSAMI[y-1] - bSAMI[y-1] * Q_Oth.V[s,r,a]) * PA_MI[y].V[s,r,a];
272           bSAMI[y] :=  (bSAMI[y-1] - bSAMI[y-1] * Q_Oth.V[s,r,a]) * (1- PA_MI[y].V[s,r,a]);
273           bPAIST[y] :=  (bSAIST[y-1] - bSAIST[y-1] * Q_Oth.V[s,r,a]) * PA_IS[y].V[s,r,a];
274           bSAIST[y] :=  (bSAIST[y-1] - bSAIST[y-1] * Q_Oth.V[s,r,a]) * (1- PA_IS[y].V[s,r,a]);
275
276           PA[y] := bPAMI[y] - PAMI[y] + bPAIST[y] - PAIST[y];
277           sumpa := sumpa + PA[y];
278           //TODO!!!!!!!!!!!!! - shuold EJ variables be adjusted here
279           _i(_CVD.MI,-(bPAMI[y] - PAMI[y]));
280           _i(_CVD.Ist,-(bPAIST[y] - PAIST[y]));
281         end;
282
283
284         _i(_CVD.Fatal, sumpa);
285         _i(_CVD.PostAcute, sumpa);
286         _i(_CVDC[r].Fatal, sumpa);
287         for ii := 1 to length(EJProps)-1 do begin
288           _i(_CVDI[ii].PostAcute, sumpa * EJProps[ii]);
289           _i(_CVDI[ii].Fatal, sumpa * EJProps[ii]);
290         end;
291
292         //valuation
293         for i := 0 to Fdata.Inp.RateCount-1 do begin
294           d := ((calbd - calod) * FGlb.VSLY[CalYear]) / FData.Inp.PreCalcDiscRate[i,OnYr+FData.Cfg.PeriodAdjust];
295           _i(O[i].Dollars,d);
296           _i(OYearly[OnYr,i].Dollars,d);
297           d := 0;
298           for y := 2 to 6 do begin
299             d := d + PA[y] * FGlb.VSLY[CalYear] / FData.Inp.PreCalcDiscRate[i,y-1+FData.Cfg.PeriodAdjust];
300             if CalYear+y-1 <= FData.Cfg.EndYear then
301               _i(OYearly[OnYr+y-1,i].Dollars,PA[y] * FGlb.VSLY[CalYear] / FData.Inp.PreCalcDiscRate[i,OnYr+y-1+FData.Cfg.PeriodAdjust]);
302           end;
303           _i(O[i].Dollars,  d / FData.Inp.PreCalcDiscRate[i,OnYr+FData.Cfg.PeriodAdjust]);
304         end;
305
306
307       {$ifdef DEBUG}
308       if _d then begin
309         var aage: string;
310         if (OnYr)>a then
311           aage:='A'+inttostr(OnYr-a)
312         else
313           aage := inttostr(A-OnYr);
314
315         var chs := s.ToString+'_'+r.ToString+'_'+aage;
316         FData.DebugLog.L('pfascvd_icf',Format('%d,%s,%g,%g,%g,%g,'+
317                           '"%s",%d,%d,%d,%g,'+
318                           '%g,%g,%g,%g,'+
319                           '%g,%g,%g,%g,%g,'+
320                           '%g,%g,%g,%g,%g,%g,%g',
321                          [OnChangeYear,P.PWSId,bcPFOA, acPFOA,bcPFOS,acPFOS,
```

```
                          chs,s,r,a,CohortPop[s,r,a],
                          scenPFOA[s,r,a],basePFOA[s,r,a],scenPFOS[s,r,a],basePFOS[s,r,a],
                          (scenPFOA[s,r,a] - basePFOA[s,r,a]),
                          (scenPFOS[s,r,a] - basePFOS[s,r,a]),
                          DeltaTC.V[s,r,a],DeltaSBP.V[s,r,a],DeltaHDLC.V[s,r,a],
                          calbmi - calomi, calbist - caloist, calbd - calod + sumpa, sumpa,
                          BaselineProb[s,r,a],OptionProb[s,r,a], calib
                          ]));
        end;
       {$endif}


       end;
     end;
   end;

end;

constructor TbmPFASCVD.Create(aData: PSWModelData; GlbUnc: TGlobalUncertainty;
  aNeededContams, aName, aGroup: string; aIteration: integer);
begin
  inherited Create(aData,GlbUnc,aNeededContams, aName, aGroup,aIteration);
  //overriding to allow qc file output on initial setup;
  if (aIteration<0) and (aData.OutPath<>'') then begin
    LoadParms;
    OutputTCCurves(-10, aData.OutPath);
  end;
end;

procedure TbmPFASCVD.LoadParms;
begin

  for var i := 1 to 6 do begin
    PA_MI[i].LoadFromFile(DataPath+'CohortData\CVD\postacute_MI.csv',2+i);
    PA_IS[i].LoadFromFile(DataPath+'CohortData\CVD\postacute_IS.csv', 2+i);
  end;
  for var h := htT to htNon do begin
    HTR[h].LoadFromFile(DataPath+'CohortData\CVD\prob_ht_treatment.csv',3+h);
    Tau[h].LoadFromFile(DataPath+'CohortData\CVD\ln_tc_ht.csv',3+h, true);
    SBP[h].LoadFromFile(DataPath+'CohortData\CVD\ln_sbp_ht.csv',3+h, true);
    XPrime[h].LoadFromFile(DataPath+'CohortData\CVD\xprime_beta.csv',3+h);
  end;
  BetaSBP[htT].LoadFromFile(DataPath+'CohortData\CVD\beta.csv',9);
  BetaSBPA[htT].LoadFromFile(DataPath+'CohortData\CVD\beta.csv',10);
  BetaSBP[htU].LoadFromFile(DataPath+'CohortData\CVD\beta.csv',11);
  BetaSBPA[htU].LoadFromFile(DataPath+'CohortData\CVD\beta.csv',12);
  BetaSBP[htNon].LoadFromFile(DataPath+'CohortData\CVD\beta.csv',11);
  BetaSBPA[htNon].LoadFromFile(DataPath+'CohortData\CVD\beta.csv',12);

  HDLC.LoadFromFile(DataPath+'CohortData\CVD\ln_hdlc_ht.csv',3,true);

  BetaTC.LoadFromFile(DataPath+'CohortData\CVD\beta.csv',4);
  BetaTCA.LoadFromFile(DataPath+'CohortData\CVD\beta.csv',5);
  BetaHDLC.LoadFromFile(DataPath+'CohortData\CVD\beta.csv',7);
  BetaHDLCA.LoadFromFile(DataPath+'CohortData\CVD\beta.csv',8);

  Mean.LoadFromFile(DataPath+'CohortData\CVD\beta.csv',6);
  Surv.LoadFromFile(DataPath+'CohortData\CVD\S_Sr.csv');

  MI.LoadFromFile(DataPath+'CohortData\CVD\first_hard_proportions.csv',3);
  Ist.LoadFromFile(DataPath+'CohortData\CVD\first_hard_proportions.csv',4);
  AcuteDeath.LoadFromFile(DataPath+'CohortData\CVD\first_hard_proportions.csv',5);
// Now loaded in base class
//  Q_Gen.LoadFromFile(DataPath+'CohortData\CVD\q_cvd.csv',3);
  Q_Oth.LoadFromFile(DataPath+'CohortData\CVD\q_cvd.csv',4);
  Prev_CVD.LoadFromFile(DataPath+'CohortData\CVD\prevalence_cvd.csv');

  for var a := 40 to CMaxAge do begin
    for var s := gMale to gFemale do begin
      for var  r := reWhite to reOther do begin
        Q_CVD.V[s,r,a] := Q_Gen.V[s,r,a] - Q_Oth.V[s,r,a];
      end;
    end;
  end;
end;
```

```
procedure TbmPFASCVD.OutputTCCurves(DTC: double; aFilenameRoot: string);
var Curves: array [htT..htNon] of TCohortData; //other factors precalulated
    AdjFactors : TCohortData;
begin
exit;
  var TSW := TStreamWriter.Create(aFileNameRoot+'Probs10PointTCChange.csv');
  TSW.WriteLine('ht,a,s,r,lna,betatc,betatca,xprime,mean,prob,baseprob,htr,surv,tc,dtc,diff');
  var TSC := TStreamWriter.Create(aFileNameRoot+'ProbDiff10PointTCChange_Curves.csv');
  var TSA := TStreamWriter.Create(aFileNameRoot+'ASCVDAdjustmentFactors.csv');

  var x, prob, baseprob, lna, b, ascvd : double;
  var a,s,r,h: integer;
  var ol : string;
  for a := 40 to CMaxAge do begin
    lna := ln(a);
    for s := gMale to gFemale do begin
      for r := reWhite to reOther do begin
        b := (BetaTC.V[s,r,a] + BetaTCA.V[s,r,a] * lna);
        ascvd:=0;
        for h := htT to htNon do begin
          x := XPrime[h].V[s,r,a] - Mean.V[s,r,a];
          prob := 1 - power(Surv.V[s,r,a], exp(b * ln(Tau[h].V[s,r,a] + DTC) + x));
          baseprob := 1 - power(Surv.V[s,r,a], exp(b * ln(Tau[h].V[s,r,a]) + x));
          ascvd := ascvd + (1 - power(1 - baseprob , 0.1)) * HTR[h].V[s,r,a];
          var diff := baseprob-prob;
          Curves[h].V[s,r,a] := diff;
          ol := format('%d, %d,%d,%d,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g',
              [
              h,a,s,r,
              lna, BetaTC.V[s,r,a], BetaTCA.V[s,r,a], XPrime[h].V[s,r,a],
              Mean.V[s,r,a], prob, baseprob,
              HTR[h].V[s,r,a], Surv.V[s,r,a], Tau[h].V[s,r,a], DTC, diff
              ]);
           TSW.WriteLine(ol);
        end;
        AdjFactors.V[s,r,a] :=  (ascvd * AcuteDeath.V[s,r,a]) / ( Q_CVD.V[s,r,a] * (1 - Prev_CVD.V[s,r,a]));
      end;
    end;
  end;

  ol:='age';
  for h := htT to htNon do begin
    for s := gMale to gFemale do begin
      for r := reWhite to reOther do begin
         if h=0 then ol:=ol+',htt_' else
         if h=1 then ol:=ol+',htu_' else
           ol:=ol+',nht_';
         if s=0 then ol:=ol+'m_' else ol:=ol+'f_';
         if r=0 then ol:=ol+'w' else
         if r=1 then ol:=ol+'b' else
         if r=2 then ol:=ol+'h' else
           ol:=ol+'o'
      end;
    end;
  end;
  TSC.WriteLine(ol);

  ol:='age';
  for s := gMale to gFemale do begin
    for r := reWhite to reOther do begin
       if s=0 then ol:=ol+',m_' else ol:=ol+',f_';
       if r=0 then ol:=ol+'w' else
       if r=1 then ol:=ol+'b' else
       if r=2 then ol:=ol+'h' else
         ol:=ol+'o'
    end;
  end;
  TSA.WriteLine(ol);

  for a := 40 to CMaxAge do begin
    ol:=inttostr(a);
    for h := htT to htNon do begin
      for s := gMale to gFemale do begin
```

```
471         for r := reWhite to reOther do begin
472            ol := ol + ','+floattostr(Curves[h].V[s,r,a]);
473         end;
474       end;
475     end;
476     TSC.WriteLine(ol);
477   end;
478
479   for a := 40 to CMaxAge do begin
480     ol:=inttostr(a);
481     for s := gMale to gFemale do begin
482       for r := reWhite to reOther do begin
483          ol := ol + ','+floattostr(AdjFactors.V[s,r,a]);
484       end;
485     end;
486     TSA.WriteLine(ol);
487   end;
488
489   TSA.Free;
490   TSC.Free;
491   TSW.Free;
492 end;
493
494 procedure TbmPFASCVD.ResetBenefits;
495 begin
496   inherited;
497   _CVD := Default(TbmCVDOutputs);
498   for var r  := reWhite  to reOther do
499     _CVDC[r] := Default(TbmCVDOutputs);
500   for var r  := 0 to length(_CVDI) - 1 do
501     _CVDI[r] := Default(TbmCVDOutputs);
502 end;
503
504 procedure TbmPFASCVD.SetProbs();
505 begin
506   var x, prob, haz, dsbp, scA, scS:  double;
507   var a,s,r,h: integer;
508   {$ifdef DEBUG}
509   if _d then
510     FData.DebugLog.L('pfascvd','DeltaProb CVD');
511   {$endif}
512   for a := StartAge to CMaxAge do begin
513     for s := gMale to gFemale do begin
514       for r := reWhite to reOther do begin
515         OptionProb[s,r,a] := 0;
516         scA := scenPFOA[s,r,a] - basePFOA[s,r,a];
517         scS := scenPFOS[s,r,a] - basePFOS[s,r,a];
518         DeltaTC.V[s,r,a] := TCBetaPFOA.fValue * scA  + TCBetaPFOS.fValue * scS;
519         DeltaHDLC.V[s,r,a] := HDLBetaPFOA.fValue * scA  + HDLBetaPFOS.fValue * scS;
520         DeltaSBP.V[s,r,a] := BPBetaPFOA.fValue * scA  + BPBetaPFOS.fValue * scS;
521
522         for h := htT to htNon do begin
523           if HTR[h].V[s,r,a] = 0 then continue;
524           x := XPrime[h].V[s,r,a] - Mean.V[s,r,a];
525           dsbp := DeltaSBP.V[s,r,a];
526           if h=htT then dsbp:=0;
527           prob := 1 - power(Surv.V[s,r,a], exp(
528                                                cBetaTC[h].V[s,r,a] * ln(Tau[h].V[s,r,a] + DeltaTC.V[s,r,a])
529                                               +cBetaSBP[h].V[s,r,a] * ln(SBP[h].V[s,r,a] + dsbp)
530                                               +cBetaHDLC[h].V[s,r,a] * ln(HDLC.V[s,r,a] + DeltaHDLC.V[s,r,a])
531                                               + x));
532           haz := 1 - power(1 - prob , 0.1);
533           //store annual probability
534           OptionProb[s,r,a] := OptionProb[s,r,a] + (haz) * HTR[h].V[s,r,a];
535         end;
536
537         {$ifdef DEBUG}
538         if _d then
539           FData.DebugLog.L('pfascvd',Format('a:%d, r:%d, s:%d, dProbCVD:%g',[a,r,s,optionProb[s,r,a]]));
540         {$endif}
541       end;
542     end;
543   end;
544 end;
545
```

```
procedure TbmPFASCVD.SetupModule;
begin
  inherited;
  StartAge := 40;
  NumYears := 10;
  //TODO guard against not finding calculators...
  sPFOS := FData.SerumCalc.GetSerumCalc('PFOS');
  sPFOA := FData.SerumCalc.GetSerumCalc('PFOA');
  LoadParms;
  BaseprobDone:=false;
end;

procedure TbmPFASCVD.SetupOutputs(Metrics: TCategoryMetrics);
begin
  inherited;
  Metrics.AddMetric(fName+'.Fatal',
                    'PFASCVD Fatal Cases', @_CVD.Fatal, mtBenefitCases,mcSumOnly, False);
  Metrics.AddMetric(fName+'.MI',
                    'PFASCVD MI Cases', @_CVD.MI, mtBenefitCases,mcSumOnly, False);
  Metrics.AddMetric(fName+'.ISt',
                    'PFASCVD ISt Cases', @_CVD.Ist, mtBenefitCases,mcSumOnly, False);
  Metrics.AddMetric(fName+'.PostAcute',
                    'PFASCVD Post Acute Deaths', @_CVD.PostAcute, mtBenefitCases,mcSumOnly, False);

  setlength(_CVDI,length(fData.EJProps.Names));
  if FData.Cfg.PFASEJOutput then begin
    for var i := 1 to length(fData.EJProps.Names) - 1 do begin
      var s := fData.EJProps.Names[i];
      Metrics.AddMetric(fName+'.Fatal'+s,
                        'PFASCVD Fatal Cases '+s, @_CVDI[i].Fatal, mtEJOutput,mcSumOnly, False, '', @EJInd);
      Metrics.AddMetric(fName+'.MI'+s,
                        'PFASCVD MI Cases '+s, @_CVDI[i].MI, mtEJOutput,mcSumOnly, False, '', @EJInd);
      Metrics.AddMetric(fName+'.ISt'+s,
                        'PFASCVD ISt Cases '+s, @_CVDI[i].Ist, mtEJOutput,mcSumOnly, False, '', @EJInd);
      Metrics.AddMetric(fName+'.PostAcute'+s,
                        'PFASCVD Post Acute Deaths '+s, @_CVDI[i].PostAcute, mtEJOutput, mcSumOnly, False, '', @EJInd);
    end;

    Metrics.AddMetric(fName+'.FatalWhite',
                        'PFASCVD Fatal Cases White', @_CVDC[reWhite].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.FatalBlack',
                        'PFASCVD Fatal Cases Black', @_CVDC[reBlack].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.FatalOther',
                        'PFASCVD Fatal Cases Other', @_CVDC[reOther].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.FatalHispanic',
                        'PFASCVD Fatal Cases Hispanic', @_CVDC[reHispanic].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);

    Metrics.AddMetric(fName+'.MIWhite',
                        'PFASCVD MI Cases White', @_CVDC[reWhite].MI, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.MIBlack',
                        'PFASCVD MI Cases Black', @_CVDC[reBlack].MI, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.MIOther',
                        'PFASCVD MI Cases Other', @_CVDC[reOther].MI, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.MIHispanic',
                        'PFASCVD MI Cases Hispanic', @_CVDC[reHIspanic].MI, mtEJOutput,mcSumOnly,False, '', @EJInd);

    Metrics.AddMetric(fName+'.IStlWhite',
                        'PFASCVD ISt Cases White', @_CVDC[reWhite].ISt, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.IStBlack',
                        'PFASCVD ISt Cases Black', @_CVDC[reBlack].ISt, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.IStOther',
                        'PFASCVD ISt Cases Other', @_CVDC[reOther].ISt, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.IStHispanic',
                        'PFASCVD ISt Cases Hispanic', @_CVDC[reHispanic].ISt, mtEJOutput,mcSumOnly,False, '', @EJInd);
  end;

end;

procedure TbmPFASCVD.SetupUncertainty(Unc: TUncertaintyStudy);
begin
  inherited;
//setup VCV relationships....
//  Unc.AddVCVMatrix('PFASCVD','TCBetaPFOA,TCBetaPFOS');

```

```
  var censor:double := -9e99;
  if FData.Cfg.Censor then censor:=0;

  TCBetaPFOA := Unc.AddInVar(nil, dNormal, 1.57, 0, 0, censor, 1.57, 0.79, 0,
    'TCBetaPFOA', 'TC Beta for PFOA');
  TCBetaPFOS := Unc.AddInVar(nil, dNormal, 0.08, 0, 0, censor, 0.08, 0.04, 0,
    'TCBetaPFOS', 'TC Beta for PFOS');
  BPBetaPFOA := Unc.AddInVar(nil, dNone, 0, 0, 0, censor, 0, 0, 0,
    'BPBetaPFOA', 'BP Beta for PFOA');
  BPBetaPFOS := Unc.AddInVar(nil, dNormal, 0.044, 0, 0, censor, 0.044, 0.02, 0,
    'BPBetaPFOS', 'BP Beta for PFOS');

//  HDLBetaPFOA := Unc.AddInVar(nil, dNormal, 0.11, 0, 0, censor, 0.11, 0.17, 0,
//    'HDLBetaPFOA', 'HDL Beta for PFOA');
//  HDLBetaPFOS := Unc.AddInVar(nil, dNormal, 0.05, 0, 0, censor, 0.05, 0.03, 0,
//    'HDLBetaPFOS', 'HDL Beta for PFOS');

  //Note:  Set HDL coeffs to 0 - 9/19/22
  HDLBetaPFOA := Unc.AddInVar(nil, dNone, 0, 0, 0, censor, 0, 0, 0,
    'HDLBetaPFOA', 'HDL Beta for PFOA');
  HDLBetaPFOS := Unc.AddInVar(nil, dNone, 0, 0, 0, censor, 0, 0, 0,
    'HDLBetaPFOS', 'HDL Beta for PFOS');


  if round(FData.Cfg.DiscountRate*100) = 7 then begin
    ciMI := Unc.AddInVar(nil, dNone, 98828, 0, 0, -9E99, 100028, 0, 0,
      'ciMI', 'Cost Illness MI 40-64');
    ciMI65u := Unc.AddInVar(nil, dNone, 87128, 0, 0, -9E99, 87919, 0, 0,
      'ciMI65Up', 'Cost Illness MI 65Up');
    ciIst := Unc.AddInVar(nil, dNone, 27533, 0, 0, -9E99, 100028, 0, 0,
      'ciISt', 'Cost Illness IS 40-64');
    ciISt65u := Unc.AddInVar(nil, dNone, 25393, 0, 0, -9E99, 87919, 0, 0,
      'ciISt65Up', 'Cost Illness IS 65Up');
  end else begin
    ciMI := Unc.AddInVar(nil, dNone, 100028, 0, 0, -9E99, 100028, 0, 0,
      'ciMI', 'Cost Illness MI 40-64');
    ciMI65u := Unc.AddInVar(nil, dNone, 87919, 0, 0, -9E99, 87919, 0, 0,
      'ciMI65Up', 'Cost Illness MI 65Up');
    ciIst := Unc.AddInVar(nil, dNone, 27663, 0, 0, -9E99, 100028, 0, 0,
      'ciISt', 'Cost Illness IS 40-64');
    ciISt65u := Unc.AddInVar(nil, dNone, 25471, 0, 0, -9E99, 87919, 0, 0,
      'ciISt65Up', 'Cost Illness IS 65Up');
  end;

end;

end.

```

code\BenefitModules\bmPFASDBP.pas

```
unit bmPFASDBP;
interface

uses SysUtils, Classes, Math, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
     SafeWaterOccurrence, SafeWaterUtility, SafewaterMetrics, SafeWaterGlobals,
     bmPFASBase ;

type

  TbmDBPOutputs=record
    Fatal, Nonfatal: double;
  end;
  TbmDBPDollarOutputs=record
    WTPDollars: double;
  end;

  TbmPFASDBP = class(TbmPFASBase)
  protected
    _BinFatalS,_BinNonFatalS: array[1..5] of double;
    _BinFatalG,_BinNonFatalG: array[1..5] of double;
    _BinDollarsWTPS,_BinDollarsS: array of array[1..5] of double;
    _BinDollarsWTPG,_BinDollarsG: array of array[1..5] of double;
```

```
23
24      //yearly at 0% - Year,Bin
25      _BinDollarsWTPS0,_BinDollarsS0: array of array[1..5] of double;
26      _BinDollarsWTPG0,_BinDollarsG0: array of array[1..5] of double;
27
28      _DBP : TbmDBPOutputs;
29      _DBPD : array of TbmDBPDollarOutputs;
30
31      procedure SetupModule; override;
32    public
33
34      procedure ResetBenefits(); override;
35      procedure SetupOutputs(Metrics: TCategoryMetrics); override;
36      procedure CollectPWSWideEPValues(const P: TPWSRecord); override;
37      procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit); override;
38    end;
39
40
41  implementation
42
43  { TbmPFASDBP }
44
45  procedure TbmPFASDBP.AddEntryPointBenefits(const P: TPWSRecord; const Conc,
46    FinalConc: TContamArray; const CostInfo: TCostInfo2Benefit);
47  begin
48    inherited;
49    if CostInfo.GACTOCBin<1 then exit;
50    var i, yr: integer;
51    var v: double;
52    const AllYrs = FData.Cfg.EndYear - FData.Cfg.StartYear + 1;
53    if P.SourceWater = swGroundWater then begin
54      _i(_DBP.Fatal,_BinFatalG[CostInfo.GACTOCBin] * CurPop);
55      _i(_DBP.NonFatal,_BinNonFatalG[CostInfo.GACTOCBin] * CurPop);
56      for yr := 0 to AllYrs-1 do begin
57        for i := 0 to FData.Inp.RateCount-1 do begin
58         v := _BinDollarsG0[yr][CostInfo.GACTOCBin] * CurPop / FData.Inp.PreCalcDiscRate[i,Yr+FData.Cfg.PeriodAdjust];
59         _i(O[i].Dollars, v);
60         _i(OYearly[Yr,i].Dollars,  v);
61         _i(_DBPD[i].WTPDollars, _BinDollarsWTPG0[yr][CostInfo.GACTOCBin] * CurPop / FData.Inp.PreCalcDiscRate[i,Yr+FData.Cfg.PeriodAdjust]);
62        end;
63      end;
64    end else begin
65      _i(_DBP.Fatal,_BinFatalS[CostInfo.GACTOCBin] * CurPop);
66      _i(_DBP.NonFatal,_BinNonFatalS[CostInfo.GACTOCBin] * CurPop);
67      for yr := 0 to AllYrs-1 do begin
68        for i := 0 to FData.Inp.RateCount-1 do begin
69         v := _BinDollarsS0[yr][CostInfo.GACTOCBin] * CurPop / FData.Inp.PreCalcDiscRate[i,Yr+FData.Cfg.PeriodAdjust];
70         _i(O[i].Dollars, v);
71         _i(OYearly[Yr,i].Dollars,  v);
72         _i(_DBPD[i].WTPDollars, _BinDollarsWTPS0[yr][CostInfo.GACTOCBin] * CurPop / FData.Inp.PreCalcDiscRate[i,Yr+FData.Cfg.PeriodAdjust]);
73        end;
74      end;
75    end;
76  end;
77
78  procedure TbmPFASDBP.CollectPWSWideEPValues(const P: TPWSRecord);
79  begin
80    inherited;
81    //annualizing
82    for var i := 0 to FData.Inp.RateCount-1 do begin
83      _DBPD[i].WTPDollars := _DBPD[i].WTPDollars *  FData.Inp.PreCalcAnnRate[i, FData.Cfg.EndYear - FData.Cfg.StartYear + 1];
84    end;
85  end;
86
87
88  procedure TbmPFASDBP.ResetBenefits;
89  begin
90    inherited;
91    _DBP := Default(TbmDBPOutputs);
92    for var i := 0 to FData.Inp.RateCount-1 do begin
93      _DBPD[i] := Default(TbmDBPDollarOutputs);
94    end;
95  end;
96
97  procedure TbmPFASDBP.SetupModule;
```

```
98  begin
99    inherited;
100
101   fillchar(_binFatalG,sizeof(_binFatalG),0);
102   fillchar(_binNonFatalG,sizeof(_binNonFatalG),0);
103   fillchar(_binFatalS,sizeof(_binFatalS),0);
104   fillchar(_binNonFatalS,sizeof(_binNonFatalS),0);
105
106   setlength(_DBPD,FData.Inp.RateCount);
107   setlength(_binDollarsG,FData.Inp.RateCount);
108   setlength(_binDollarsWTPG,FData.Inp.RateCount);
109   setlength(_binDollarsS,FData.Inp.RateCount);
110   setlength(_binDollarsWTPS,FData.Inp.RateCount);
111   for var i := 0 to FData.Inp.RateCount-1 do begin
112     fillchar(_binDollarsG[i],sizeof(_binDollarsG[i]),0);
113     fillchar(_binDollarsWTPG[i],sizeof(_binDollarsWTPG[i]),0);
114     fillchar(_binDollarsS[i],sizeof(_binDollarsS[i]),0);
115     fillchar(_binDollarsWTPS[i],sizeof(_binDollarsWTPS[i]),0);
116   end;
117
118   //new - use undiscounted as the base
119   const FYears = Fdata.Cfg.EndYear - FData.Cfg.StartYear + 1;
120   setlength(_binDollarsG0,fYears);
121   setlength(_binDollarsWTPG0,fYears);
122   setlength(_binDollarsS0,fYears);
123   setlength(_binDollarsWTPS0,fYears);
124   for var i := 0 to FYears - 1 do begin
125     fillchar(_binDollarsG0[i],sizeof(_binDollarsG0[i]),0);
126     fillchar(_binDollarsWTPG0[i],sizeof(_binDollarsWTPG0[i]),0);
127     fillchar(_binDollarsS0[i],sizeof(_binDollarsS0[i]),0);
128     fillchar(_binDollarsWTPS0[i],sizeof(_binDollarsWTPS0[i]),0);
129   end;
130
131   if FIteration<0 then exit;
132
133   //Note:  Hardcoding here to accomodate inclusion of ICF DBP analysis.
134   //can be generalized if needed.
135   var ILine := TStringLIst.Create;
136   var aFile := TStreamReader.Create(DataPath+'CohortData\DBP\benefits_uncertainty-Total cancer-related deaths avoided.csv');
137
138   //use this to find the correct column for iteration -  4 columns before ierations start
139   //current set has 4000 so circle to beginning of more than 4000
140   var getCol:integer := 4 + (FIteration MOD 4000);
141   ILine.CommaText := aFile.ReadLine;
142   var binCol := ILine.IndexOf('TOCbin');
143   var wCol := ILine.IndexOf('Source');
144   var bint:string;
145   var bin: integer;
146
147   while not aFile.EndOfStream do begin
148     ILine.CommaText := aFile.ReadLine;
149     bint := ILine[binCol];
150     //text in the form popGACTOCBinN
151     bin := strtoint(copy(bint,13,1));
152     if ILine[wCol] = 'Ground' then
153       _binFatalG[bin] := strtofloat(ILine[getCol])
154     else
155       _binFatalS[bin] := strtofloat(ILine[getCol]);
156   end;
157   aFile.Free;
158
159   aFile := TStreamReader.Create(DataPath+'CohortData\DBP\benefits_uncertainty-Total non-fatal cancer cases avoided.csv');
160   ILine.CommaText := aFile.ReadLine;
161   while not aFile.EndOfStream do begin
162     ILine.CommaText := aFile.ReadLine;
163     bint := ILine[binCol];
164     //text in the form popGACTOCBinN
165     bin := strtoint(copy(bint,13,1));
166     if ILine[wCol] = 'Ground' then
167       _binNonFatalG[bin] := strtofloat(ILine[getCol])
168     else
169       _binNonFatalS[bin] := strtofloat(ILine[getCol]);
170   end;
171   aFile.Free;
```

```
172
173   (*
174   var s := 'COI-based';
175   if fdata.Cfg.UseWTP then
176     s := 'WTP-Based';
177
178   for var i := 0 to FData.Inp.RateCount-1 do begin
179     var DF := 'not set';
180     if FData.Inp.Rates[i] = 0  then
181       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 0%).csv'
182     else
183     if FData.Inp.Rates[i] = 2  then
184       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 2%).csv'
185     else
186     if FData.Inp.Rates[i] = 3  then
187       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 3%).csv'
188     else
189     if FData.Inp.Rates[i] = 7  then
190       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 7%).csv';
191     aFile := TStreamReader.Create(DF);
192     ILine.CommaText := aFile.ReadLine;
193     while not aFile.EndOfStream do begin
194       ILine.CommaText := aFile.ReadLine;
195       bint := ILine[binCol];
196       //text in the form popGACTOCBinN
197       bin := strtoint(copy(bint,13,1));
198       if ILine[wCol] = 'Ground' then
199         _binDollarsG[i][bin] := strtofloat(ILine[getCol])
200       else
201         _binDollarsS[i][bin] := strtofloat(ILine[getCol]);
202     end;
203     aFile.Free;
204   end;
205
206   //load WTP values always...
207   s := 'WTP-Based';
208   for var i := 0 to FData.Inp.RateCount-1 do begin
209     var DF := 'not set - wtp';
210     if FData.Inp.Rates[i] = 0  then
211       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 0%).csv'
212     else
213     if FData.Inp.Rates[i] = 2  then
214       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 2%).csv'
215     else
216     if FData.Inp.Rates[i] = 3  then
217       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 3%).csv'
218     else
219     if FData.Inp.Rates[i] = 7  then
220       DF := DataPath+'CohortData\DBP\benefits_uncertainty-Annualized '+s+' benefits (dollars $2022, 7%).csv';
221
222     aFile := TStreamReader.Create(DF);
223     ILine.CommaText := aFile.ReadLine;
224     while not aFile.EndOfStream do begin
225       ILine.CommaText := aFile.ReadLine;
226       bint := ILine[binCol];
227       //text in the form popGACTOCBinN
228       bin := strtoint(copy(bint,13,1));
229       if ILine[wCol] = 'Ground' then
230         _binDollarsWTPG[i][bin] := strtofloat(ILine[getCol])
231       else
232         _binDollarsWTPS[i][bin] := strtofloat(ILine[getCol]);
233     end;
234     aFile.Free;
235   end;
236   *)
237
238   //use this to find the correct column for iteration -  5 columns before ierations start
239   //current set has 4000 so circle to beginning of more than 4000
240   getCol := 5 + (FIteration MOD 4000);
241   var s := 'COI-based';
242   if fdata.Cfg.UseWTP then
243     s := 'WTP-Based';
244   var DF := DataPath+'CohortData\DBP\benefits_annual_uncertainty-Present '+s+' value of benefits (dollars $2022, 0%).csv';
245   aFile := TStreamReader.Create(DF);
246   ILine.CommaText := aFile.ReadLine;
```

```
  while not aFile.EndOfStream do begin
    ILine.CommaText := aFile.ReadLine;
    bint := ILine[binCol];
    //text in the form popGACTOCBinN
    bin := strtoint(copy(bint,13,1));
    var yr := strtoint(ILine[4]);
    if ILine[wCol] = 'Ground' then
      _binDollarsG0[yr -  FData.Cfg.StartYear][bin] := strtofloat(ILine[getCol])
    else
      _binDollarsS0[yr -  FData.Cfg.StartYear][bin] := strtofloat(ILine[getCol]);
  end;
  aFile.Free;

  s := 'WTP-Based';
  DF := DataPath+'CohortData\DBP\benefits_annual_uncertainty-Present '+s+' value of benefits (dollars $2022, 0%).csv';
  aFile := TStreamReader.Create(DF);
  ILine.CommaText := aFile.ReadLine;
  while not aFile.EndOfStream do begin
    ILine.CommaText := aFile.ReadLine;
    bint := ILine[binCol];
    //text in the form popGACTOCBinN
    bin := strtoint(copy(bint,13,1));
    var yr := strtoint(ILine[4]);
    if ILine[wCol] = 'Ground' then
      _binDollarsWTPG0[yr -  FData.Cfg.StartYear][bin] := strtofloat(ILine[getCol])
    else
      _binDollarsWTPS0[yr -  FData.Cfg.StartYear][bin] := strtofloat(ILine[getCol]);
  end;
  aFile.Free;
  ILine.Free;
  //PostAnnualize:=False;
end;

procedure TbmPFASDBP.SetupOutputs(Metrics: TCategoryMetrics);
begin
  inherited;
  Metrics.AddMetric(fName+'.Fatal',
                    'PFASDBP Bladder Cancer Fatal Cases', @_DBP.Fatal, mtBenefitCases,mcSumOnly, False);
  Metrics.AddMetric(fName+'.NonFatal',
                    'PFASDBP Bladder Cancer Non Fatal Cases', @_DBP.NonFatal, mtBenefitCases,mcSumOnly, False);
  for var i := 0 to FData.Inp.RateCount-1 do begin
    var tg := floattostr(FData.Inp.Rates[i]);
    Metrics.AddMetric(fName+'.Dollars WTP',
                      'PFASDBP Bladder Cancer Dollars WTP', @_DBPD[i].WTPDollars, mtBenefitDollars,mcSumOnly, False, tg);
  end;
end;

end.

```

code\BenefitModules\bmPFASKidneyC.pas

```
unit bmPFASKidneyC;

interface

uses SysUtils, Classes, Math, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
     SafeWaterUtility, SafeWaterUncertBucket, SafewaterMetrics, SafeWaterGlobals,
     SafeWaterCalcDist, PFASSerumCalc, SafewaterCohortArray,
     bmPFASBase, bmPFASAbstractCancSC ;


type

  TbmPFASKidneyC = class(TbmAbstractCancSC)
  protected
    procedure LoadParms; override;
  public
    procedure SetupUncertainty(Unc: TUncertaintyStudy); override;
  end;

implementation

```

```
22 { TbmPFASKidneyC }
23
24
25 procedure TbmPFASKidneyC.LoadParms;
26 begin
27   SumProbSurvAge.LoadFromFile(DataPath+'CohortData\KidneyCancer\cumsum_p_cancer_survival_through_age.csv');
28   ProbSurvAge.LoadFromFile(DataPath+'CohortData\KidneyCancer\p_cancer_survival_through_age.csv');
29   GenProb.LoadFromFile(DataPath+'CohortData\KidneyCancer\gen_prob_mortality.csv');
30   MortShare.LoadFromFile(DataPath+'CohortData\KidneyCancer\kidney_cancer_mortality_share.csv');
31   //loading files for pre-calcs...
32   dist.LoadFromFile(DataPath+'CohortData\KidneyCancer\distant_cancer.csv');
33   reg.LoadFromFile(DataPath+'CohortData\KidneyCancer\regional_cancer.csv');
34   local.LoadFromFile(DataPath+'CohortData\KidneyCancer\localized_cancer.csv');
35   uns.LoadFromFile(DataPath+'CohortData\KidneyCancer\unstaged_cancer.csv');
36   for var y := 0 to 10 do begin
37     ps_dist[y].LoadFromFile(DataPath+'CohortData\KidneyCancer\prob_survival_distant_0_11.csv',3+y);
38     ps_reg[y].LoadFromFile(DataPath+'CohortData\KidneyCancer\prob_survival_regional_0_11.csv',3+y);
39     ps_local[y].LoadFromFile(DataPath+'CohortData\KidneyCancer\prob_survival_localized_0_11.csv',3+y);
40     ps_uns[y].LoadFromFile(DataPath+'CohortData\KidneyCancer\prob_survival_unstaged_0_11.csv',3+y);
41   end;
42 end;
43
44
45
46 procedure TbmPFASKidneyC.SetupUncertainty(Unc: TUncertaintyStudy);
47 begin
48   inherited;
49   //TODO 0.0147 is ICF lifeprob - document this here...
50   Slope := Unc.AddInVar(nil, dNormal, 0.00178/0.014702402, 0, 0, -9E99, 0.00178/0.014702402, 0.00089/0.014702402, 0,
51    'PFOAKidneyCancerSlope', 'Kidney Cancer Slope PFOA');
52
53   //TODO separate rates and make RCC_Share uncertaion
54   C_Share := Unc.AddInVar(nil, dNone, 0.9, 0, 0, -9E99, 0.9, 0, 0,
55     'KidneyCancerC_Share', 'Kidney Cancer RCC Share');
56
57   COI_Year_1 := Unc.AddInVar(nil, dNone, 239828, 0, 0, -9E99, 239828, 0, 0,
58     'KidneyCancerCOI1', 'Kidney Cancer COI Year 1');
59   COI_Year_2 := Unc.AddInVar(nil, dNone, 182464, 0, 0, -9E99, 182464, 0, 0,
60     'KidneyCancerCOI2', 'Kidney Cancer COI Year 2');
61   COI_Year_3 := Unc.AddInVar(nil, dNone, 1525, 0, 0, -9E99, 1525, 0, 0,
62     'KidneyCancerCOI3', 'Kidney Cancer CO3 Year 3+');
63
64   bWTP := 364060;
65   UpdateWTPYearly;
66
67 end;
68
69 end.
70
71
72
```

code\BenefitModules\bmPFASLBW.pas

```
1 unit bmPFASLBW;
2
3 interface
4
5 uses SysUtils, Classes, Math, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
6      SafeWaterUtility, SafeWaterUncertBucket, SafewaterMetrics, SafeWaterGlobals,
7      SafeWaterCalcDist, PFASSerumCalc, SafewaterCohortArray, Generics.Collections,
8      StrUtils,
9      bmPFASBase ;
10
11
12 const aBWMin = 0; aBWMax=81; aBWInc=100;
13
14 type
15   TbmLBWOutputs=record
16     Fatal, Grams, AffectedBirths, AffectedBirths200 : double;
17   end;
18   TbmLBWDollarOutputs=record
19     NonFatalDollars : double;
```

```
20    end;
21
22    TPopCounts = array[reWhite..reOther] of double;
23    TRaceRates = array[reWhite..reOther,aBWMin..aBWMax] of double;
24    TLBWData = class
25      Odds, Births, Deaths, InfantBW, UOD : TRaceRates;
26    end;
27    TLBWDataDict = TObjectDictionary<string,TLBWData>;
28
29    TbmPFASLBW = class(TbmPFASBase)
30    protected
31      _LBW : TbmLBWOutputs;
32      _LBWD : array of TbmLBWDollarOutputs;
33      _LBWC : array[reWhite..reOther] of TbmLBWOutputs;
34      _LBWI : array of TbmLBWOutputs;
35
36      //table of values by birth weight change category
37      LBWDataDict : TLBWDataDict;
38      Data: TLBWData;
39      tpChildBearing,tpAll : TPopCounts;
40      BetaPFOA, BetaPFOS: TDistributed;
41      //TODO for now, hard coding COI table size based on LBW_COI.csv
42      COI: array[907..4536,0..2] of double;
43      StartAge, EndAge: integer;  //age to start calc - just a time saver...
44
45      procedure LoadParms;
46      procedure GetChildBearingPop;
47      procedure SetupModule; override;
48      procedure BreakdownModule; override;
49      procedure CalcCases(const P: TPWSRecord; const OnYr, OnChangeYear: integer);
50    public
51
52      procedure ResetBenefits(); override;
53      procedure SetupUncertainty(Unc: TUncertaintyStudy); override;
54      procedure SetupOutputs(Metrics: TCategoryMetrics); override;
55      procedure CollectPWSWideEPValues(const P: TPWSRecord); override;
56      procedure SetPWSWideEPValues(const P: TPWSRecord);  override;
57      procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit); override;
58    end;
59
60  implementation
61
62  { TbmPFASLBW }
63
64  procedure TbmPFASLBW.AddEntryPointBenefits(const P: TPWSRecord;
65    const Conc, FinalConc: TContamArray; const CostInfo: TCostInfo2Benefit);
66  begin
67    //always call inherited
68    inherited;
69    if not (PFOACHange or PFOSCHange) then exit;
70    {$IfDef DEBUG}
71    StartTimer;
72    {$endif}
73
74    //get total child bearing pop by race/eth
75    GetChildBearingPop;
76
77    var OnYr, YearsSinceChange : integer;
78    for var Yr := FData.Cfg.PFASChangeYear to FData.Cfg.EndYear do begin
79      OnYr := Yr - FData.Cfg.StartYear;
80      YearsSinceChange := Yr - FData.Cfg.PFASChangeYear;
81      sPFOA.GetBeforeAfterSerum(basePFOA, scenPFOA, YearsSinceChange, bcPFOA, acPFOA, FData.Cfg.DW_RSC, 0);
82      sPFOS.GetBeforeAfterSerum(basePFOS, scenPFOS, YearsSinceChange, bcPFOS, acPFOS, FData.Cfg.DW_RSC, 0);
83      CalcCases(P, OnYr, YearsSinceChange);
84      AdvanceCohort;
85      if FUseCF then GetChildBearingPop;
86    end;
87
88    {$IfDef DEBUG}
89    TickTimer;
90    {$endif}
91
92  end;
93
94  procedure TbmPFASLBW.BreakdownModule;
```

```
95  begin
96    inherited;
97    LBWDataDict.Free;
98  end;
99
100 procedure TbmPFASLBW.CalcCases(const P: TPWSRecord; const OnYr, OnChangeYear: integer);
101 var a,r, bwbin,coibin, ibw: integer;
102     d,deltabw, pc_mort, bw, birthcount, nfd, mort,
103     red_mort, rem_births, tmp: double;
104
105 begin
106   d:=0;
107   nfd:=0;
108   var G200 : boolean;
109   const CalYear = FData.Cfg.StartYear + OnYr;
110   for a := StartAge to EndAge do begin
111     for r := reWhite to reOther do begin
112       if CohortPop[gFemale,r,a] =  0 then continue;
113       deltabw := BetaPFOA.fValue * (scenPFOA[gFemale,r,a] - basePFOA[gFemale,r,a])
114                  + BetaPFOS.fValue * (scenPFOS[gFemale,r,a] - basePFOS[gFemale,r,a]);
115       G200 := deltabw > 200;
116       deltabw := min(deltabw,200);
117       for bwbin := aBWMin to aBWMax do begin
118         if Data.Births[r, bwBin] = 0 then continue;
119
120         birthcount := Data.Births[r, bwBin]
121                       * tpAll[r]
122                       * (CohortPop[gFemale,r,a] / tpChildBearing[r]);
123
124         _i(_LBW.Grams, deltabw * birthcount);
125         _i(_LBW.AffectedBirths, birthcount);
126         if G200 then
127           _i(_LBW.AffectedBirths200, birthcount);
128
129         tmp := exp(Data.Odds[r,bwbin] * deltabw);
130
131         pc_mort := ( tmp * Data.uod[r,bwbin]) / (1 + tmp * Data.uod[r,bwbin]);
132
133         mort := birthcount * pc_mort;
134         red_mort :=  (birthcount * Data.Deaths[r,bwbin]) - mort ;
135
136         _i(_LBW.Fatal, red_mort);
137         _i(_LBWC[r].Fatal, red_mort);
138         _i(_LBWC[r].Grams, deltabw * birthcount);
139         for var ii := 1 to length(EJProps)-1 do begin
140           _i(_LBWI[ii].Grams, deltabw * birthcount * EJProps[ii]);
141           _i(_LBWI[ii].Fatal, red_mort * EJProps[ii]);
142         end;
143
144         d := d + red_mort * FGlb.VSLY[CalYear];
145
146         //Morbidity
147         rem_births := birthcount - mort;
148         bw := Data.InfantBW[r,bwbin];
149         ibw := round(bw);
150         //keep BW within within COI ranges
151         if ibw < low(COI) then ibw:=low(COI);
152         if ibw > high(COI) then ibw:=high(COI);
153
154         if deltabw <=18.14 then COIBin := 0 else
155         if deltabw <=49.89 then COIBin := 1 else
156            COIBin := 2;
157
158         tmp := - rem_births * deltabw * COI[ibw,COIBin];
159         nfd := nfd + tmp;
160         d := d + tmp;
161
162         {$ifdef DEBUG}
163         if _d then begin
164           FData.DebugLog.L('pfaslbw',Format('%d,%s,%d,%d,%d,'+
165                              '%g,%g,'+
166                              '%g,%g,'+
167                              '%g,%g,%g,%g',
168                             [OnChangeYear,P.PWSId,r,a,bwbin,
```

```
169                             scenPFOA[gFemale,r,a],basePFOA[gFemale,r,a],
170                             scenPFOS[gFemale,r,a],basePFOS[gFemale,r,a],
171                             CohortPop[gFemale,r,a],birthcount,red_mort,deltabw
172                             ]));
173          end;
174         {$endif}
175
176        end;
177      end;
178    end;
179    //discount and accumulate
180    for var i := 0 to FData.Inp.RateCount-1 do begin
181      _i(_LBWD[i].NonfatalDollars, nfd / FData.Inp.PreCalcDiscRate[i,OnYr+FData.Cfg.PeriodAdjust]);
182      _i(O[i].Dollars, d / FData.Inp.PreCalcDiscRate[i,OnYr+FData.Cfg.PeriodAdjust]);
183      _i(OYearly[OnYr,i].Dollars, d / FData.Inp.PreCalcDiscRate[i,OnYr+FData.Cfg.PeriodAdjust]);
184    end;
185
186  end;
187
188  procedure TbmPFASLBW.CollectPWSWideEPValues(const P: TPWSRecord);
189  begin
190    inherited;
191    for var i := 0 to FData.Inp.RateCount-1 do
192      _LBWD[i].NonfatalDollars := _LBWD[i].NonfatalDollars * FData.Inp.PreCalcAnnRate[i, FData.Cfg.EndYear - FData.Cfg.StartYear + 1];
193  end;
194
195  procedure TbmPFASLBW.GetChildBearingPop;
196  begin
197    fillchar(tpChildBearing, sizeof(tpChildBearing),0);
198    fillchar(tpAll, sizeof(tpAll),0);
199    var a,r: integer;
200    for a := StartAge to EndAge do begin
201      for r := reWhite to reOther do begin
202        tpChildBearing[r] := tpChildBearing[r] + CohortPop[gFemale,r,a];
203      end;
204    end;
205    for a := 0 to CMaxAge do begin
206      for r := reWhite to reOther do begin
207        tpAll[r] := tpAll[r] + CohortPop[gFemale,r,a] + CohortPop[gMale,r,a];
208      end;
209    end;
210  end;
211
212  procedure TbmPFASLBW.LoadParms;
213  begin
214    var aFS := TBufferedFileStream.Create(DataPath+'CohortData\LBW\20220519_LBW_InputData.csv',
215               fmOpenRead + fmShareDenyNone, 32768);
216    var T := TStreamReader.Create(aFS);
217    var hdr := T.ReadLine;
218    var F: TArray<string>;
219    var IL,state: string;
220    var r, wb: integer;
221    var Empty: TRaceRates;
222    for r := reWhite to reOther do
223      for wb := aBWMin to aBWMax do
224        Empty[r,wb] := 0;
225
226    while not T.EndOfStream do begin
227      IL := T.ReadLine;
228      IL := StringReplace(IL,'"','',[rfReplaceAll]);
229      F := SplitString(IL, ',');
230      var cc := lowercase(F[0]);
231      if cc = 'nhwhite' then r := reWhite else
232      if cc = 'other' then r := reOther else
233      if cc = 'nhblack' then r := reBlack else
234      if cc = 'hispanic' then r := reHispanic;
235      state := uppercase(F[2]);
236      wb := strtoint(F[1]) DIV aBWInc;
237      if (wb>=aBWMin) and (wb<=aBWMax) then begin
238        if not LBWDataDict.TryGetValue(state,Data) then begin
239          Data := TLBWData.Create;
240          LBWDataDict.AddOrSetValue(state,Data);
241        end;
242        Data.Odds[r,wb] := ln(strtofloat(F[6]));
243        Data.Births[r,wb] := strtofloat(F[3])/1000;
```

```
244       Data.Deaths[r,wb] := strtofloat(F[5])/1000;
245       Data.InfantBW[r,wb] := strtofloat(F[4]);
246       if (Data.Births[r,wb] > 0) and (Data.Deaths[r,wb]<>1) then
247         Data.UOD[r,wb] := Data.Deaths[r,wb] / (1 - Data.Deaths[r,wb])
248       else
249         Data.UOD[r,wb] := 99999;
250
251     end;
252   end;
253   T.Free;
254   aFS.Free;
255
256   aFS := TBufferedFileStream.Create(DataPath+'CohortData\LBW\LBW_COI.csv',
257             fmOpenRead + fmShareDenyNone, 32768);
258   T := TStreamReader.Create(aFS);
259   hdr := T.ReadLine;
260   while not T.EndOfStream do begin
261     IL := T.ReadLine;
262     IL := StringReplace(IL,'"','',[rfReplaceAll]);
263     F := SplitString(IL, ',');
264     var v := strtoint(F[0]);
265     //convert per half pound to per gram
266     COI[v,0]:=strtofloat(F[1])/226.796;
267     COI[v,1]:=strtofloat(F[2])/226.796;
268     COI[v,2]:=strtofloat(F[3])/226.796;
269   end;
270   T.Free;
271   aFS.Free;
272 end;
273
274
275 procedure TbmPFASLBW.ResetBenefits;
276 begin
277   inherited;
278   _LBW := Default(TbmLBWOutputs);
279   for var r  := reWhite  to reOther do
280     _LBWC[r] := Default(TbmLBWOutputs);
281   for var r  := 0  to length(_LBWI) -1  do
282     _LBWI[r] := Default(TbmLBWOutputs);
283   for var i := 0 to FData.Inp.RateCount-1 do
284     _LBWD[i] := Default(TbmLBWDollarOutputs)
285 end;
286
287 procedure TbmPFASLBW.SetPWSWideEPValues(const P: TPWSRecord);
288 begin
289   inherited;
290   //TODO using WA state as default per ICF code....
291   if not LBWDataDict.TryGetValue(P.State, Data) then
292     LBWDataDict.TryGetValue('WA', Data);
293
294 end;
295
296 procedure TbmPFASLBW.SetupModule;
297 begin
298   inherited;
299   FUseCF:=False;  //Austin H.  directed LBW to no use CF technique 7/19/22
300   StartAge := 15;
301   EndAge := 44;
302   //TODO guard against not finding calculators...
303   sPFOA := FData.SerumCalc.GetSerumCalc('PFOA');
304   sPFOS := FData.SerumCalc.GetSerumCalc('PFOS');
305   LBWDataDict := TLBWDataDict.Create([doOwnsValues]);
306   LoadParms;
307 end;
308
309 procedure TbmPFASLBW.SetupOutputs(Metrics: TCategoryMetrics);
310 begin
311   inherited;
312   Metrics.AddMetric(fName+'.AffectedBirths',
313                     'PFASLBW Affected Births', @_LBW.AffectedBirths, mtBenefitCases,mcSumOnly, False);
314   Metrics.AddMetric(fName+'.AffectedBirths200',
315                     'PFASLBW Affected Births > 200 grams', @_LBW.AffectedBirths200, mtBenefitCases,mcSumOnly, False);
316
317   Metrics.AddMetric(fName+'.Fatal',
```

```
                    'PFASLBW Fatal Cases', @_LBW.Fatal, mtBenefitCases,mcSumOnly, False);
  Metrics.AddMetric(fName+'.Grams',
                    'PFASLBW Grams Cases', @_LBW.Grams, mtBenefitCases,mcSumOnly, False);

  setlength(_LBWD,FData.Inp.RateCount);
  for var i := 0 to FData.Inp.RateCount-1 do begin
    var tg := floattostr(FData.Inp.Rates[i]);
    Metrics.AddMetric(fName+'.NonfatalDollars',
                      'PFASLBW Non Fatal Dollars', @_LBWD[i].NonfatalDollars, mtBenefitDollars,mcSumOnly, False, tg );
  end;


  setlength(_LBWI,length(fData.EJProps.Names));
  if FData.Cfg.PFASEJOutput then begin
    for var i := 1 to length(fData.EJProps.Names) - 1 do begin
      var s := fData.EJProps.Names[i];
      Metrics.AddMetric(fName+'.Fatal'+s,
                      'PFASLBW Fatal Cases '+s, @_LBWI[i].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
      Metrics.AddMetric(fName+'.Grams'+s,
                      'PFASLBW Grams Cases '+s, @_LBWI[i].Grams, mtEJOutput,mcSumOnly,False, '', @EJInd);
    end;


    Metrics.AddMetric(fName+'.FatalWhite',
                      'PFASLBW Fatal Cases White', @_LBWC[reWhite].Fatal, mtEJOutput,mcSumOnly,False, '', @EJInd);
    Metrics.AddMetric(fName+'.FatalHispanic',
                      'PFASLBW Fatal Cases Hispanic', @_LBWC[reHispanic].Fatal, mtEJOutput,mcSumOnly,False, '',@EJInd);
    Metrics.AddMetric(fName+'.FatalBlack',
                      'PFASLBW Fatal Cases Black', @_LBWC[reBlack].Fatal, mtEJOutput,mcSumOnly,False,'',@EJInd);
    Metrics.AddMetric(fName+'.FatalOther',
                      'PFASLBW Fatal Cases Other', @_LBWC[reOther].Fatal, mtEJOutput,mcSumOnly,False,'', @EJInd);


    Metrics.AddMetric(fName+'.GramsWhite',
                      'PFASLBW Grams Cases White', @_LBWC[reWhite].Grams, mtEJOutput,mcSumOnly,False,'',@EJInd);
    Metrics.AddMetric(fName+'.GramsHispanic',
                      'PFASLBW Grams Cases Hispanic', @_LBWC[reHispanic].Grams, mtEJOutput,mcSumOnly,False,'',@EJInd);
    Metrics.AddMetric(fName+'.GramsBlack',
                      'PFASLBW Grams Cases Black', @_LBWC[reBlack].Grams, mtEJOutput,mcSumOnly,False,'', @EJInd);
    Metrics.AddMetric(fName+'.GramsOther',
                      'PFASLBW Grams Cases Other', @_LBWC[reOther].Grams, mtEJOutput,mcSumOnly,False,'', @EJInd);
  end;


end;

procedure TbmPFASLBW.SetupUncertainty(Unc: TUncertaintyStudy);
begin
  inherited;
  BetaPFOA := Unc.AddInVar(nil, dNormal, -10.5, 0, 0, -9E99, -10.5, 3.14, 0,
    'BetaPFOALBW', 'BetaPFOA LBW');
  BetaPFOS := Unc.AddInVar(nil, dNormal, -3.0, 0, 0, -9E99, -3.0, 0.97, 0,
    'BetaPFOSLBW', 'BetaPFOS LBW');

end;

end.
```

code\BenefitModules\bmPFASLiverC.pas

```
unit bmPFASLiverC;

interface

uses SysUtils, Classes, Math, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
     SafeWaterUtility, SafeWaterUncertBucket, SafewaterMetrics, SafeWaterGlobals,
     SafeWaterCalcDist, PFASSerumCalc, SafewaterCohortArray,
     bmPFASBase, bmPFASAbstractCancSC ;

type
```

```
  TbmPFASLiverC = class(TbmAbstractCancSC)
  protected
    procedure LoadParms; override;
  public
    procedure SetupUncertainty(Unc: TUncertaintyStudy); override;
  end;


implementation

{ TbmPFASLiverC }

procedure TbmPFASLiverC.LoadParms;
begin
  SumProbSurvAge.LoadFromFile(DataPath+'CohortData\LiverCancer\cumsum_p_cancer_survival_through_age.csv');
  ProbSurvAge.LoadFromFile(DataPath+'CohortData\LiverCancer\p_cancer_survival_through_age.csv');
  GenProb.LoadFromFile(DataPath+'CohortData\LiverCancer\gen_prob_mortality.csv');
  MortShare.LoadFromFile(DataPath+'CohortData\LiverCancer\liver_cancer_mortality_share.csv');
  //loading files for pre-calcs...
  dist.LoadFromFile(DataPath+'CohortData\LiverCancer\distant_cancer.csv');
  reg.LoadFromFile(DataPath+'CohortData\LiverCancer\regional_cancer.csv');
  local.LoadFromFile(DataPath+'CohortData\LiverCancer\localized_cancer.csv');
  uns.LoadFromFile(DataPath+'CohortData\LiverCancer\unstaged_cancer.csv');
  for var y := 0 to 10 do begin
    ps_dist[y].LoadFromFile(DataPath+'CohortData\LiverCancer\prob_survival_distant_0_11.csv',3+y);
    ps_reg[y].LoadFromFile(DataPath+'CohortData\LiverCancer\prob_survival_regional_0_11.csv',3+y);
    ps_local[y].LoadFromFile(DataPath+'CohortData\LiverCancer\prob_survival_localized_0_11.csv',3+y);
    ps_uns[y].LoadFromFile(DataPath+'CohortData\LiverCancer\prob_survival_unstaged_0_11.csv',3+y);
  end;
end;

procedure TbmPFASLiverC.SetupUncertainty(Unc: TUncertaintyStudy);
begin
  Slope := Unc.AddInVar(nil, dNone, 0.000002688/0.01034257, 0, 0, -9E99, 0.000002688/0.01034257, 0, 0,
   'PFOSLiverCancerSlope', 'Liver Cancer Slope PFOS');

  //Not Used
  C_Share := Unc.AddInVar(nil, dNone, 1, 0, 0, -9E99, 1, 0, 0,
    'LiverCancerC_Share', 'Liver Cancer Param Not Used');

  bWTP := 364060;
  UpdateWTPYearly;
end;

end.
```

code\BenefitModules\bmPFASStats.pas

```
unit bmPFASStats;

interface

uses SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterBenefitBase,
     SafeWaterUtility, SafeWaterUncertBucket, SafewaterMetrics, SafewaterGlobals,
     bmPFASBase ;


type
  TbmPFASStatsOutputs=record
    PFOASerumB, PFOSSerumB: double;
    PFOASerumA, PFOSSerumA: double;
  end;


  TbmPFASStats = class(TbmPFASBase)
    _StatO: TbmPFASStatsOutputs;

    procedure ResetBenefits(); override;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
```

```
    procedure SetupUncertainty(Unc: TUncertaintyStudy); override;

    procedure AddEntryPointBenefits(const P: TPWSRecord; const Conc, FinalConc : TContamArray; const CostInfo: TCostInfo2Benefit); override;
  end;


implementation

{ TbmPFASStats }

procedure TbmPFASStats.AddEntryPointBenefits(const P: TPWSRecord; const Conc,
  FinalConc: TContamArray; const CostInfo: TCostInfo2Benefit);
begin
  inherited;
  var bspfoa, sspfoa, bspfos, sspfos: double;
  bspfoa:=0;
  sspfoa:=0;
  bspfos:=0;
  sspfos:=0;
  sPFOA.GetBeforeAfterSerum(basePFOA, scenPFOA, 20, bcPFOA, acPFOA, 0.2);
  sPFOS.GetBeforeAfterSerum(basePFOS, scenPFOS, 20, bcPFOS, acPFOS, 0.2);
  for var s := gMale to gFemale do begin
    for var r := reWhite to reOther do begin
      //for var a := 0 to CMaxAge do begin
      for var a := 40 to 40 do begin
         bspfoa := bspfoa + basePFOA[s,r,a] * P.CohortPop[s,r,a];
         sspfoa := sspfoa + scenPFOA[s,r,a] * P.CohortPop[s,r,a];
         bspfos := bspfos + basePFOS[s,r,a] * P.CohortPop[s,r,a];
         sspfos := sspfos + scenPFOS[s,r,a] * P.CohortPop[s,r,a];
      end;
    end;
  end;

  _StatO.PFOASerumB := bspfoa / P.Population;
  _StatO.PFOASerumA := sspfoa / P.Population;
  _StatO.PFOSSerumB := bspfos / P.Population;
  _StatO.PFOSSerumA := sspfos / P.Population;

end;

procedure TbmPFASStats.ResetBenefits;
begin
  inherited;
  _StatO := Default(TbmPFASStatsOutputs);
end;

procedure TbmPFASStats.SetupOutputs(Metrics: TCategoryMetrics);
begin
  //Note: inherited not called here - we are creating serum results here - cases, dollars don't matter
  //inherited;
  Metrics.AddMetric(fName+'.PFOASerumB',
                    fName + ' PFOA Serum Level (40yo) Base', @_StatO.PFOASerumB, mtInfo,mcMeanOnly, False);
  Metrics.AddMetric(fName+'.PFOSSerumB',
                    fName + ' PFOS Serum Level (40yo) Base', @_StatO.PFOSSerumB, mtInfo,mcMeanOnly, False);
  Metrics.AddMetric(fName+'.PFOASerumA',
                    fName + ' PFOA Serum Level (40yo) Option', @_StatO.PFOASerumA, mtInfo,mcMeanOnly, False);
  Metrics.AddMetric(fName+'.PFOSSerumA',
                    fName + ' PFOS Serum Level (40yo) Option', @_StatO.PFOSSerumA, mtInfo,mcMeanOnly, False);

end;

procedure TbmPFASStats.SetupUncertainty(Unc: TUncertaintyStudy);
begin
  inherited;

end;

end.

```

code\CalcDistSpecFun.pas

```
{$N+}
```

```
unit CalcDistSpecFun;

interface

uses Math;

//TODO move this and other support functions in CalcDists into the more general numerics lib.
//Chaff in here 6/96
const

  FLOATEPS = 3.4E-9; // TODO check this since this is old code...

  CBRT2 = 1.2599210498948731647672; { cube root of 2 }
  CBRT3 = 1.4422495703074083823216; { cube root of 3 }
  D2R = 0.017453292519943295769237; { # radians in a degree }
  E = 2.71828182845904523536 03; { base of natural logs }
  EULERC = 0.57721566490153286060 65; { Euler's constant }
  LN2 = 0.69314718055994530941 72; { natural logarithm of 2 }
  LN3 = 1.0986122886681096913952; { natural logarithm of 3 }
  LN10 = 2.3025850929940456840280; { natural logarithm of 10 }
  LNPI = 1.1447298858494001741434; { natural logarithm of PI }
  LOG2E = 1.4426950408889634073599; { 1/ln(2) }
  LOG10E = 0.43429448190325182765 11; { 1/ln(10) }
  M2R = 0.00029088820866572159615 4; { # rads in a minute }
  PI = 3.14159265358979323846 26; { plain ol' round pi }
  PI_2 = 1.5707963267948966192313; { pi/2 }
  PI_4 = 0.7853981633974483096156; { pi/4 }
  R1_E = 0.36787944117144232159 55; { 1/e }
  R1_PI = 0.31830988618379067153 78; { 1/pi }
  R1_SQRTPI = 0.5641895835477562869481; { 1/sqrt(pi) }
  R2_PI = 0.63661977236758134307 55; { 2/pi }
  R2_SQRTPI = 1.12837916709551257390; { 2/sqrt(pi) }
  R2D = 57.295779513082320876798; { # degrees in a radian }
  S2R = 0.00000484813681109535993 6; { # rads in a second }
  SQRPI = 9.8696044010893586188345; { sqr(pi) (not round!) }
  SQRT_2 = 0.70710678118654752440 1; { sqrt(1/2) }
  SQRT2 = 1.4142135623730950488017; { sqrt(2) }
  SQRT2PI = 2.5066282746310005024158; { sqrt(2*pi) }
  SQRT3 = 1.7320508075688772935; { sqrt(3) }
  SQRT5 = 2.2360679774997896964; { sqrt(5) }
  SQRTPI = 1.7724538509055160272982; { sqrt(pi) = gamma(1/2) }
  TWOPI = 6.2831853071795864769253; { 2*pi }

type
  float = double;
  xfloat = extended;

function factorial(n: word): xfloat;
function lnfactorial(n: word): xfloat;

function permutation(n, k: word): xfloat;
function combination(n, k: word): xfloat;

function calcgamma(x: xfloat): xfloat;
function lngamma(x: xfloat): xfloat;

function beta(x, y: xfloat): xfloat;
function lnbeta(x, y: xfloat): xfloat;

function igamma(a, x: xfloat): xfloat;
function igammac(a, x: xfloat): xfloat;
function igammap(a, x: xfloat): xfloat;
function igammaq(a, x: xfloat): xfloat;

function ibeta(a, b, x: xfloat): xfloat;
function ibetap(a, b, x: xfloat): xfloat;
function ibetaq(a, b, x: xfloat): xfloat;

function erf(x: xfloat): xfloat;
function erfc(x: xfloat): xfloat;

implementation

{ IBETACF is a "continued fraction" evaluation of an incomplete
  beta auxiliary function. (See HMF: eqn. 26.5.8; NR: BETACF ) }
```

```
function ibetacf(a, b, x: xfloat): xfloat;
var
  am, ap, apb, app, azlast, az, bm, bp, bpp, bz, d, deca, inca: xfloat;
  m, tm: longint;
begin
  am := 1;
  az := 1;
  bm := 1;
  m := 1;
  apb := a + b;
  deca := a - 1;
  inca := a + 1;
  bz := 1 - apb * x / inca;
  tm := 2;
  d := (b - 1) * x / (inca * (a + 2));
  ap := d + 1;
  bp := bz + d;
  d := -inca * (apb + 1) * x / ((a + 2) * (a + 3));
  app := ap + d;
  bpp := bp + d * bz;
  am := ap / bpp;
  bm := bp / bpp;
  azlast := az;
  az := app / bpp;
  while abs(az - azlast) > (abs(az) * FLOATEPS) do begin
    inc(m);
    inc(tm, 2);
    d := m * (b - m) * x / ((deca + tm) * (a + tm));
    ap := az + d * am;
    bp := d * bm + 1;
    d := -(a + m) * (apb + m) * x / ((a + tm) * (inca + tm));
    app := ap + d * az;
    bpp := bp + d
  end;
  ibetacf := az
end; { ibetacf }

{ IGAMMACF is a "continued fraction" evaluation of an incomplete
  gamma auxiliary function.  ( See HMF: eqn. 6.5.31; NR: GCF ) }

function igammacf(a, x: xfloat): xfloat;
var
  n: longint;
  a0, a1, b0, b1, f, fn, g, glast, na: xfloat;
begin
  a0 := 1;
  a1 := x;
  b0 := 0;
  b1 := 1;
  f := 1;
  g := 0;
  glast := 1;
  n := 0;
  repeat
    inc(n);
    fn := f * n;
    na := n - a;
    a0 := (a1 + a0 * na) * f;
    b0 := (b1 + b0 * na) * f;
    a1 := x * a0 + fn * a1;
    b1 := x * b0 + fn * b1;
    if (a1 <> 0) then begin
      f := 1 / a1;
      glast := g;
      g := b1 * f
    end;
  until abs(g - glast) < (abs(g) * FLOATEPS);
  igammacf := g;
end; { igammacf }

{ IGAMMASER is a "power series" evaluation of the incomplete
  gamma auxiliary function.  ( See HMF: eqn. 6.5.29; NR: GSER ) }
```

```
151 function igammaser(a, x: xfloat): xfloat;
152 var
153   d, s: xfloat;
154 begin
155   if x = 0 then
156     igammaser := 0
157   else begin
158     d := 1 / a;
159     s := d;
160     repeat
161       a := a + 1;
162       d := d * x / a;
163       s := s + d;
164     until abs(d) < (abs(s) * FLOATEPS);
165     igammaser := s
166   end
167 end; { igammaser }
168
169 { *** interface'd functions in alphabetical order *** }
170
171 function beta;
172 begin
173   beta := exp(lnbeta(x, y))
174 end;
175
176 function combination;
177 begin
178   combination := round(exp(lnfactorial(n) - lnfactorial(k) -
179     lnfactorial(n - k)))
180 end;
181
182 function iif(p: boolean; t, f: xfloat): xfloat;
183 begin
184   if p then
185     iif := t
186   else
187     iif := f
188 end;
189
190 function sgn(const x: xfloat): xfloat;
191 begin
192   if x = 0 then
193     sgn := 0
194   else
195     sgn := iif(x > 0, 1, -1)
196 end;
197
198 function erf;
199 var
200   y, z: xfloat;
201 begin
202   if abs(x) < FLOATEPS then
203     erf := x * R2_SQRTPI
204   else
205     if abs(x) > 10 then
206       erf := sgn(x)
207     else begin
208       y := sqr(x);
209       z := x * exp(-y - SQRTPI);
210       if y < 1.5 then
211         erf := z * igammaser(0.5, y)
212       else
213         erf := sgn(x) - z * igammacf(0.5, y)
214     end
215 end; { erf }
216
217 function erfc;
218 begin
219   erfc := 1 - erf(x)
220 end;
221
222 function factorial;
223 const
224   MAXATYPE = 32;
225 type
```

```
  atype = array [0 .. MAXATYPE] of xfloat;
var
  a: ^atype;
  atop: shortint;
  j: byte;
begin
  a := nil;
  atop := -1;
  if a = NIL then begin
    new(a);
    if a <> NIL then begin
      a^[0] := 1;
      atop := 0
    end
  end;
  if (a <> NIL) and (n > atop) and (n <= MAXATYPE) then begin
    for j := succ(atop) to n do
      a^[j] := a^[pred(j)] * j;
    atop := n
  end;
  if (n <= atop) then
    factorial := a^[n]
  else
    factorial := round(calcgamma(succ(n)))
end; { factorial }

function calcgamma;
var
  y: xfloat;
begin
  y := exp(lngamma(x));
  if x < 0 then
    if not odd(trunc(x)) then
      y := -y;
  calcgamma := y
end;

function ibeta;
var
  y: xfloat;
begin
  if x <= 0 then
    ibeta := 0
  else
    if x >= 1 then
      ibeta := beta(a, b)
    else begin
      y := exp(a * ln(x) + b * ln(1 - x));
      if (x * (a + b + 2)) <= (a + 1) then
        ibeta := y * ibetacf(a, b, x) / a
      else
        ibeta := beta(a, b) - y * ibetacf(b, a, 1 - x) / b
    end
end; { ibeta }

function ibetap;
var
  y: xfloat;
begin
  if x <= 0 then
    ibetap := 0
  else
    if x >= 1 then
      ibetap := 1
    else begin
      y := exp(-lnbeta(a, b) + a * ln(x) + b * ln(1 - x));
      if x <= (a + 1) / (a + b + 2) then
        ibetap := y * ibetacf(a, b, x) / a
      else
        ibetap := 1 - y * ibetacf(b, a, 1 - x) / b
    end
end; { ibetap }

function ibetaq;
```

```
begin
  ibetaq := 1 - ibetap(b, a, 1 - x)
end;

function igamma;
begin
  if x <= 0 then
    igamma := 0
  else
    if x < (a + 1) then
      igamma := exp(-x + a * ln(x)) * igammaser(a, x)
    else
      igamma := calcgamma(a) - igammac(a, x)
end;

function igammac;
begin
  if x < (a + 1) then
    igammac := calcgamma(a) - igamma(a, x)
  else
    igammac := exp(-x + a * ln(x)) * igammacf(a, x)
end;

function igammap;
begin
  if x <= 0 then
    igammap := 0
  else
    if x < (a + 1) then
      igammap := exp(-x + a * ln(x) - lngamma(a)) * igammaser(a, x)
    else
      igammap := 1 - igammaq(a, x)
end;

function igammaq;
begin
  if x < (a + 1) then
    igammaq := 1 - igammap(a, x)
  else
    igammaq := exp(-x + a * ln(x) - lngamma(a)) * igammacf(a, x)
end;

function lnbeta;
begin
  lnbeta := lngamma(x) + lngamma(y) - lngamma(x + y)
end;

function lnfactorial;
const
  MAXATYPE = 99;
type
  atype = array [0 .. MAXATYPE] of xfloat;
var
  a: ^atype;
  j: byte;
begin
  a := nil;
  if a = NIL then begin
    new(a);
    if a <> NIL then
      for j := 0 to MAXATYPE do
        a^[j] := -1
  end;
  if (n < MAXATYPE) and (a <> NIL) then begin
    if a^[n] < 0 then
      a^[n] := lngamma(succ(n));
    lnfactorial := a^[n]
  end
  else
    lnfactorial := lngamma(succ(n))
end; { lnfactorial }

function lngamma;
var
  s, t: xfloat;
```

```
begin
  if x < 0 then
    lngamma := LNPI - lngamma(1 - x) - ln(sin(frac(x * 0.5) * TWOPI))
  else
    if x < 1 then
      lngamma := lngamma(x + 1) - ln(x)
    else begin
      t := x + 4.5;
      t := (x - 0.5) * ln(t) - t;
      s := 1 + 76.18009173 / x - 86.50532033 / (x + 1) + 24.01409822 / (x + 2) -
        1.231739516 / (x + 3) + 1.20858003E-3 / (x + 4) - 5.3682E-6 / (x + 5);
      lngamma := t + ln(SQRT2PI * s)
    end;
end; { lngamma }

function permutation;
begin
  permutation := round(exp(lnfactorial(n) - lnfactorial(n - k)))
end;

end. { unit specfun/specf87 }

```

code\CostModules\CompiledCost.pas

```
unit CompiledCost;

// Autogenerated unit - DO NOT ALTER
// Generated - 12/5/2023 12:26:59 PM
// Baseline Workbook -
// Found variables in workbook - 0
// Found costing steps - 0
// Option Workbook - K:\Safewater2022\data\CostWorkbook\PFAS_AdminCosts_v17_Expressions.csv
// Found variables in workbook - 11
// Found costing steps - 58

interface

uses Math, Classes, SysUtils, SafewaterGlobals, SafewaterUtility, SafewaterPWSRecords,
  SafewaterModelData;

type
  TCCValueStore = array [0 .. 57] of double;
  // max 6 disc rates including 0
  TCCValueAccum = array [0 .. 5, 0 .. 57] of pdouble;
  TCCValueYearlyAccum = array [0 .. 100, 0 .. 5, 0 .. 57] of pdouble;
  TCCNameStore = array [0 .. 57] of string;
  TCCValueStoreFlag = array [0 .. 57] of boolean;
  // added large,small system split
  TCCCalcYearStore = array [0 .. 1, 0 .. 100, 0 .. 57] of boolean;

  _VariablesRecBASELINE = record
    d: pdouble;
  end;

  _VariablesRecOPTION = record
    aw, b_annual, b_exceed, b_nondetect, b_quarterly, b_source, b_treat, b_triennial, epa533, epa537,
      statepfas: pdouble;

  end;

  TCCCompiledCost = class
    _CalcCost, _Cost, _Hours, _Labor: TCCValueStore;
    _pCost, _pCost_pws: TCCValueAccum;
    _pCostY, _pCostY_pws: TCCValueYearlyAccum;
    _Names, _Cat: TCCNameStore;
    _EPCost: TCCValueStoreFlag;
    _YearOK: TCCCalcYearStore;
    _HighDRIdx: integer;
    TotEval: int64;
    constructor create();
    procedure _Reset(ResetSystem: boolean);
    procedure _SetSingleYear(Yr: integer);
```

```
49      procedure _SetYears; virtual; abstract;
50      procedure _SetNames; virtual; abstract;
51      procedure _SetEPCosts; virtual; abstract;
52      procedure _LoadVars; virtual; abstract;
53      function _LoadVarsOK: boolean; virtual; abstract;
54      procedure _SetVarPointer(const name: string; const pd: pdouble); virtual; abstract;
55      procedure _SetOutPointer(var O: TMDRCostOutputs; var Y: TMDRYearlyCostOutputs); virtual;
56      procedure _Evaluate(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
57        const PeriodAdjust: integer); virtual; abstract;
58      procedure _EvaluateSystem(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
59        const PeriodAdjust: integer); virtual; abstract;
60      function _DumpVars: string; virtual; abstract;
61      function _DumpYearTable: TStringList;
62      function _DumpCalcs: TStringList;
63    end;
64
65    TCCCompiledCostBaseline = class(TCCCompiledCost)
66      _Variables: _VariablesRecBASELINE;
67
68      constructor create();
69      procedure _SetYears; override;
70      procedure _SetNames; override;
71      procedure _SetEPCosts; override;
72      procedure _LoadVars; override;
73      function _LoadVarsOK: boolean; override;
74      procedure _SetVarPointer(const name: string; const pd: pdouble); override;
75      procedure _Evaluate(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
76        const PeriodAdjust: integer); override;
77      procedure _EvaluateSystem(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
78        const PeriodAdjust: integer); override;
79      function _DumpVars: string; override;
80    end;
81
82    TCCCompiledCostOption = class(TCCCompiledCost)
83      _Variables: _VariablesRecOPTION;
84
85      constructor create();
86      procedure _SetYears; override;
87      procedure _SetNames; override;
88      procedure _SetEPCosts; override;
89      procedure _LoadVars; override;
90      function _LoadVarsOK: boolean; override;
91      procedure _SetVarPointer(const name: string; const pd: pdouble); override;
92      procedure _Evaluate(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
93        const PeriodAdjust: integer); override;
94      procedure _EvaluateSystem(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
95        const PeriodAdjust: integer); override;
96      function _DumpVars: string; override;
97    end;
98
99  const _CWorkBookBASELINE = '';
100   _CWorkBookOPTION = 'K:\Safewater2022\data\CostWorkbook\PFAS_AdminCosts_v17_Expressions.csv';
101   _CWorkBookDate = '12/5/2023 12:26:59 PM';
102
103 var _UseCompiled: boolean = true;
104
105 implementation
106
107 type
108   TSSArray = array [swGroundWater .. swSurfaceWater, ss0_100 .. ss1Mp] of double;
109
110 const Dummy = 1;
111   detection_level_pfoa: TSSArray = ((0.53, 0.53, 0.53, 0.53, 0.53, 0.53, 0.53, 0.53, 0.53),
112     (0.53, 0.53, 0.53, 0.53, 0.53, 0.53, 0.53, 0.53, 0.53));
113   detection_level_pfos: TSSArray = ((1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1),
114     (1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1));
115   detection_level_genx: TSSArray = ((1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9),
116     (1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9));
117   detection_level_pfhxs: TSSArray = ((1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4),
118     (1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4));
119   detection_level_pfna: TSSArray = ((0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7),
120     (0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7));
121   detection_level_pfhxa: TSSArray = ((1, 1, 1, 1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1, 1, 1, 1, 1));
122   detection_level_pfda: TSSArray = ((1.6, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6),
```

```
    (1.6, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6, 1.6));
  detection_level_pfhpa: TSSArray = ((0.71, 0.71, 0.71, 0.71, 0.71, 0.71, 0.71, 0.71, 0.71),
    (0.71, 0.71, 0.71, 0.71, 0.71, 0.71, 0.71, 0.71, 0.71));
  hrs_sys_adopt_rule: TSSArray = ((1.333333333, 1.333333333, 1.333333333, 1.333333333, 1.333333333, 1.333333333,
    1.333333333, 1.333333333, 1.333333333), (1.333333333, 1.333333333, 1.333333333, 1.333333333, 1.333333333,
    1.333333333, 1.333333333, 1.333333333, 1.333333333));
  hrs_pa_adopt_rule: TSSArray = ((1340, 1340, 1340, 1340, 1340, 1340, 1340, 1340, 1340),
    (1340, 1340, 1340, 1340, 1340, 1340, 1340, 1340, 1340));
  hrs_pa_write_reg: TSSArray = ((100, 100, 100, 100, 100, 100, 100, 100, 100), (100, 100, 100, 100, 100, 100, 100, 100,
    100)); hrs_sys_initial_ta: TSSArray = ((5.333333333, 5.333333333, 5.333333333, 5.333333333, 10.66666667,
    10.66666667, 10.66666667, 10.66666667, 10.66666667), (5.333333333, 5.333333333, 5.333333333, 5.333333333,
    10.66666667, 10.66666667, 10.66666667, 10.66666667, 10.66666667));
  hrs_pa_initial_ta: TSSArray = ((500, 500, 500, 500, 500, 500, 500, 500, 500), (500, 500, 500, 500, 500, 500, 500, 500,
    500)); hrs_pa_train: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_sdwis: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  numb_nondetect_samples: TSSArray = ((1, 1, 1, 1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1, 1, 1, 1, 1));
  hrs_samp: TSSArray = ((1, 1, 1, 1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1, 1, 1, 1, 1));
  hrs_sys_report_ep: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_pa_report_ep: TSSArray = ((1, 1, 1, 1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1, 1, 1, 1, 1));
  hrs_sys_waiver: TSSArray = ((16, 16, 16, 16, 16, 16, 16, 16, 16), (16, 16, 16, 16, 16, 16, 16, 16, 16));
  hrs_pa_waiver: TSSArray = ((8, 8, 8, 8, 8, 8, 8, 8, 8), (8, 8, 8, 8, 8, 8, 8, 8, 8));
  hrs_sys_treat: TSSArray = ((3, 5, 7, 12, 22, 22, 42, 42, 42), (3, 5, 7, 12, 22, 22, 42, 42, 42));
  hrs_pa_treat: TSSArray = ((80, 80, 80, 80, 70, 70, 50, 50, 50), (80, 80, 80, 80, 70, 70, 50, 50, 50));
  hrs_sys_source: TSSArray = ((6, 6, 6, 6, 6, 6, 6, 6, 6), (6, 6, 6, 6, 6, 6, 6, 6, 6));
  hrs_pa_source: TSSArray = ((4, 4, 4, 4, 4, 4, 4, 4, 4), (4, 4, 4, 4, 4, 4, 4, 4, 4));
  hrs_tier2_template: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_tier2_approve: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_tier2_distribute: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_tier2_certify: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_tier3_template: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_tier3_approve: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_tier3_distribute: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  hrs_tier3_certify: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  numb_hh: TSSArray = ((2.53, 2.53, 2.53, 2.53, 2.53, 2.53, 2.53, 2.53, 2.53),
    (2.53, 2.53, 2.53, 2.53, 2.53, 2.53, 2.53, 2.53, 2.53));
  tier2_cost: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  tier3_cost: TSSArray = ((0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0));
  labor_sys_rate: TSSArray = ((36.43, 36.43, 36.43, 36.43, 38.84, 41, 42.81, 50.03, 50.03),
    (36.43, 36.43, 36.43, 36.43, 38.84, 41, 42.81, 50.03, 50.03));
  labor_pa_rate: TSSArray = ((59.69, 59.69, 59.69, 59.69, 59.69, 59.69, 59.69, 59.69, 59.69),
    (59.69, 59.69, 59.69, 59.69, 59.69, 59.69, 59.69, 59.69, 59.69));
  epa533_cost: TSSArray = ((385, 385, 385, 385, 385, 385, 385, 385, 385), (385, 385, 385, 385, 385, 385, 385, 385, 385)
    ); epa537_cost: TSSArray = ((309, 309, 309, 309, 309, 309, 309, 309, 309), (309, 309, 309, 309, 309, 309, 309, 309,
    309)); ucmr5: TSSArray = ((0, 0, 0, 0, 1, 1, 1, 1, 1), (0, 0, 0, 0, 1, 1, 1, 1, 1));
  epa533_fieldblank_cost: TSSArray = ((335, 335, 335, 335, 335, 335, 335, 335, 335),
    (335, 335, 335, 335, 335, 335, 335, 335, 335));
  epa537_fieldblank_cost: TSSArray = ((273, 273, 273, 273, 273, 273, 273, 273, 273),
    (273, 273, 273, 273, 273, 273, 273, 273, 273));
  numb_initial_samples: TSSArray = ((2, 2, 2, 2, 2, 4, 4, 4, 4), (4, 4, 4, 4, 4, 4, 4, 4, 4));
  small_flex1: TSSArray = ((1, 1, 1, 1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1, 1, 1, 1, 1));

  { TCCCCompiledCost }

constructor TCCCompiledCost.create;
begin
  TotEval := 0;
  _SetYears;
  _SetNames;
  _SetEPCosts;
  for var i := 0 to high(_pCost) do begin
    for var j := 0 to high(_pCost[i]) do begin
      _pCost[i, j] := nil;
      _pCost_pws[i, j] := nil;
    end;
  end;
end;

function TCCCompiledCost._DumpCalcs: TStringList;
begin
  Result := TStringList.create;
  for var i := 0 to high(_Cost) do begin
    Result.Add(_Names[i] + ': ' + floattostr(_Cost[i]));
  end;
end;

```

```
198 function TCCCompiledCost._DumpYearTable: TStringList;
199 begin
200   Result := TStringList.create;
201   var tmps: string;
202   for var sz := 0 to 1 do begin
203     for var c := Low(_Names) to High(_Names) do begin
204       tmps := _Names[c] + ',' + sz.ToString + ',';
205       for var Y := 0 to 100 do begin
206         var
207         b := '';
208         if _YearOK[0, Y, c] then
209           b := '1';
210         tmps := tmps + b + ',';
211       end;
212       Result.Add(tmps);
213     end;
214   end;
215
216 end;
217
218 procedure TCCCompiledCost._Reset(ResetSystem: boolean);
219 var i: integer;
220 begin
221   for i := 0 to high(_CalcCost) do begin
222     if (ResetSystem and not _EPCost[i]) or (not ResetSystem and _EPCost[i]) then begin
223       _CalcCost[i] := 0;
224       _Cost[i] := 0;
225       // _Hours[i]:=0;
226       // _OM[i]:=0;
227       // _Labor[i]:=0;
228     end;
229   end;
230 end;
231
232 procedure TCCCompiledCost._SetOutPointer(var O: TMDRCostOutputs; var Y: TMDRYearlyCostOutputs);
233 // O is indexed by discount rate - Y is indexe by year for discount rate 0 only
234 begin
235   // TODO make these more general
236   _HighDRIdx := high(O);
237   var
238   iy := high(Y);
239   var yy: integer;
240   for var i := 0 to high(_Cat) do begin
241     var
242     s := _Cat[i];
243     for var dr := 0 to _HighDRIdx do begin
244       if s = 'Administration - System' then begin
245         _pCost[dr, i] := @O[dr].PWSImpAdmin;
246         _pCost_pws[dr, i] := @O[dr].PWSImpAdmin_pws;
247         for yy := 0 to iy do begin
248           _pCostY[yy, dr, i] := @Y[yy, dr].PWSImpAdmin;
249           _pCostY_pws[yy, dr, i] := @Y[yy, dr].PWSImpAdmin_pws;
250         end;
251       end else if s = 'Administration - Primacy Agency' then begin
252         _pCost[dr, i] := @O[dr].AgencyImpAdmin;
253         _pCost_pws[dr, i] := @O[dr].DontCare;
254         for yy := 0 to iy do begin
255           _pCostY[yy, dr, i] := @Y[yy, dr].AgencyImpAdmin;
256           _pCostY_pws[yy, dr, i] := @Y[yy, dr].DontCare;
257         end;
258       end else if s = 'Treatment Administration - System' then begin
259         _pCost[dr, i] := @O[dr].PWSTreatmentAdmin;
260         _pCost_pws[dr, i] := @O[dr].PWSTreatmentAdmin_pws;
261         for yy := 0 to iy do begin
262           _pCostY[yy, dr, i] := @Y[yy, dr].PWSTreatmentAdmin;
263           _pCostY_pws[yy, dr, i] := @Y[yy, dr].PWSTreatmentAdmin_pws;
264         end;
265       end else if s = 'Treatment Administration - Primacy Agency' then begin
266         _pCost[dr, i] := @O[dr].AgencyTreatmentAdmin;
267         _pCost_pws[dr, i] := @O[dr].DontCare;
268         for yy := 0 to iy do begin
269           _pCostY[yy, dr, i] := @Y[yy, dr].AgencyTreatmentAdmin;
270           _pCostY_pws[yy, dr, i] := @Y[yy, dr].DontCare;
271         end;
```

```
272       end else if s = 'Monitoring - System' then begin
273         _pCost[dr, i] := @O[dr].PWSSampling;
274         _pCost_pws[dr, i] := @O[dr].PWSSampling_pws;
275         for yy := 0 to iy do begin
276           _pCostY[yy, dr, i] := @Y[yy, dr].PWSSampling;
277           _pCostY_pws[yy, dr, i] := @Y[yy, dr].PWSSampling_pws;
278         end;
279       end else if s = 'Treatment - System' then begin
280         // ignore - handled elswhere with WBS curves
281       end else if s = 'Monitoring - Primacy Agency' then begin
282         _pCost[dr, i] := @O[dr].AgencyImpSampling;
283         _pCost_pws[dr, i] := @O[dr].DontCare;
284         for yy := 0 to iy do begin
285           _pCostY[yy, dr, i] := @Y[yy, dr].AgencyImpSampling;
286           _pCostY_pws[yy, dr, i] := @Y[yy, dr].DontCare;
287         end;
288
289       end
290       else
291         raise Exception.create('Aggregation Category Not Supported: ' + s);
292     end;
293   end;
294 end;
295
296 procedure TCCCompiledCost._SetSingleYear(Yr: integer);
297 begin
298   for var s := 0 to 1 do begin
299     for var Y := 0 to 100 do begin
300       if Y = Yr then
301         continue;
302       for var c := 0 to high(_YearOK[s, Y]) do
303         _YearOK[s, Y, c] := false;
304     end;
305   end;
306 end;
307
308 { TCCCompiledCostBaseline }
309
310 procedure TCCCompiledCostBaseline._LoadVars;
311 begin
312   (* LoadVarsBASELINE *)
313 end;
314
315 function TCCCompiledCostBaseline._LoadVarsOK: boolean;
316 begin
317   Result := (true);
318
319 end;
320
321 procedure TCCCompiledCostBaseline._Evaluate(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
322   const PeriodAdjust: integer);
323 begin
324   _Reset(false);
325   // _LoadVars();
326   var
327   sz := 0;
328   var v: double;
329   var DRIdx: integer;
330   if P.SystemSize >= ss3301_10k then
331     sz := 1;
332
333   with _Variables do begin
334
335   end;
336 end;
337
338 procedure TCCCompiledCostBaseline._EvaluateSystem(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
339   const PeriodAdjust: integer);
340 begin
341   _Reset(true);
342   // _LoadVars();
343   var
344   sz := 0;
345   var v: double;
346   var DRIdx: integer;
```

```
347   if P.SystemSize >= ss3301_10k then
348     sz := 1;
349
350   with _Variables do begin
351     (* EVALUATESYSTEMBASELINE *)
352   end;
353 end;
354
355 procedure TCCCompiledCostBaseline._SetEPCosts;
356 begin
357   for var i := low(_EPCost) to high(_EPCost) do
358     _EPCost[i] := true;
359   (* SetEPCostsBASELINE *)
360 end;
361
362 procedure TCCCompiledCostBaseline._SetNames;
363 begin
364   (* SetNamesBASELINE *)
365 end;
366
367 procedure TCCCompiledCostBaseline._SetVarPointer(const name: string; const pd: pdouble);
368 var s: string;
369 begin
370   s := lowercase(name);
371
372 end;
373
374 procedure TCCCompiledCostBaseline._SetYears;
375 begin
376   // Size, Year, CostID
377   (* SetYearsBASELINE *)
378 end;
379
380 function TCCCompiledCostBaseline._DumpVars: string;
381 var s: string;
382 begin
383   s := '';
384
385   Result := s;
386 end;
387
388 constructor TCCCompiledCostBaseline.create;
389 begin
390   inherited create;
391
392 end;
393
394 { TCCCompiledCostOption }
395
396 procedure TCCCompiledCostOption._LoadVars;
397 begin
398   (* LoadVarsOPTION *)
399 end;
400
401 function TCCCompiledCostOption._LoadVarsOK: boolean;
402 begin
403   Result := (Assigned(_Variables.aw) and Assigned(_Variables.b_annual) and Assigned(_Variables.b_exceed) and
404     Assigned(_Variables.b_nondetect) and Assigned(_Variables.b_quarterly) and Assigned(_Variables.b_source) and
405     Assigned(_Variables.b_treat) and Assigned(_Variables.b_triennial) and Assigned(_Variables.epa533) and
406     Assigned(_Variables.epa537) and Assigned(_Variables.statepfas));
407
408 end;
409
410 procedure TCCCompiledCostOption._Evaluate(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
411   const PeriodAdjust: integer);
412 begin
413   _Reset(false);
414   // _LoadVars();
415   var
416   sz := 0;
417   var DRIdx: integer;
418   var v: double;
419   if P.SystemSize >= ss3301_10k then
420     sz := 1;
```

```
421
422   with _Variables do begin
423
424     // sys_initial_monitor_EPA533  row: 8
425     if (_YearOK[sz, Yr, 6]) then begin
426       _CalcCost[6] := epa533^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
427         (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
428       inc(TotEval);
429       if _CalcCost[6] > 0 then begin
430         _Cost[6] := 1 * (numb_initial_samples[P.SourceWater, P.SystemSize] * hrs_samp[P.SourceWater, P.SystemSize] *
431           labor_sys_rate[P.SourceWater, P.SystemSize] + numb_initial_samples[P.SourceWater, P.SystemSize] *
432           epa533_cost[P.SourceWater, P.SystemSize]);
433         for DRIdx := 0 to _HighDRIdx do begin
434           v := _Cost[6] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
435           _i(_pCost[DRIdx, 6]^, v);
436           _i(_pCostY[Yr, DRIdx, 6]^, v);
437           v := _Cost[6] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
438           _i(_pCost_pws[DRIdx, 6]^, v);
439           _i(_pCostY_pws[Yr, DRIdx, 6]^, v);
440         end;
441         inc(TotEval);
442
443       end;
444     end;
445     // sys_initial_monitor_EPA537  row: 9
446     if (_YearOK[sz, Yr, 7]) then begin
447       _CalcCost[7] := epa537^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
448         (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
449       inc(TotEval);
450       if _CalcCost[7] > 0 then begin
451         _Cost[7] := 1 * (numb_initial_samples[P.SourceWater, P.SystemSize] * hrs_samp[P.SourceWater, P.SystemSize] *
452           labor_sys_rate[P.SourceWater, P.SystemSize] + numb_initial_samples[P.SourceWater, P.SystemSize] *
453           epa537_cost[P.SourceWater, P.SystemSize]);
454         for DRIdx := 0 to _HighDRIdx do begin
455           v := _Cost[7] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
456           _i(_pCost[DRIdx, 7]^, v);
457           _i(_pCostY[Yr, DRIdx, 7]^, v);
458           v := _Cost[7] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
459           _i(_pCost_pws[DRIdx, 7]^, v);
460           _i(_pCostY_pws[Yr, DRIdx, 7]^, v);
461         end;
462         inc(TotEval);
463
464       end;
465     end;
466     // sys_initial_fieldblank_EPA533  row: 10
467     if (_YearOK[sz, Yr, 8]) then begin
468       _CalcCost[8] := epa533^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
469         (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]) * (1 - b_nondetect^);
470       inc(TotEval);
471       if _CalcCost[8] > 0 then begin
472         _Cost[8] := 1 * (numb_initial_samples[P.SourceWater, P.SystemSize] * epa533_fieldblank_cost[P.SourceWater,
473           P.SystemSize]);
474         for DRIdx := 0 to _HighDRIdx do begin
475           v := _Cost[8] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
476           _i(_pCost[DRIdx, 8]^, v);
477           _i(_pCostY[Yr, DRIdx, 8]^, v);
478           v := _Cost[8] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
479           _i(_pCost_pws[DRIdx, 8]^, v);
480           _i(_pCostY_pws[Yr, DRIdx, 8]^, v);
481         end;
482         inc(TotEval);
483
484       end;
485     end;
486     // sys_initial_fieldblank_EPA537  row: 11
487     if (_YearOK[sz, Yr, 9]) then begin
488       _CalcCost[9] := epa537^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
489         (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]) * (1 - b_nondetect^);
490       inc(TotEval);
491       if _CalcCost[9] > 0 then begin
492         _Cost[9] := 1 * (numb_initial_samples[P.SourceWater, P.SystemSize] * epa537_fieldblank_cost[P.SourceWater,
493           P.SystemSize]);
494         for DRIdx := 0 to _HighDRIdx do begin
495           v := _Cost[9] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
```

```
496          _i(_pCost[DRIdx, 9]^, v);
497          _i(_pCostY[Yr, DRIdx, 9]^, v);
498          v := _Cost[9] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
499          _i(_pCost_pws[DRIdx, 9]^, v);
500          _i(_pCostY_pws[Yr, DRIdx, 9]^, v);
501        end;
502        inc(TotEval);
503
504      end;
505    end;
506    // sys_report_initial_monitor  row: 12
507    if (_YearOK[sz, Yr, 10]) then begin
508      _CalcCost[10] := (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
509        (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
510      inc(TotEval);
511      if _CalcCost[10] > 0 then begin
512        _Cost[10] := 1 * (numb_initial_samples[P.SourceWater, P.SystemSize] * hrs_sys_report_ep[P.SourceWater,
513          P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
514        for DRIdx := 0 to _HighDRIdx do begin
515          v := _Cost[10] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
516          _i(_pCost[DRIdx, 10]^, v);
517          _i(_pCostY[Yr, DRIdx, 10]^, v);
518          v := _Cost[10] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
519          _i(_pCost_pws[DRIdx, 10]^, v);
520          _i(_pCostY_pws[Yr, DRIdx, 10]^, v);
521        end;
522        inc(TotEval);
523
524      end;
525    end;
526    // pa_review_initial_monitor  row: 13
527    if (_YearOK[sz, Yr, 11]) then begin
528      _CalcCost[11] := (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
529        (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
530      inc(TotEval);
531      if _CalcCost[11] > 0 then begin
532        _Cost[11] := 1 * (numb_initial_samples[P.SourceWater, P.SystemSize] * hrs_pa_report_ep[P.SourceWater,
533          P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
534        for DRIdx := 0 to _HighDRIdx do begin
535          v := _Cost[11] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
536          _i(_pCost[DRIdx, 11]^, v);
537          _i(_pCostY[Yr, DRIdx, 11]^, v);
538          v := _Cost[11] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
539          _i(_pCost_pws[DRIdx, 11]^, v);
540          _i(_pCostY_pws[Yr, DRIdx, 11]^, v);
541        end;
542        inc(TotEval);
543
544      end;
545    end;
546    // sys_initial_confirm_EPA533  row: 14
547    if (_YearOK[sz, Yr, 12]) then begin
548      _CalcCost[12] := epa533^ * b_exceed^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater,
549        P.SystemSize]) * (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
550      inc(TotEval);
551      if _CalcCost[12] > 0 then begin
552        _Cost[12] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
553          epa533_cost[P.SourceWater, P.SystemSize]);
554        for DRIdx := 0 to _HighDRIdx do begin
555          v := _Cost[12] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
556          _i(_pCost[DRIdx, 12]^, v);
557          _i(_pCostY[Yr, DRIdx, 12]^, v);
558          v := _Cost[12] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
559          _i(_pCost_pws[DRIdx, 12]^, v);
560          _i(_pCostY_pws[Yr, DRIdx, 12]^, v);
561        end;
562        inc(TotEval);
563
564      end;
565    end;
566    // sys_initial_confirm_EPA537  row: 15
567    if (_YearOK[sz, Yr, 13]) then begin
568      _CalcCost[13] := epa537^ * b_exceed^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater,
569        P.SystemSize]) * (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
```

```
570       inc(TotEval);
571       if _CalcCost[13] > 0 then begin
572         _Cost[13] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
573           epa537_cost[P.SourceWater, P.SystemSize]);
574         for DRIdx := 0 to _HighDRIdx do begin
575           v := _Cost[13] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
576           _i(_pCost[DRIdx, 13]^, v);
577           _i(_pCostY[Yr, DRIdx, 13]^, v);
578           v := _Cost[13] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
579           _i(_pCost_pws[DRIdx, 13]^, v);
580           _i(_pCostY_pws[Yr, DRIdx, 13]^, v);
581         end;
582         inc(TotEval);
583
584       end;
585     end;
586     // sys_initial_confirm_fieldblank_EPA533  row: 16
587     if (_YearOK[sz, Yr, 14]) then begin
588       _CalcCost[14] := epa533^ * b_exceed^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater,
589         P.SystemSize]) * (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
590       inc(TotEval);
591       if _CalcCost[14] > 0 then begin
592         _Cost[14] := 1 * (epa533_fieldblank_cost[P.SourceWater, P.SystemSize]);
593         for DRIdx := 0 to _HighDRIdx do begin
594           v := _Cost[14] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
595           _i(_pCost[DRIdx, 14]^, v);
596           _i(_pCostY[Yr, DRIdx, 14]^, v);
597           v := _Cost[14] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
598           _i(_pCost_pws[DRIdx, 14]^, v);
599           _i(_pCostY_pws[Yr, DRIdx, 14]^, v);
600         end;
601         inc(TotEval);
602
603       end;
604     end;
605     // sys_initial_confirm_fieldblank_EPA537  row: 17
606     if (_YearOK[sz, Yr, 15]) then begin
607       _CalcCost[15] := epa537^ * b_exceed^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater,
608         P.SystemSize]) * (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
609       inc(TotEval);
610       if _CalcCost[15] > 0 then begin
611         _Cost[15] := 1 * (epa537_fieldblank_cost[P.SourceWater, P.SystemSize]);
612         for DRIdx := 0 to _HighDRIdx do begin
613           v := _Cost[15] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
614           _i(_pCost[DRIdx, 15]^, v);
615           _i(_pCostY[Yr, DRIdx, 15]^, v);
616           v := _Cost[15] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
617           _i(_pCost_pws[DRIdx, 15]^, v);
618           _i(_pCostY_pws[Yr, DRIdx, 15]^, v);
619         end;
620         inc(TotEval);
621
622       end;
623     end;
624     // sys_report_confirm_monitor  row: 18
625     if (_YearOK[sz, Yr, 16]) then begin
626       _CalcCost[16] := b_exceed^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
627         (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
628       inc(TotEval);
629       if _CalcCost[16] > 0 then begin
630         _Cost[16] := 1 * (hrs_sys_report_ep[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
631         for DRIdx := 0 to _HighDRIdx do begin
632           v := _Cost[16] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
633           _i(_pCost[DRIdx, 16]^, v);
634           _i(_pCostY[Yr, DRIdx, 16]^, v);
635           v := _Cost[16] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
636           _i(_pCost_pws[DRIdx, 16]^, v);
637           _i(_pCostY_pws[Yr, DRIdx, 16]^, v);
638         end;
639         inc(TotEval);
640
641       end;
642     end;
643     // pa_review_confirm_monitor  row: 19
644     if (_YearOK[sz, Yr, 17]) then begin
```

```
645       _CalcCost[17] := b_exceed^ * (1 - ucmr5[P.SourceWater, P.SystemSize] * small_flex1[P.SourceWater, P.SystemSize]) *
646         (1 - statepfas^ * small_flex1[P.SourceWater, P.SystemSize]);
647       inc(TotEval);
648       if _CalcCost[17] > 0 then begin
649         _Cost[17] := 1 * (hrs_pa_report_ep[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
650         for DRIdx := 0 to _HighDRIdx do begin
651           v := _Cost[17] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
652           _i(_pCost[DRIdx, 17]^, v);
653           _i(_pCostY[Yr, DRIdx, 17]^, v);
654           v := _Cost[17] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
655           _i(_pCost_pws[DRIdx, 17]^, v);
656           _i(_pCostY_pws[Yr, DRIdx, 17]^, v);
657         end;
658         inc(TotEval);
659
660       end;
661     end;
662     // sys_stand_monitor_triennial_EPA533  row: 20
663     if (_YearOK[sz, Yr, 18]) then begin
664       _CalcCost[18] := epa533^ * (b_triennial^);
665       inc(TotEval);
666       if _CalcCost[18] > 0 then begin
667         _Cost[18] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * numb_nondetect_samples[P.SourceWater, P.SystemSize] *
668           labor_sys_rate[P.SourceWater, P.SystemSize] + epa533_cost[P.SourceWater, P.SystemSize] *
669           numb_nondetect_samples[P.SourceWater, P.SystemSize]);
670         for DRIdx := 0 to _HighDRIdx do begin
671           v := _Cost[18] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
672           _i(_pCost[DRIdx, 18]^, v);
673           _i(_pCostY[Yr, DRIdx, 18]^, v);
674           v := _Cost[18] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
675           _i(_pCost_pws[DRIdx, 18]^, v);
676           _i(_pCostY_pws[Yr, DRIdx, 18]^, v);
677         end;
678         inc(TotEval);
679
680       end;
681     end;
682     // sys_stand_monitor_triennial_EPA537  row: 21
683     if (_YearOK[sz, Yr, 19]) then begin
684       _CalcCost[19] := epa537^ * (b_triennial^);
685       inc(TotEval);
686       if _CalcCost[19] > 0 then begin
687         _Cost[19] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * numb_nondetect_samples[P.SourceWater, P.SystemSize] *
688           labor_sys_rate[P.SourceWater, P.SystemSize] + epa537_cost[P.SourceWater, P.SystemSize] *
689           numb_nondetect_samples[P.SourceWater, P.SystemSize]);
690         for DRIdx := 0 to _HighDRIdx do begin
691           v := _Cost[19] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
692           _i(_pCost[DRIdx, 19]^, v);
693           _i(_pCostY[Yr, DRIdx, 19]^, v);
694           v := _Cost[19] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
695           _i(_pCost_pws[DRIdx, 19]^, v);
696           _i(_pCostY_pws[Yr, DRIdx, 19]^, v);
697         end;
698         inc(TotEval);
699
700       end;
701     end;
702     // sys_triennial_fieldblank_EPA533  row: 22
703     if (_YearOK[sz, Yr, 20]) then begin
704       _CalcCost[20] := epa533^ * (b_triennial^) * (1 - b_nondetect^);
705       inc(TotEval);
706       if _CalcCost[20] > 0 then begin
707         _Cost[20] := 1 * (epa533_fieldblank_cost[P.SourceWater, P.SystemSize] * numb_nondetect_samples[P.SourceWater,
708           P.SystemSize]);
709         for DRIdx := 0 to _HighDRIdx do begin
710           v := _Cost[20] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
711           _i(_pCost[DRIdx, 20]^, v);
712           _i(_pCostY[Yr, DRIdx, 20]^, v);
713           v := _Cost[20] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
714           _i(_pCost_pws[DRIdx, 20]^, v);
715           _i(_pCostY_pws[Yr, DRIdx, 20]^, v);
716         end;
717         inc(TotEval);
718
```

```
719         end;
720       end;
721       // sys_triennial_fieldblank_EPA537  row: 23
722       if (_YearOK[sz, Yr, 21]) then begin
723         _CalcCost[21] := epa537^ * (b_triennial^) * (1 - b_nondetect^);
724         inc(TotEval);
725         if _CalcCost[21] > 0 then begin
726           _Cost[21] := 1 * (epa537_fieldblank_cost[P.SourceWater, P.SystemSize] * numb_nondetect_samples[P.SourceWater,
727             P.SystemSize]);
728           for DRIdx := 0 to _HighDRIdx do begin
729             v := _Cost[21] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
730             _i(_pCost[DRIdx, 21]^, v);
731             _i(_pCostY[Yr, DRIdx, 21]^, v);
732             v := _Cost[21] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
733             _i(_pCost_pws[DRIdx, 21]^, v);
734             _i(_pCostY_pws[Yr, DRIdx, 21]^, v);
735           end;
736           inc(TotEval);
737
738         end;
739       end;
740       // sys_report_stand_monitor_triennial  row: 24
741       if (_YearOK[sz, Yr, 22]) then begin
742         _CalcCost[22] := b_triennial^;
743         inc(TotEval);
744         if _CalcCost[22] > 0 then begin
745           _Cost[22] := 1 * (hrs_sys_report_ep[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
746           for DRIdx := 0 to _HighDRIdx do begin
747             v := _Cost[22] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
748             _i(_pCost[DRIdx, 22]^, v);
749             _i(_pCostY[Yr, DRIdx, 22]^, v);
750             v := _Cost[22] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
751             _i(_pCost_pws[DRIdx, 22]^, v);
752             _i(_pCostY_pws[Yr, DRIdx, 22]^, v);
753           end;
754           inc(TotEval);
755
756         end;
757       end;
758       // pa_review_stand_monitor_triennial  row: 25
759       if (_YearOK[sz, Yr, 23]) then begin
760         _CalcCost[23] := b_triennial^;
761         inc(TotEval);
762         if _CalcCost[23] > 0 then begin
763           _Cost[23] := 1 * (hrs_pa_report_ep[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
764           for DRIdx := 0 to _HighDRIdx do begin
765             v := _Cost[23] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
766             _i(_pCost[DRIdx, 23]^, v);
767             _i(_pCostY[Yr, DRIdx, 23]^, v);
768             v := _Cost[23] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
769             _i(_pCost_pws[DRIdx, 23]^, v);
770             _i(_pCostY_pws[Yr, DRIdx, 23]^, v);
771           end;
772           inc(TotEval);
773
774         end;
775       end;
776       // sys_stand_monitor_detect_EPA533  row: 26
777       if (_YearOK[sz, Yr, 24]) then begin
778         _CalcCost[24] := epa533^ * b_annual^;
779         inc(TotEval);
780         if _CalcCost[24] > 0 then begin
781           _Cost[24] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
782             epa533_cost[P.SourceWater, P.SystemSize]);
783           for DRIdx := 0 to _HighDRIdx do begin
784             v := _Cost[24] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
785             _i(_pCost[DRIdx, 24]^, v);
786             _i(_pCostY[Yr, DRIdx, 24]^, v);
787             v := _Cost[24] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
788             _i(_pCost_pws[DRIdx, 24]^, v);
789             _i(_pCostY_pws[Yr, DRIdx, 24]^, v);
790           end;
791           inc(TotEval);
792
793         end;
```

```
794     end;
795     // sys_stand_monitor_detect_EPA537  row: 27
796     if (_YearOK[sz, Yr, 25]) then begin
797       _CalcCost[25] := epa537^ * b_annual^;
798       inc(TotEval);
799       if _CalcCost[25] > 0 then begin
800         _Cost[25] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
801           epa537_cost[P.SourceWater, P.SystemSize]);
802         for DRIdx := 0 to _HighDRIdx do begin
803           v := _Cost[25] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
804           _i(_pCost[DRIdx, 25]^, v);
805           _i(_pCostY[Yr, DRIdx, 25]^, v);
806           v := _Cost[25] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
807           _i(_pCost_pws[DRIdx, 25]^, v);
808           _i(_pCostY_pws[Yr, DRIdx, 25]^, v);
809         end;
810         inc(TotEval);
811
812       end;
813     end;
814     // sys_stand_monitor_fieldblank_EPA533  row: 28
815     if (_YearOK[sz, Yr, 26]) then begin
816       _CalcCost[26] := epa533^ * b_annual^ * (1 - b_nondetect^);
817       inc(TotEval);
818       if _CalcCost[26] > 0 then begin
819         _Cost[26] := 1 * (epa533_fieldblank_cost[P.SourceWater, P.SystemSize]);
820         for DRIdx := 0 to _HighDRIdx do begin
821           v := _Cost[26] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
822           _i(_pCost[DRIdx, 26]^, v);
823           _i(_pCostY[Yr, DRIdx, 26]^, v);
824           v := _Cost[26] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
825           _i(_pCost_pws[DRIdx, 26]^, v);
826           _i(_pCostY_pws[Yr, DRIdx, 26]^, v);
827         end;
828         inc(TotEval);
829
830       end;
831     end;
832     // sys_stand_monitor_fieldblank_EPA537  row: 29
833     if (_YearOK[sz, Yr, 27]) then begin
834       _CalcCost[27] := epa537^ * b_annual^ * (1 - b_nondetect^);
835       inc(TotEval);
836       if _CalcCost[27] > 0 then begin
837         _Cost[27] := 1 * (epa537_fieldblank_cost[P.SourceWater, P.SystemSize]);
838         for DRIdx := 0 to _HighDRIdx do begin
839           v := _Cost[27] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
840           _i(_pCost[DRIdx, 27]^, v);
841           _i(_pCostY[Yr, DRIdx, 27]^, v);
842           v := _Cost[27] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
843           _i(_pCost_pws[DRIdx, 27]^, v);
844           _i(_pCostY_pws[Yr, DRIdx, 27]^, v);
845         end;
846         inc(TotEval);
847
848       end;
849     end;
850     // sys_report_stand_monitor_detect  row: 30
851     if (_YearOK[sz, Yr, 28]) then begin
852       _CalcCost[28] := b_annual^;
853       inc(TotEval);
854       if _CalcCost[28] > 0 then begin
855         _Cost[28] := 1 * (hrs_sys_report_ep[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
856         for DRIdx := 0 to _HighDRIdx do begin
857           v := _Cost[28] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
858           _i(_pCost[DRIdx, 28]^, v);
859           _i(_pCostY[Yr, DRIdx, 28]^, v);
860           v := _Cost[28] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
861           _i(_pCost_pws[DRIdx, 28]^, v);
862           _i(_pCostY_pws[Yr, DRIdx, 28]^, v);
863         end;
864         inc(TotEval);
865
866       end;
867     end;
```

```
868     // pa_review_stand_monitor_detect  row: 31
869     if (_YearOK[sz, Yr, 29]) then begin
870       _CalcCost[29] := b_annual^;
871       inc(TotEval);
872       if _CalcCost[29] > 0 then begin
873         _Cost[29] := 1 * (hrs_pa_report_ep[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
874         for DRIdx := 0 to _HighDRIdx do begin
875           v := _Cost[29] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
876           _i(_pCost[DRIdx, 29]^, v);
877           _i(_pCostY[Yr, DRIdx, 29]^, v);
878           v := _Cost[29] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
879           _i(_pCost_pws[DRIdx, 29]^, v);
880           _i(_pCostY_pws[Yr, DRIdx, 29]^, v);
881         end;
882         inc(TotEval);
883
884       end;
885     end;
886     // sys_stand_monitor_exceed_EPA533  row: 32
887     if (_YearOK[sz, Yr, 30]) then begin
888       _CalcCost[30] := epa533^ * b_quarterly^;
889       inc(TotEval);
890       if _CalcCost[30] > 0 then begin
891         _Cost[30] := 4 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
892           epa533_cost[P.SourceWater, P.SystemSize]);
893         for DRIdx := 0 to _HighDRIdx do begin
894           v := _Cost[30] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
895           _i(_pCost[DRIdx, 30]^, v);
896           _i(_pCostY[Yr, DRIdx, 30]^, v);
897           v := _Cost[30] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
898           _i(_pCost_pws[DRIdx, 30]^, v);
899           _i(_pCostY_pws[Yr, DRIdx, 30]^, v);
900         end;
901         inc(TotEval);
902
903       end;
904     end;
905     // sys_stand_monitor_exceed_EPA537  row: 33
906     if (_YearOK[sz, Yr, 31]) then begin
907       _CalcCost[31] := epa537^ * b_quarterly^;
908       inc(TotEval);
909       if _CalcCost[31] > 0 then begin
910         _Cost[31] := 4 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
911           epa537_cost[P.SourceWater, P.SystemSize]);
912         for DRIdx := 0 to _HighDRIdx do begin
913           v := _Cost[31] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
914           _i(_pCost[DRIdx, 31]^, v);
915           _i(_pCostY[Yr, DRIdx, 31]^, v);
916           v := _Cost[31] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
917           _i(_pCost_pws[DRIdx, 31]^, v);
918           _i(_pCostY_pws[Yr, DRIdx, 31]^, v);
919         end;
920         inc(TotEval);
921
922       end;
923     end;
924     // sys_stand_monitor_exceed_fieldblank_EPA533  row: 34
925     if (_YearOK[sz, Yr, 32]) then begin
926       _CalcCost[32] := epa533^ * b_quarterly^ * (1 - b_nondetect^);
927       inc(TotEval);
928       if _CalcCost[32] > 0 then begin
929         _Cost[32] := 4 * (epa533_fieldblank_cost[P.SourceWater, P.SystemSize]);
930         for DRIdx := 0 to _HighDRIdx do begin
931           v := _Cost[32] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
932           _i(_pCost[DRIdx, 32]^, v);
933           _i(_pCostY[Yr, DRIdx, 32]^, v);
934           v := _Cost[32] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
935           _i(_pCost_pws[DRIdx, 32]^, v);
936           _i(_pCostY_pws[Yr, DRIdx, 32]^, v);
937         end;
938         inc(TotEval);
939
940       end;
941     end;
942     // sys_stand_monitor_exceed_fieldblank_EPA537  row: 35
```

```
943     if (_YearOK[sz, Yr, 33]) then begin
944       _CalcCost[33] := epa537^ * b_quarterly^ * (1 - b_nondetect^);
945       inc(TotEval);
946       if _CalcCost[33] > 0 then begin
947         _Cost[33] := 4 * (epa537_fieldblank_cost[P.SourceWater, P.SystemSize]);
948         for DRIdx := 0 to _HighDRIdx do begin
949           v := _Cost[33] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
950           _i(_pCost[DRIdx, 33]^, v);
951           _i(_pCostY[Yr, DRIdx, 33]^, v);
952           v := _Cost[33] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
953           _i(_pCost_pws[DRIdx, 33]^, v);
954           _i(_pCostY_pws[Yr, DRIdx, 33]^, v);
955         end;
956         inc(TotEval);
957
958       end;
959     end;
960     // sys_report_stand_monitor_exceed  row: 36
961     if (_YearOK[sz, Yr, 34]) then begin
962       _CalcCost[34] := b_quarterly^;
963       inc(TotEval);
964       if _CalcCost[34] > 0 then begin
965         _Cost[34] := 4 * (hrs_sys_report_ep[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
966         for DRIdx := 0 to _HighDRIdx do begin
967           v := _Cost[34] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
968           _i(_pCost[DRIdx, 34]^, v);
969           _i(_pCostY[Yr, DRIdx, 34]^, v);
970           v := _Cost[34] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
971           _i(_pCost_pws[DRIdx, 34]^, v);
972           _i(_pCostY_pws[Yr, DRIdx, 34]^, v);
973         end;
974         inc(TotEval);
975
976       end;
977     end;
978     // pa_review_stand_monitor_exceed  row: 37
979     if (_YearOK[sz, Yr, 35]) then begin
980       _CalcCost[35] := b_quarterly^;
981       inc(TotEval);
982       if _CalcCost[35] > 0 then begin
983         _Cost[35] := 4 * (hrs_pa_report_ep[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
984         for DRIdx := 0 to _HighDRIdx do begin
985           v := _Cost[35] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
986           _i(_pCost[DRIdx, 35]^, v);
987           _i(_pCostY[Yr, DRIdx, 35]^, v);
988           v := _Cost[35] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
989           _i(_pCost_pws[DRIdx, 35]^, v);
990           _i(_pCostY_pws[Yr, DRIdx, 35]^, v);
991         end;
992         inc(TotEval);
993
994       end;
995     end;
996     // sys_waiver_sample_EPA533  row: 40
997     if (_YearOK[sz, Yr, 38]) then begin
998       _CalcCost[38] := 0;
999       inc(TotEval);
1000      if _CalcCost[38] > 0 then begin
1001        _Cost[38] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
1002          epa533_cost[P.SourceWater, P.SystemSize]);
1003        for DRIdx := 0 to _HighDRIdx do begin
1004          v := _Cost[38] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1005          _i(_pCost[DRIdx, 38]^, v);
1006          _i(_pCostY[Yr, DRIdx, 38]^, v);
1007          v := _Cost[38] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1008          _i(_pCost_pws[DRIdx, 38]^, v);
1009          _i(_pCostY_pws[Yr, DRIdx, 38]^, v);
1010        end;
1011        inc(TotEval);
1012
1013      end;
1014    end;
1015    // sys_waiver_sample_EPA537  row: 41
1016    if (_YearOK[sz, Yr, 39]) then begin
```

```
1017        _CalcCost[39] := 0;
1018        inc(TotEval);
1019        if _CalcCost[39] > 0 then begin
1020          _Cost[39] := 1 * (hrs_samp[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize] +
1021            epa537_cost[P.SourceWater, P.SystemSize]);
1022          for DRIdx := 0 to _HighDRIdx do begin
1023            v := _Cost[39] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1024            _i(_pCost[DRIdx, 39]^, v);
1025            _i(_pCostY[Yr, DRIdx, 39]^, v);
1026            v := _Cost[39] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1027            _i(_pCost_pws[DRIdx, 39]^, v);
1028            _i(_pCostY_pws[Yr, DRIdx, 39]^, v);
1029          end;
1030          inc(TotEval);
1031
1032        end;
1033      end;
1034      // sys_waiver_sample_report  row: 42
1035      if (_YearOK[sz, Yr, 40]) then begin
1036        _CalcCost[40] := 0;
1037        inc(TotEval);
1038        if _CalcCost[40] > 0 then begin
1039          _Cost[40] := 1 * (hrs_sys_report_ep[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1040          for DRIdx := 0 to _HighDRIdx do begin
1041            v := _Cost[40] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1042            _i(_pCost[DRIdx, 40]^, v);
1043            _i(_pCostY[Yr, DRIdx, 40]^, v);
1044            v := _Cost[40] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1045            _i(_pCost_pws[DRIdx, 40]^, v);
1046            _i(_pCostY_pws[Yr, DRIdx, 40]^, v);
1047          end;
1048          inc(TotEval);
1049
1050        end;
1051      end;
1052      // pa_waiver_sample_report  row: 43
1053      if (_YearOK[sz, Yr, 41]) then begin
1054        _CalcCost[41] := 0;
1055        inc(TotEval);
1056        if _CalcCost[41] > 0 then begin
1057          _Cost[41] := 1 * (hrs_pa_report_ep[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1058          for DRIdx := 0 to _HighDRIdx do begin
1059            v := _Cost[41] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1060            _i(_pCost[DRIdx, 41]^, v);
1061            _i(_pCostY[Yr, DRIdx, 41]^, v);
1062            v := _Cost[41] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1063            _i(_pCost_pws[DRIdx, 41]^, v);
1064            _i(_pCostY_pws[Yr, DRIdx, 41]^, v);
1065          end;
1066          inc(TotEval);
1067
1068        end;
1069      end;
1070      // sys_permit_application_treat  row: 44
1071      if (_YearOK[sz, Yr, 42]) then begin
1072        _CalcCost[42] := b_exceed^ * b_treat^;
1073        inc(TotEval);
1074        if _CalcCost[42] > 0 then begin
1075          _Cost[42] := 1 * (hrs_sys_treat[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1076          for DRIdx := 0 to _HighDRIdx do begin
1077            v := _Cost[42] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1078            _i(_pCost[DRIdx, 42]^, v);
1079            _i(_pCostY[Yr, DRIdx, 42]^, v);
1080            v := _Cost[42] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1081            _i(_pCost_pws[DRIdx, 42]^, v);
1082            _i(_pCostY_pws[Yr, DRIdx, 42]^, v);
1083          end;
1084          inc(TotEval);
1085
1086        end;
1087      end;
1088      // pa_permit_application_treat  row: 45
1089      if (_YearOK[sz, Yr, 43]) then begin
1090        _CalcCost[43] := b_exceed^ * b_treat^;
1091        inc(TotEval);
```

```
1092        if _CalcCost[43] > 0 then begin
1093          _Cost[43] := 1 * (hrs_pa_treat[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1094          for DRIdx := 0 to _HighDRIdx do begin
1095            v := _Cost[43] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1096            _i(_pCost[DRIdx, 43]^, v);
1097            _i(_pCostY[Yr, DRIdx, 43]^, v);
1098            v := _Cost[43] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1099            _i(_pCost_pws[DRIdx, 43]^, v);
1100            _i(_pCostY_pws[Yr, DRIdx, 43]^, v);
1101          end;
1102          inc(TotEval);
1103
1104        end;
1105      end;
1106      // sys_permit_application_source  row: 47
1107      if (_YearOK[sz, Yr, 44]) then begin
1108        _CalcCost[44] := b_exceed^ * b_source^;
1109        inc(TotEval);
1110        if _CalcCost[44] > 0 then begin
1111          _Cost[44] := 1 * (hrs_sys_source[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1112          for DRIdx := 0 to _HighDRIdx do begin
1113            v := _Cost[44] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1114            _i(_pCost[DRIdx, 44]^, v);
1115            _i(_pCostY[Yr, DRIdx, 44]^, v);
1116            v := _Cost[44] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1117            _i(_pCost_pws[DRIdx, 44]^, v);
1118            _i(_pCostY_pws[Yr, DRIdx, 44]^, v);
1119          end;
1120          inc(TotEval);
1121
1122        end;
1123      end;
1124      // pa_permit_application_source  row: 48
1125      if (_YearOK[sz, Yr, 45]) then begin
1126        _CalcCost[45] := b_exceed^ * b_source^;
1127        inc(TotEval);
1128        if _CalcCost[45] > 0 then begin
1129          _Cost[45] := 1 * (hrs_pa_source[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1130          for DRIdx := 0 to _HighDRIdx do begin
1131            v := _Cost[45] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1132            _i(_pCost[DRIdx, 45]^, v);
1133            _i(_pCostY[Yr, DRIdx, 45]^, v);
1134            v := _Cost[45] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1135            _i(_pCost_pws[DRIdx, 45]^, v);
1136            _i(_pCostY_pws[Yr, DRIdx, 45]^, v);
1137          end;
1138          inc(TotEval);
1139
1140        end;
1141      end;
1142      // pws_distribute_tier2  row: 53
1143      if (_YearOK[sz, Yr, 49]) then begin
1144        _CalcCost[49] := 0;
1145        inc(TotEval);
1146        if _CalcCost[49] > 0 then begin
1147          _Cost[49] := 4 * (numb_hh[P.SourceWater, P.SystemSize] * hrs_tier2_distribute[P.SourceWater, P.SystemSize] *
1148            labor_sys_rate[P.SourceWater, P.SystemSize] + numb_hh[P.SourceWater, P.SystemSize] * tier2_cost[P.SourceWater,
1149            P.SystemSize]);
1150          for DRIdx := 0 to _HighDRIdx do begin
1151            v := _Cost[49] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1152            _i(_pCost[DRIdx, 49]^, v);
1153            _i(_pCostY[Yr, DRIdx, 49]^, v);
1154            v := _Cost[49] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1155            _i(_pCost_pws[DRIdx, 49]^, v);
1156            _i(_pCostY_pws[Yr, DRIdx, 49]^, v);
1157          end;
1158          inc(TotEval);
1159
1160        end;
1161      end;
1162    end;
1163  end;
1164
1165  procedure TCCCompiledCostOption._EvaluateSystem(const P: TPWSRecord; const Yr: integer; const Data: PSWModelInputData;
```

```
1166   const PeriodAdjust: integer);
1167 begin
1168   _Reset(true);
1169   // _LoadVars();
1170   var
1171   sz := 0;
1172   var v: double;
1173   var DRIdx: integer;
1174   if P.SystemSize >= ss3301_10k then
1175     sz := 1;
1176
1177   with _Variables do begin
1178
1179     // system_read  row: 2
1180     if (_YearOK[sz, Yr, 0]) then begin
1181       _Cost[0] := 1 * (hrs_sys_adopt_rule[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1182       for DRIdx := 0 to _HighDRIdx do begin
1183         v := _Cost[0] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1184         _i(_pCost[DRIdx, 0]^, v);
1185         _i(_pCostY[Yr, DRIdx, 0]^, v);
1186         v := _Cost[0] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1187         _i(_pCost_pws[DRIdx, 0]^, v);
1188         _i(_pCostY_pws[Yr, DRIdx, 0]^, v);
1189       end;
1190       inc(TotEval);
1191     end;
1192     // system_implement_train  row: 3
1193     if (_YearOK[sz, Yr, 1]) then begin
1194       _Cost[1] := 1 * (hrs_sys_initial_ta[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1195       for DRIdx := 0 to _HighDRIdx do begin
1196         v := _Cost[1] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1197         _i(_pCost[DRIdx, 1]^, v);
1198         _i(_pCostY[Yr, DRIdx, 1]^, v);
1199         v := _Cost[1] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1200         _i(_pCost_pws[DRIdx, 1]^, v);
1201         _i(_pCostY_pws[Yr, DRIdx, 1]^, v);
1202       end;
1203       inc(TotEval);
1204     end;
1205     // pa_adopt  row: 4
1206     if (_YearOK[sz, Yr, 2]) then begin
1207       _Cost[2] := 1 * (aw^ * (hrs_pa_adopt_rule[P.SourceWater, P.SystemSize] + hrs_pa_write_reg[P.SourceWater,
1208         P.SystemSize]) * labor_pa_rate[P.SourceWater, P.SystemSize]);
1209       for DRIdx := 0 to _HighDRIdx do begin
1210         v := _Cost[2] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1211         _i(_pCost[DRIdx, 2]^, v);
1212         _i(_pCostY[Yr, DRIdx, 2]^, v);
1213         v := _Cost[2] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1214         _i(_pCost_pws[DRIdx, 2]^, v);
1215         _i(_pCostY_pws[Yr, DRIdx, 2]^, v);
1216       end;
1217       inc(TotEval);
1218     end;
1219     // pa_implement_train  row: 5
1220     if (_YearOK[sz, Yr, 3]) then begin
1221       _Cost[3] := 1 * (aw^ * hrs_pa_initial_ta[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater,
1222         P.SystemSize]);
1223       for DRIdx := 0 to _HighDRIdx do begin
1224         v := _Cost[3] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1225         _i(_pCost[DRIdx, 3]^, v);
1226         _i(_pCostY[Yr, DRIdx, 3]^, v);
1227         v := _Cost[3] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1228         _i(_pCost_pws[DRIdx, 3]^, v);
1229         _i(_pCostY_pws[Yr, DRIdx, 3]^, v);
1230       end;
1231       inc(TotEval);
1232     end;
1233     // pa_internal_train  row: 6
1234     if (_YearOK[sz, Yr, 4]) then begin
1235       _Cost[4] := 1 * (aw^ * hrs_pa_train[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1236       for DRIdx := 0 to _HighDRIdx do begin
1237         v := _Cost[4] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1238         _i(_pCost[DRIdx, 4]^, v);
1239         _i(_pCostY[Yr, DRIdx, 4]^, v);
1240         v := _Cost[4] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
```

```
1241        _i(_pCost_pws[DRIdx, 4]^, v);
1242        _i(_pCostY_pws[Yr, DRIdx, 4]^, v);
1243      end;
1244      inc(TotEval);
1245    end;
1246    // pa_sdwis  row: 7
1247    if (_YearOK[sz, Yr, 5]) then begin
1248      _Cost[5] := 1 * (aw^ * hrs_sdwis[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1249      for DRIdx := 0 to _HighDRIdx do begin
1250        v := _Cost[5] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1251        _i(_pCost[DRIdx, 5]^, v);
1252        _i(_pCostY[Yr, DRIdx, 5]^, v);
1253        v := _Cost[5] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1254        _i(_pCost_pws[DRIdx, 5]^, v);
1255        _i(_pCostY_pws[Yr, DRIdx, 5]^, v);
1256      end;
1257      inc(TotEval);
1258    end;
1259    // sys_monitor_waiver  row: 38
1260    if (_YearOK[sz, Yr, 36]) then begin
1261      _CalcCost[36] := 0;
1262      inc(TotEval);
1263      if _CalcCost[36] > 0 then begin
1264        _Cost[36] := 1 * (hrs_sys_waiver[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1265        for DRIdx := 0 to _HighDRIdx do begin
1266          v := _Cost[36] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1267          _i(_pCost[DRIdx, 36]^, v);
1268          _i(_pCostY[Yr, DRIdx, 36]^, v);
1269          v := _Cost[36] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1270          _i(_pCost_pws[DRIdx, 36]^, v);
1271          _i(_pCostY_pws[Yr, DRIdx, 36]^, v);
1272        end;
1273        inc(TotEval);
1274
1275      end;
1276    end;
1277    // pa_monitor_waiver  row: 39
1278    if (_YearOK[sz, Yr, 37]) then begin
1279      _CalcCost[37] := 0;
1280      inc(TotEval);
1281      if _CalcCost[37] > 0 then begin
1282        _Cost[37] := 1 * (hrs_pa_waiver[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1283        for DRIdx := 0 to _HighDRIdx do begin
1284          v := _Cost[37] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1285          _i(_pCost[DRIdx, 37]^, v);
1286          _i(_pCostY[Yr, DRIdx, 37]^, v);
1287          v := _Cost[37] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1288          _i(_pCost_pws[DRIdx, 37]^, v);
1289          _i(_pCostY_pws[Yr, DRIdx, 37]^, v);
1290        end;
1291        inc(TotEval);
1292
1293      end;
1294    end;
1295    // pa_template_tier2  row: 50
1296    if (_YearOK[sz, Yr, 46]) then begin
1297      _CalcCost[46] := 0;
1298      inc(TotEval);
1299      if _CalcCost[46] > 0 then begin
1300        _Cost[46] := 1 * (aw^ * hrs_tier2_template[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater,
1301          P.SystemSize]);
1302        for DRIdx := 0 to _HighDRIdx do begin
1303          v := _Cost[46] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1304          _i(_pCost[DRIdx, 46]^, v);
1305          _i(_pCostY[Yr, DRIdx, 46]^, v);
1306          v := _Cost[46] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1307          _i(_pCost_pws[DRIdx, 46]^, v);
1308          _i(_pCostY_pws[Yr, DRIdx, 46]^, v);
1309        end;
1310        inc(TotEval);
1311
1312      end;
1313    end;
1314    // sys_template_tier2  row: 51
```

```
1315      if (_YearOK[sz, Yr, 47]) then begin
1316        _CalcCost[47] := 0;
1317        inc(TotEval);
1318        if _CalcCost[47] > 0 then begin
1319          _Cost[47] := 1 * (hrs_tier2_template[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater,
1320            P.SystemSize]);
1321          for DRIdx := 0 to _HighDRIdx do begin
1322            v := _Cost[47] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1323            _i(_pCost[DRIdx, 47]^, v);
1324            _i(_pCostY[Yr, DRIdx, 47]^, v);
1325            v := _Cost[47] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1326            _i(_pCost_pws[DRIdx, 47]^, v);
1327            _i(_pCostY_pws[Yr, DRIdx, 47]^, v);
1328          end;
1329          inc(TotEval);
1330
1331        end;
1332      end;
1333      // pa_approve_tier2  row: 52
1334      if (_YearOK[sz, Yr, 48]) then begin
1335        _CalcCost[48] := 0;
1336        inc(TotEval);
1337        if _CalcCost[48] > 0 then begin
1338          _Cost[48] := 1 * (hrs_tier2_approve[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1339          for DRIdx := 0 to _HighDRIdx do begin
1340            v := _Cost[48] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1341            _i(_pCost[DRIdx, 48]^, v);
1342            _i(_pCostY[Yr, DRIdx, 48]^, v);
1343            v := _Cost[48] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1344            _i(_pCost_pws[DRIdx, 48]^, v);
1345            _i(_pCostY_pws[Yr, DRIdx, 48]^, v);
1346          end;
1347          inc(TotEval);
1348
1349        end;
1350      end;
1351      // pws_report_tier2  row: 54
1352      if (_YearOK[sz, Yr, 50]) then begin
1353        _CalcCost[50] := 0;
1354        inc(TotEval);
1355        if _CalcCost[50] > 0 then begin
1356          _Cost[50] := 1 * (hrs_tier2_certify[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1357          for DRIdx := 0 to _HighDRIdx do begin
1358            v := _Cost[50] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1359            _i(_pCost[DRIdx, 50]^, v);
1360            _i(_pCostY[Yr, DRIdx, 50]^, v);
1361            v := _Cost[50] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1362            _i(_pCost_pws[DRIdx, 50]^, v);
1363            _i(_pCostY_pws[Yr, DRIdx, 50]^, v);
1364          end;
1365          inc(TotEval);
1366
1367        end;
1368      end;
1369      // pa_review_tier2  row: 55
1370      if (_YearOK[sz, Yr, 51]) then begin
1371        _CalcCost[51] := 0;
1372        inc(TotEval);
1373        if _CalcCost[51] > 0 then begin
1374          _Cost[51] := 1 * (hrs_tier2_certify[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1375          for DRIdx := 0 to _HighDRIdx do begin
1376            v := _Cost[51] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1377            _i(_pCost[DRIdx, 51]^, v);
1378            _i(_pCostY[Yr, DRIdx, 51]^, v);
1379            v := _Cost[51] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1380            _i(_pCost_pws[DRIdx, 51]^, v);
1381            _i(_pCostY_pws[Yr, DRIdx, 51]^, v);
1382          end;
1383          inc(TotEval);
1384
1385        end;
1386      end;
1387      // pa_template_tier3  row: 56
1388      if (_YearOK[sz, Yr, 52]) then begin
1389        _CalcCost[52] := 0;
```

```
1390        inc(TotEval);
1391        if _CalcCost[52] > 0 then begin
1392          _Cost[52] := 1 * (aw^ * hrs_tier3_template[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater,
1393            P.SystemSize]);
1394          for DRIdx := 0 to _HighDRIdx do begin
1395            v := _Cost[52] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1396            _i(_pCost[DRIdx, 52]^, v);
1397            _i(_pCostY[Yr, DRIdx, 52]^, v);
1398            v := _Cost[52] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1399            _i(_pCost_pws[DRIdx, 52]^, v);
1400            _i(_pCostY_pws[Yr, DRIdx, 52]^, v);
1401          end;
1402          inc(TotEval);
1403
1404        end;
1405      end;
1406      // sys_template_tier3  row: 57
1407      if (_YearOK[sz, Yr, 53]) then begin
1408        _CalcCost[53] := 0;
1409        inc(TotEval);
1410        if _CalcCost[53] > 0 then begin
1411          _Cost[53] := 1 * (hrs_tier3_template[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater,
1412            P.SystemSize]);
1413          for DRIdx := 0 to _HighDRIdx do begin
1414            v := _Cost[53] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1415            _i(_pCost[DRIdx, 53]^, v);
1416            _i(_pCostY[Yr, DRIdx, 53]^, v);
1417            v := _Cost[53] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1418            _i(_pCost_pws[DRIdx, 53]^, v);
1419            _i(_pCostY_pws[Yr, DRIdx, 53]^, v);
1420          end;
1421          inc(TotEval);
1422
1423        end;
1424      end;
1425      // pa_approve_tier3  row: 58
1426      if (_YearOK[sz, Yr, 54]) then begin
1427        _CalcCost[54] := 0;
1428        inc(TotEval);
1429        if _CalcCost[54] > 0 then begin
1430          _Cost[54] := 1 * (hrs_tier3_approve[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1431          for DRIdx := 0 to _HighDRIdx do begin
1432            v := _Cost[54] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1433            _i(_pCost[DRIdx, 54]^, v);
1434            _i(_pCostY[Yr, DRIdx, 54]^, v);
1435            v := _Cost[54] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1436            _i(_pCost_pws[DRIdx, 54]^, v);
1437            _i(_pCostY_pws[Yr, DRIdx, 54]^, v);
1438          end;
1439          inc(TotEval);
1440
1441        end;
1442      end;
1443      // pws_distribute_tier3  row: 59
1444      if (_YearOK[sz, Yr, 55]) then begin
1445        _CalcCost[55] := 0;
1446        inc(TotEval);
1447        if _CalcCost[55] > 0 then begin
1448          _Cost[55] := 1 * (hrs_tier3_distribute[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater,
1449            P.SystemSize] + tier3_cost[P.SourceWater, P.SystemSize]);
1450          for DRIdx := 0 to _HighDRIdx do begin
1451            v := _Cost[55] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1452            _i(_pCost[DRIdx, 55]^, v);
1453            _i(_pCostY[Yr, DRIdx, 55]^, v);
1454            v := _Cost[55] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1455            _i(_pCost_pws[DRIdx, 55]^, v);
1456            _i(_pCostY_pws[Yr, DRIdx, 55]^, v);
1457          end;
1458          inc(TotEval);
1459
1460        end;
1461      end;
1462      // pws_report_tier3  row: 60
1463      if (_YearOK[sz, Yr, 56]) then begin
```

```
1464        _CalcCost[56] := 0;
1465        inc(TotEval);
1466        if _CalcCost[56] > 0 then begin
1467          _Cost[56] := 1 * (hrs_tier3_certify[P.SourceWater, P.SystemSize] * labor_sys_rate[P.SourceWater, P.SystemSize]);
1468          for DRIdx := 0 to _HighDRIdx do begin
1469            v := _Cost[56] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1470            _i(_pCost[DRIdx, 56]^, v);
1471            _i(_pCostY[Yr, DRIdx, 56]^, v);
1472            v := _Cost[56] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1473            _i(_pCost_pws[DRIdx, 56]^, v);
1474            _i(_pCostY_pws[Yr, DRIdx, 56]^, v);
1475          end;
1476          inc(TotEval);
1477
1478        end;
1479      end;
1480      // pa_review_tier3  row: 61
1481      if (_YearOK[sz, Yr, 57]) then begin
1482        _CalcCost[57] := 0;
1483        inc(TotEval);
1484        if _CalcCost[57] > 0 then begin
1485          _Cost[57] := 1 * (hrs_tier3_certify[P.SourceWater, P.SystemSize] * labor_pa_rate[P.SourceWater, P.SystemSize]);
1486          for DRIdx := 0 to _HighDRIdx do begin
1487            v := _Cost[57] / Data^.PreCalcDiscRate[DRIdx, Yr + PeriodAdjust];
1488            _i(_pCost[DRIdx, 57]^, v);
1489            _i(_pCostY[Yr, DRIdx, 57]^, v);
1490            v := _Cost[57] / Data^.PWSPreCalcDiscRate[P.SourceWater, P.SystemSize, DRIdx, Yr + PeriodAdjust];
1491            _i(_pCost_pws[DRIdx, 57]^, v);
1492            _i(_pCostY_pws[Yr, DRIdx, 57]^, v);
1493          end;
1494          inc(TotEval);
1495
1496        end;
1497      end;
1498    end;
1499  end;
1500
1501  procedure TCCCompiledCostOption._SetEPCosts;
1502  begin
1503    for var i := low(_EPCost) to high(_EPCost) do
1504      _EPCost[i] := true;
1505    _EPCost[0] := false;
1506    _EPCost[1] := false;
1507    _EPCost[2] := false;
1508    _EPCost[3] := false;
1509    _EPCost[4] := false;
1510    _EPCost[5] := false;
1511    _EPCost[36] := false;
1512    _EPCost[37] := false;
1513    _EPCost[46] := false;
1514    _EPCost[47] := false;
1515    _EPCost[48] := false;
1516    _EPCost[50] := false;
1517    _EPCost[51] := false;
1518    _EPCost[52] := false;
1519    _EPCost[53] := false;
1520    _EPCost[54] := false;
1521    _EPCost[55] := false;
1522    _EPCost[56] := false;
1523    _EPCost[57] := false;
1524
1525  end;
1526
1527  procedure TCCCompiledCostOption._SetNames;
1528  begin
1529    _Names[0] := 'system_read';
1530    _Cat[0] := 'Administration - System';
1531    _Names[1] := 'system_implement_train';
1532    _Cat[1] := 'Administration - System';
1533    _Names[2] := 'pa_adopt';
1534    _Cat[2] := 'Administration - Primacy Agency';
1535    _Names[3] := 'pa_implement_train';
1536    _Cat[3] := 'Administration - Primacy Agency';
1537    _Names[4] := 'pa_internal_train';
1538    _Cat[4] := 'Administration - Primacy Agency';
```

```
_Names[5] := 'pa_sdwis';
_Cat[5] := 'Administration - Primacy Agency';
_Names[6] := 'sys_initial_monitor_EPA533';
_Cat[6] := 'Monitoring - System';
_Names[7] := 'sys_initial_monitor_EPA537';
_Cat[7] := 'Monitoring - System';
_Names[8] := 'sys_initial_fieldblank_EPA533';
_Cat[8] := 'Monitoring - System';
_Names[9] := 'sys_initial_fieldblank_EPA537';
_Cat[9] := 'Monitoring - System';
_Names[10] := 'sys_report_initial_monitor';
_Cat[10] := 'Monitoring - System';
_Names[11] := 'pa_review_initial_monitor';
_Cat[11] := 'Monitoring - Primacy Agency';
_Names[12] := 'sys_initial_confirm_EPA533';
_Cat[12] := 'Monitoring - System';
_Names[13] := 'sys_initial_confirm_EPA537';
_Cat[13] := 'Monitoring - System';
_Names[14] := 'sys_initial_confirm_fieldblank_EPA533';
_Cat[14] := 'Monitoring - System';
_Names[15] := 'sys_initial_confirm_fieldblank_EPA537';
_Cat[15] := 'Monitoring - System';
_Names[16] := 'sys_report_confirm_monitor';
_Cat[16] := 'Monitoring - System';
_Names[17] := 'pa_review_confirm_monitor';
_Cat[17] := 'Monitoring - Primacy Agency';
_Names[18] := 'sys_stand_monitor_triennial_EPA533';
_Cat[18] := 'Monitoring - System';
_Names[19] := 'sys_stand_monitor_triennial_EPA537';
_Cat[19] := 'Monitoring - System';
_Names[20] := 'sys_triennial_fieldblank_EPA533';
_Cat[20] := 'Monitoring - System';
_Names[21] := 'sys_triennial_fieldblank_EPA537';
_Cat[21] := 'Monitoring - System';
_Names[22] := 'sys_report_stand_monitor_triennial';
_Cat[22] := 'Monitoring - System';
_Names[23] := 'pa_review_stand_monitor_triennial';
_Cat[23] := 'Monitoring - Primacy Agency';
_Names[24] := 'sys_stand_monitor_detect_EPA533';
_Cat[24] := 'Monitoring - System';
_Names[25] := 'sys_stand_monitor_detect_EPA537';
_Cat[25] := 'Monitoring - System';
_Names[26] := 'sys_stand_monitor_fieldblank_EPA533';
_Cat[26] := 'Monitoring - System';
_Names[27] := 'sys_stand_monitor_fieldblank_EPA537';
_Cat[27] := 'Monitoring - System';
_Names[28] := 'sys_report_stand_monitor_detect';
_Cat[28] := 'Monitoring - System';
_Names[29] := 'pa_review_stand_monitor_detect';
_Cat[29] := 'Monitoring - Primacy Agency';
_Names[30] := 'sys_stand_monitor_exceed_EPA533';
_Cat[30] := 'Monitoring - System';
_Names[31] := 'sys_stand_monitor_exceed_EPA537';
_Cat[31] := 'Monitoring - System';
_Names[32] := 'sys_stand_monitor_exceed_fieldblank_EPA533';
_Cat[32] := 'Monitoring - System';
_Names[33] := 'sys_stand_monitor_exceed_fieldblank_EPA537';
_Cat[33] := 'Monitoring - System';
_Names[34] := 'sys_report_stand_monitor_exceed';
_Cat[34] := 'Monitoring - System';
_Names[35] := 'pa_review_stand_monitor_exceed';
_Cat[35] := 'Monitoring - Primacy Agency';
_Names[36] := 'sys_monitor_waiver';
_Cat[36] := 'Monitoring - System';
_Names[37] := 'pa_monitor_waiver';
_Cat[37] := 'Monitoring - Primacy Agency';
_Names[38] := 'sys_waiver_sample_EPA533';
_Cat[38] := 'Monitoring - System';
_Names[39] := 'sys_waiver_sample_EPA537';
_Cat[39] := 'Monitoring - System';
_Names[40] := 'sys_waiver_sample_report';
_Cat[40] := 'Monitoring - System';
_Names[41] := 'pa_waiver_sample_report';
_Cat[41] := 'Monitoring - Primacy Agency';
```

```
1613   _Names[42] := 'sys_permit_application_treat';
1614   _Cat[42] := 'Treatment Administration - System';
1615   _Names[43] := 'pa_permit_application_treat';
1616   _Cat[43] := 'Treatment Administration - Primacy Agency';
1617   _Names[44] := 'sys_permit_application_source';
1618   _Cat[44] := 'Treatment Administration - System';
1619   _Names[45] := 'pa_permit_application_source';
1620   _Cat[45] := 'Treatment Administration - Primacy Agency';
1621   _Names[46] := 'pa_template_tier2';
1622   _Cat[46] := 'Administration - Primacy Agency';
1623   _Names[47] := 'sys_template_tier2';
1624   _Cat[47] := 'Administration - System';
1625   _Names[48] := 'pa_approve_tier2';
1626   _Cat[48] := 'Administration - Primacy Agency';
1627   _Names[49] := 'pws_distribute_tier2';
1628   _Cat[49] := 'Administration - System';
1629   _Names[50] := 'pws_report_tier2';
1630   _Cat[50] := 'Administration - System';
1631   _Names[51] := 'pa_review_tier2';
1632   _Cat[51] := 'Administration - Primacy Agency';
1633   _Names[52] := 'pa_template_tier3';
1634   _Cat[52] := 'Administration - Primacy Agency';
1635   _Names[53] := 'sys_template_tier3';
1636   _Cat[53] := 'Administration - System';
1637   _Names[54] := 'pa_approve_tier3';
1638   _Cat[54] := 'Administration - Primacy Agency';
1639   _Names[55] := 'pws_distribute_tier3';
1640   _Cat[55] := 'Administration - System';
1641   _Names[56] := 'pws_report_tier3';
1642   _Cat[56] := 'Administration - System';
1643   _Names[57] := 'pa_review_tier3';
1644   _Cat[57] := 'Administration - Primacy Agency';
1645
1646 end;
1647
1648 procedure TCCCompiledCostOption._SetVarPointer(const name: string; const pd: pdouble);
1649 var s: string;
1650 begin
1651   s := lowercase(name);
1652   if s = 'aw' then
1653     _Variables.aw := pd
1654   else if s = 'b_annual' then
1655     _Variables.b_annual := pd
1656   else if s = 'b_exceed' then
1657     _Variables.b_exceed := pd
1658   else if s = 'b_nondetect' then
1659     _Variables.b_nondetect := pd
1660   else if s = 'b_quarterly' then
1661     _Variables.b_quarterly := pd
1662   else if s = 'b_source' then
1663     _Variables.b_source := pd
1664   else if s = 'b_treat' then
1665     _Variables.b_treat := pd
1666   else if s = 'b_triennial' then
1667     _Variables.b_triennial := pd
1668   else if s = 'epa533' then
1669     _Variables.epa533 := pd
1670   else if s = 'epa537' then
1671     _Variables.epa537 := pd
1672   else if s = 'statepfas' then
1673     _Variables.statepfas := pd;
1674
1675 end;
1676
1677 procedure TCCCompiledCostOption._SetYears;
1678 begin
1679   // Size, Year, CostID
1680   _YearOK[0, 0, 0] := true;
1681   _YearOK[1, 0, 0] := true;
1682   _YearOK[0, 1, 0] := true;
1683   _YearOK[1, 1, 0] := true;
1684   _YearOK[0, 2, 0] := true;
1685   _YearOK[1, 2, 0] := true;
1686   _YearOK[0, 0, 1] := true;
1687   _YearOK[1, 0, 1] := true;
```

```
_YearOK[0, 1, 1] := true;
_YearOK[1, 1, 1] := true;
_YearOK[0, 2, 1] := true;
_YearOK[1, 2, 1] := true;
_YearOK[0, 0, 2] := true;
_YearOK[1, 0, 2] := true;
_YearOK[0, 1, 2] := true;
_YearOK[1, 1, 2] := true;
_YearOK[0, 2, 2] := true;
_YearOK[1, 2, 2] := true;
_YearOK[0, 0, 3] := true;
_YearOK[1, 0, 3] := true;
_YearOK[0, 1, 3] := true;
_YearOK[1, 1, 3] := true;
_YearOK[0, 2, 3] := true;
_YearOK[1, 2, 3] := true;
_YearOK[0, 0, 4] := true;
_YearOK[1, 0, 4] := true;
_YearOK[0, 1, 4] := true;
_YearOK[1, 1, 4] := true;
_YearOK[0, 2, 4] := true;
_YearOK[1, 2, 4] := true;
_YearOK[0, 0, 5] := true;
_YearOK[1, 0, 5] := true;
_YearOK[0, 1, 5] := true;
_YearOK[1, 1, 5] := true;
_YearOK[0, 2, 5] := true;
_YearOK[1, 2, 5] := true;
_YearOK[0, 3, 5] := true;
_YearOK[1, 3, 5] := true;
_YearOK[0, 4, 5] := true;
_YearOK[1, 4, 5] := true;
_YearOK[0, 5, 5] := true;
_YearOK[1, 5, 5] := true;
_YearOK[0, 6, 5] := true;
_YearOK[1, 6, 5] := true;
_YearOK[0, 7, 5] := true;
_YearOK[1, 7, 5] := true;
_YearOK[0, 8, 5] := true;
_YearOK[1, 8, 5] := true;
_YearOK[0, 9, 5] := true;
_YearOK[1, 9, 5] := true;
_YearOK[0, 10, 5] := true;
_YearOK[1, 10, 5] := true;
_YearOK[0, 11, 5] := true;
_YearOK[1, 11, 5] := true;
_YearOK[0, 12, 5] := true;
_YearOK[1, 12, 5] := true;
_YearOK[0, 13, 5] := true;
_YearOK[1, 13, 5] := true;
_YearOK[0, 14, 5] := true;
_YearOK[1, 14, 5] := true;
_YearOK[0, 15, 5] := true;
_YearOK[1, 15, 5] := true;
_YearOK[0, 16, 5] := true;
_YearOK[1, 16, 5] := true;
_YearOK[0, 17, 5] := true;
_YearOK[1, 17, 5] := true;
_YearOK[0, 18, 5] := true;
_YearOK[1, 18, 5] := true;
_YearOK[0, 19, 5] := true;
_YearOK[1, 19, 5] := true;
_YearOK[0, 20, 5] := true;
_YearOK[1, 20, 5] := true;
_YearOK[0, 21, 5] := true;
_YearOK[1, 21, 5] := true;
_YearOK[0, 22, 5] := true;
_YearOK[1, 22, 5] := true;
_YearOK[0, 23, 5] := true;
_YearOK[1, 23, 5] := true;
_YearOK[0, 24, 5] := true;
_YearOK[1, 24, 5] := true;
_YearOK[0, 25, 5] := true;
_YearOK[1, 25, 5] := true;
```

```
  _YearOK[0, 26, 5] := true;
  _YearOK[1, 26, 5] := true;
  _YearOK[0, 27, 5] := true;
  _YearOK[1, 27, 5] := true;
  _YearOK[0, 28, 5] := true;
  _YearOK[1, 28, 5] := true;
  _YearOK[0, 29, 5] := true;
  _YearOK[1, 29, 5] := true;
  _YearOK[0, 30, 5] := true;
  _YearOK[1, 30, 5] := true;
  _YearOK[0, 31, 5] := true;
  _YearOK[1, 31, 5] := true;
  _YearOK[0, 32, 5] := true;
  _YearOK[1, 32, 5] := true;
  _YearOK[0, 33, 5] := true;
  _YearOK[1, 33, 5] := true;
  _YearOK[0, 34, 5] := true;
  _YearOK[1, 34, 5] := true;
  _YearOK[0, 35, 5] := true;
  _YearOK[1, 35, 5] := true;
  _YearOK[0, 36, 5] := true;
  _YearOK[1, 36, 5] := true;
  _YearOK[0, 37, 5] := true;
  _YearOK[1, 37, 5] := true;
  _YearOK[0, 38, 5] := true;
  _YearOK[1, 38, 5] := true;
  _YearOK[0, 39, 5] := true;
  _YearOK[1, 39, 5] := true;
  _YearOK[0, 40, 5] := true;
  _YearOK[1, 40, 5] := true;
  _YearOK[0, 41, 5] := true;
  _YearOK[1, 41, 5] := true;
  _YearOK[0, 42, 5] := true;
  _YearOK[1, 42, 5] := true;
  _YearOK[0, 43, 5] := true;
  _YearOK[1, 43, 5] := true;
  _YearOK[0, 44, 5] := true;
  _YearOK[1, 44, 5] := true;
  _YearOK[0, 45, 5] := true;
  _YearOK[1, 45, 5] := true;
  _YearOK[0, 46, 5] := true;
  _YearOK[1, 46, 5] := true;
  _YearOK[0, 47, 5] := true;
  _YearOK[1, 47, 5] := true;
  _YearOK[0, 48, 5] := true;
  _YearOK[1, 48, 5] := true;
  _YearOK[0, 49, 5] := true;
  _YearOK[1, 49, 5] := true;
  _YearOK[0, 50, 5] := true;
  _YearOK[1, 50, 5] := true;
  _YearOK[0, 51, 5] := true;
  _YearOK[1, 51, 5] := true;
  _YearOK[0, 52, 5] := true;
  _YearOK[1, 52, 5] := true;
  _YearOK[0, 53, 5] := true;
  _YearOK[1, 53, 5] := true;
  _YearOK[0, 54, 5] := true;
  _YearOK[1, 54, 5] := true;
  _YearOK[0, 55, 5] := true;
  _YearOK[1, 55, 5] := true;
  _YearOK[0, 56, 5] := true;
  _YearOK[1, 56, 5] := true;
  _YearOK[0, 57, 5] := true;
  _YearOK[1, 57, 5] := true;
  _YearOK[0, 58, 5] := true;
  _YearOK[1, 58, 5] := true;
  _YearOK[0, 59, 5] := true;
  _YearOK[1, 59, 5] := true;
  _YearOK[0, 60, 5] := true;
  _YearOK[1, 60, 5] := true;
  _YearOK[0, 61, 5] := true;
  _YearOK[1, 61, 5] := true;
  _YearOK[0, 62, 5] := true;
  _YearOK[1, 62, 5] := true;
  _YearOK[0, 63, 5] := true;
```

```
_YearOK[1, 63, 5] := true;
_YearOK[0, 64, 5] := true;
_YearOK[1, 64, 5] := true;
_YearOK[0, 65, 5] := true;
_YearOK[1, 65, 5] := true;
_YearOK[0, 66, 5] := true;
_YearOK[1, 66, 5] := true;
_YearOK[0, 67, 5] := true;
_YearOK[1, 67, 5] := true;
_YearOK[0, 68, 5] := true;
_YearOK[1, 68, 5] := true;
_YearOK[0, 69, 5] := true;
_YearOK[1, 69, 5] := true;
_YearOK[0, 70, 5] := true;
_YearOK[1, 70, 5] := true;
_YearOK[0, 71, 5] := true;
_YearOK[1, 71, 5] := true;
_YearOK[0, 72, 5] := true;
_YearOK[1, 72, 5] := true;
_YearOK[0, 73, 5] := true;
_YearOK[1, 73, 5] := true;
_YearOK[0, 74, 5] := true;
_YearOK[1, 74, 5] := true;
_YearOK[0, 75, 5] := true;
_YearOK[1, 75, 5] := true;
_YearOK[0, 76, 5] := true;
_YearOK[1, 76, 5] := true;
_YearOK[0, 77, 5] := true;
_YearOK[1, 77, 5] := true;
_YearOK[0, 78, 5] := true;
_YearOK[1, 78, 5] := true;
_YearOK[0, 79, 5] := true;
_YearOK[1, 79, 5] := true;
_YearOK[0, 80, 5] := true;
_YearOK[1, 80, 5] := true;
_YearOK[0, 81, 5] := true;
_YearOK[1, 81, 5] := true;
_YearOK[0, 82, 5] := true;
_YearOK[1, 82, 5] := true;
_YearOK[0, 83, 5] := true;
_YearOK[1, 83, 5] := true;
_YearOK[0, 84, 5] := true;
_YearOK[1, 84, 5] := true;
_YearOK[0, 85, 5] := true;
_YearOK[1, 85, 5] := true;
_YearOK[0, 86, 5] := true;
_YearOK[1, 86, 5] := true;
_YearOK[0, 87, 5] := true;
_YearOK[1, 87, 5] := true;
_YearOK[0, 88, 5] := true;
_YearOK[1, 88, 5] := true;
_YearOK[0, 89, 5] := true;
_YearOK[1, 89, 5] := true;
_YearOK[0, 90, 5] := true;
_YearOK[1, 90, 5] := true;
_YearOK[0, 91, 5] := true;
_YearOK[1, 91, 5] := true;
_YearOK[0, 92, 5] := true;
_YearOK[1, 92, 5] := true;
_YearOK[0, 93, 5] := true;
_YearOK[1, 93, 5] := true;
_YearOK[0, 94, 5] := true;
_YearOK[1, 94, 5] := true;
_YearOK[0, 95, 5] := true;
_YearOK[1, 95, 5] := true;
_YearOK[0, 96, 5] := true;
_YearOK[1, 96, 5] := true;
_YearOK[0, 97, 5] := true;
_YearOK[1, 97, 5] := true;
_YearOK[0, 98, 5] := true;
_YearOK[1, 98, 5] := true;
_YearOK[0, 99, 5] := true;
_YearOK[1, 99, 5] := true;
_YearOK[0, 100, 5] := true;
```

```
_YearOK[1, 100, 5] := true;
_YearOK[0, 0, 6] := true;
_YearOK[1, 0, 6] := true;
_YearOK[0, 0, 7] := true;
_YearOK[1, 0, 7] := true;
_YearOK[0, 0, 8] := true;
_YearOK[1, 0, 8] := true;
_YearOK[0, 0, 9] := true;
_YearOK[1, 0, 9] := true;
_YearOK[0, 0, 10] := true;
_YearOK[1, 0, 10] := true;
_YearOK[0, 0, 11] := true;
_YearOK[1, 0, 11] := true;
_YearOK[0, 0, 12] := true;
_YearOK[1, 0, 12] := true;
_YearOK[0, 0, 13] := true;
_YearOK[1, 0, 13] := true;
_YearOK[0, 0, 14] := true;
_YearOK[1, 0, 14] := true;
_YearOK[0, 0, 15] := true;
_YearOK[1, 0, 15] := true;
_YearOK[0, 0, 16] := true;
_YearOK[1, 0, 16] := true;
_YearOK[0, 0, 17] := true;
_YearOK[1, 0, 17] := true;
_YearOK[0, 3, 18] := true;
_YearOK[1, 3, 18] := true;
_YearOK[0, 6, 18] := true;
_YearOK[1, 6, 18] := true;
_YearOK[0, 9, 18] := true;
_YearOK[1, 9, 18] := true;
_YearOK[0, 12, 18] := true;
_YearOK[1, 12, 18] := true;
_YearOK[0, 15, 18] := true;
_YearOK[1, 15, 18] := true;
_YearOK[0, 18, 18] := true;
_YearOK[1, 18, 18] := true;
_YearOK[0, 21, 18] := true;
_YearOK[1, 21, 18] := true;
_YearOK[0, 24, 18] := true;
_YearOK[1, 24, 18] := true;
_YearOK[0, 27, 18] := true;
_YearOK[1, 27, 18] := true;
_YearOK[0, 30, 18] := true;
_YearOK[1, 30, 18] := true;
_YearOK[0, 33, 18] := true;
_YearOK[1, 33, 18] := true;
_YearOK[0, 36, 18] := true;
_YearOK[1, 36, 18] := true;
_YearOK[0, 39, 18] := true;
_YearOK[1, 39, 18] := true;
_YearOK[0, 42, 18] := true;
_YearOK[1, 42, 18] := true;
_YearOK[0, 45, 18] := true;
_YearOK[1, 45, 18] := true;
_YearOK[0, 48, 18] := true;
_YearOK[1, 48, 18] := true;
_YearOK[0, 51, 18] := true;
_YearOK[1, 51, 18] := true;
_YearOK[0, 54, 18] := true;
_YearOK[1, 54, 18] := true;
_YearOK[0, 57, 18] := true;
_YearOK[1, 57, 18] := true;
_YearOK[0, 60, 18] := true;
_YearOK[1, 60, 18] := true;
_YearOK[0, 63, 18] := true;
_YearOK[1, 63, 18] := true;
_YearOK[0, 66, 18] := true;
_YearOK[1, 66, 18] := true;
_YearOK[0, 69, 18] := true;
_YearOK[1, 69, 18] := true;
_YearOK[0, 72, 18] := true;
_YearOK[1, 72, 18] := true;
_YearOK[0, 75, 18] := true;
_YearOK[1, 75, 18] := true;
```

```
_YearOK[0, 78, 18] := true;
_YearOK[1, 78, 18] := true;
_YearOK[0, 81, 18] := true;
_YearOK[1, 81, 18] := true;
_YearOK[0, 84, 18] := true;
_YearOK[1, 84, 18] := true;
_YearOK[0, 87, 18] := true;
_YearOK[1, 87, 18] := true;
_YearOK[0, 90, 18] := true;
_YearOK[1, 90, 18] := true;
_YearOK[0, 93, 18] := true;
_YearOK[1, 93, 18] := true;
_YearOK[0, 96, 18] := true;
_YearOK[1, 96, 18] := true;
_YearOK[0, 99, 18] := true;
_YearOK[1, 99, 18] := true;
_YearOK[0, 3, 19] := true;
_YearOK[1, 3, 19] := true;
_YearOK[0, 6, 19] := true;
_YearOK[1, 6, 19] := true;
_YearOK[0, 9, 19] := true;
_YearOK[1, 9, 19] := true;
_YearOK[0, 12, 19] := true;
_YearOK[1, 12, 19] := true;
_YearOK[0, 15, 19] := true;
_YearOK[1, 15, 19] := true;
_YearOK[0, 18, 19] := true;
_YearOK[1, 18, 19] := true;
_YearOK[0, 21, 19] := true;
_YearOK[1, 21, 19] := true;
_YearOK[0, 24, 19] := true;
_YearOK[1, 24, 19] := true;
_YearOK[0, 27, 19] := true;
_YearOK[1, 27, 19] := true;
_YearOK[0, 30, 19] := true;
_YearOK[1, 30, 19] := true;
_YearOK[0, 33, 19] := true;
_YearOK[1, 33, 19] := true;
_YearOK[0, 36, 19] := true;
_YearOK[1, 36, 19] := true;
_YearOK[0, 39, 19] := true;
_YearOK[1, 39, 19] := true;
_YearOK[0, 42, 19] := true;
_YearOK[1, 42, 19] := true;
_YearOK[0, 45, 19] := true;
_YearOK[1, 45, 19] := true;
_YearOK[0, 48, 19] := true;
_YearOK[1, 48, 19] := true;
_YearOK[0, 51, 19] := true;
_YearOK[1, 51, 19] := true;
_YearOK[0, 54, 19] := true;
_YearOK[1, 54, 19] := true;
_YearOK[0, 57, 19] := true;
_YearOK[1, 57, 19] := true;
_YearOK[0, 60, 19] := true;
_YearOK[1, 60, 19] := true;
_YearOK[0, 63, 19] := true;
_YearOK[1, 63, 19] := true;
_YearOK[0, 66, 19] := true;
_YearOK[1, 66, 19] := true;
_YearOK[0, 69, 19] := true;
_YearOK[1, 69, 19] := true;
_YearOK[0, 72, 19] := true;
_YearOK[1, 72, 19] := true;
_YearOK[0, 75, 19] := true;
_YearOK[1, 75, 19] := true;
_YearOK[0, 78, 19] := true;
_YearOK[1, 78, 19] := true;
_YearOK[0, 81, 19] := true;
_YearOK[1, 81, 19] := true;
_YearOK[0, 84, 19] := true;
_YearOK[1, 84, 19] := true;
_YearOK[0, 87, 19] := true;
_YearOK[1, 87, 19] := true;
```

```
_YearOK[0, 90, 19] := true;
_YearOK[1, 90, 19] := true;
_YearOK[0, 93, 19] := true;
_YearOK[1, 93, 19] := true;
_YearOK[0, 96, 19] := true;
_YearOK[1, 96, 19] := true;
_YearOK[0, 99, 19] := true;
_YearOK[1, 99, 19] := true;
_YearOK[0, 3, 20] := true;
_YearOK[1, 3, 20] := true;
_YearOK[0, 6, 20] := true;
_YearOK[1, 6, 20] := true;
_YearOK[0, 9, 20] := true;
_YearOK[1, 9, 20] := true;
_YearOK[0, 12, 20] := true;
_YearOK[1, 12, 20] := true;
_YearOK[0, 15, 20] := true;
_YearOK[1, 15, 20] := true;
_YearOK[0, 18, 20] := true;
_YearOK[1, 18, 20] := true;
_YearOK[0, 21, 20] := true;
_YearOK[1, 21, 20] := true;
_YearOK[0, 24, 20] := true;
_YearOK[1, 24, 20] := true;
_YearOK[0, 27, 20] := true;
_YearOK[1, 27, 20] := true;
_YearOK[0, 30, 20] := true;
_YearOK[1, 30, 20] := true;
_YearOK[0, 33, 20] := true;
_YearOK[1, 33, 20] := true;
_YearOK[0, 36, 20] := true;
_YearOK[1, 36, 20] := true;
_YearOK[0, 39, 20] := true;
_YearOK[1, 39, 20] := true;
_YearOK[0, 42, 20] := true;
_YearOK[1, 42, 20] := true;
_YearOK[0, 45, 20] := true;
_YearOK[1, 45, 20] := true;
_YearOK[0, 48, 20] := true;
_YearOK[1, 48, 20] := true;
_YearOK[0, 51, 20] := true;
_YearOK[1, 51, 20] := true;
_YearOK[0, 54, 20] := true;
_YearOK[1, 54, 20] := true;
_YearOK[0, 57, 20] := true;
_YearOK[1, 57, 20] := true;
_YearOK[0, 60, 20] := true;
_YearOK[1, 60, 20] := true;
_YearOK[0, 63, 20] := true;
_YearOK[1, 63, 20] := true;
_YearOK[0, 66, 20] := true;
_YearOK[1, 66, 20] := true;
_YearOK[0, 69, 20] := true;
_YearOK[1, 69, 20] := true;
_YearOK[0, 72, 20] := true;
_YearOK[1, 72, 20] := true;
_YearOK[0, 75, 20] := true;
_YearOK[1, 75, 20] := true;
_YearOK[0, 78, 20] := true;
_YearOK[1, 78, 20] := true;
_YearOK[0, 81, 20] := true;
_YearOK[1, 81, 20] := true;
_YearOK[0, 84, 20] := true;
_YearOK[1, 84, 20] := true;
_YearOK[0, 87, 20] := true;
_YearOK[1, 87, 20] := true;
_YearOK[0, 90, 20] := true;
_YearOK[1, 90, 20] := true;
_YearOK[0, 93, 20] := true;
_YearOK[1, 93, 20] := true;
_YearOK[0, 96, 20] := true;
_YearOK[1, 96, 20] := true;
_YearOK[0, 99, 20] := true;
_YearOK[1, 99, 20] := true;
_YearOK[0, 3, 21] := true;
```

```
_YearOK[1, 3, 21] := true;
_YearOK[0, 6, 21] := true;
_YearOK[1, 6, 21] := true;
_YearOK[0, 9, 21] := true;
_YearOK[1, 9, 21] := true;
_YearOK[0, 12, 21] := true;
_YearOK[1, 12, 21] := true;
_YearOK[0, 15, 21] := true;
_YearOK[1, 15, 21] := true;
_YearOK[0, 18, 21] := true;
_YearOK[1, 18, 21] := true;
_YearOK[0, 21, 21] := true;
_YearOK[1, 21, 21] := true;
_YearOK[0, 24, 21] := true;
_YearOK[1, 24, 21] := true;
_YearOK[0, 27, 21] := true;
_YearOK[1, 27, 21] := true;
_YearOK[0, 30, 21] := true;
_YearOK[1, 30, 21] := true;
_YearOK[0, 33, 21] := true;
_YearOK[1, 33, 21] := true;
_YearOK[0, 36, 21] := true;
_YearOK[1, 36, 21] := true;
_YearOK[0, 39, 21] := true;
_YearOK[1, 39, 21] := true;
_YearOK[0, 42, 21] := true;
_YearOK[1, 42, 21] := true;
_YearOK[0, 45, 21] := true;
_YearOK[1, 45, 21] := true;
_YearOK[0, 48, 21] := true;
_YearOK[1, 48, 21] := true;
_YearOK[0, 51, 21] := true;
_YearOK[1, 51, 21] := true;
_YearOK[0, 54, 21] := true;
_YearOK[1, 54, 21] := true;
_YearOK[0, 57, 21] := true;
_YearOK[1, 57, 21] := true;
_YearOK[0, 60, 21] := true;
_YearOK[1, 60, 21] := true;
_YearOK[0, 63, 21] := true;
_YearOK[1, 63, 21] := true;
_YearOK[0, 66, 21] := true;
_YearOK[1, 66, 21] := true;
_YearOK[0, 69, 21] := true;
_YearOK[1, 69, 21] := true;
_YearOK[0, 72, 21] := true;
_YearOK[1, 72, 21] := true;
_YearOK[0, 75, 21] := true;
_YearOK[1, 75, 21] := true;
_YearOK[0, 78, 21] := true;
_YearOK[1, 78, 21] := true;
_YearOK[0, 81, 21] := true;
_YearOK[1, 81, 21] := true;
_YearOK[0, 84, 21] := true;
_YearOK[1, 84, 21] := true;
_YearOK[0, 87, 21] := true;
_YearOK[1, 87, 21] := true;
_YearOK[0, 90, 21] := true;
_YearOK[1, 90, 21] := true;
_YearOK[0, 93, 21] := true;
_YearOK[1, 93, 21] := true;
_YearOK[0, 96, 21] := true;
_YearOK[1, 96, 21] := true;
_YearOK[0, 99, 21] := true;
_YearOK[1, 99, 21] := true;
_YearOK[0, 3, 22] := true;
_YearOK[1, 3, 22] := true;
_YearOK[0, 6, 22] := true;
_YearOK[1, 6, 22] := true;
_YearOK[0, 9, 22] := true;
_YearOK[1, 9, 22] := true;
_YearOK[0, 12, 22] := true;
_YearOK[1, 12, 22] := true;
_YearOK[0, 15, 22] := true;
```

```
  _YearOK[1, 15, 22] := true;
  _YearOK[0, 18, 22] := true;
  _YearOK[1, 18, 22] := true;
  _YearOK[0, 21, 22] := true;
  _YearOK[1, 21, 22] := true;
  _YearOK[0, 24, 22] := true;
  _YearOK[1, 24, 22] := true;
  _YearOK[0, 27, 22] := true;
  _YearOK[1, 27, 22] := true;
  _YearOK[0, 30, 22] := true;
  _YearOK[1, 30, 22] := true;
  _YearOK[0, 33, 22] := true;
  _YearOK[1, 33, 22] := true;
  _YearOK[0, 36, 22] := true;
  _YearOK[1, 36, 22] := true;
  _YearOK[0, 39, 22] := true;
  _YearOK[1, 39, 22] := true;
  _YearOK[0, 42, 22] := true;
  _YearOK[1, 42, 22] := true;
  _YearOK[0, 45, 22] := true;
  _YearOK[1, 45, 22] := true;
  _YearOK[0, 48, 22] := true;
  _YearOK[1, 48, 22] := true;
  _YearOK[0, 51, 22] := true;
  _YearOK[1, 51, 22] := true;
  _YearOK[0, 54, 22] := true;
  _YearOK[1, 54, 22] := true;
  _YearOK[0, 57, 22] := true;
  _YearOK[1, 57, 22] := true;
  _YearOK[0, 60, 22] := true;
  _YearOK[1, 60, 22] := true;
  _YearOK[0, 63, 22] := true;
  _YearOK[1, 63, 22] := true;
  _YearOK[0, 66, 22] := true;
  _YearOK[1, 66, 22] := true;
  _YearOK[0, 69, 22] := true;
  _YearOK[1, 69, 22] := true;
  _YearOK[0, 72, 22] := true;
  _YearOK[1, 72, 22] := true;
  _YearOK[0, 75, 22] := true;
  _YearOK[1, 75, 22] := true;
  _YearOK[0, 78, 22] := true;
  _YearOK[1, 78, 22] := true;
  _YearOK[0, 81, 22] := true;
  _YearOK[1, 81, 22] := true;
  _YearOK[0, 84, 22] := true;
  _YearOK[1, 84, 22] := true;
  _YearOK[0, 87, 22] := true;
  _YearOK[1, 87, 22] := true;
  _YearOK[0, 90, 22] := true;
  _YearOK[1, 90, 22] := true;
  _YearOK[0, 93, 22] := true;
  _YearOK[1, 93, 22] := true;
  _YearOK[0, 96, 22] := true;
  _YearOK[1, 96, 22] := true;
  _YearOK[0, 99, 22] := true;
  _YearOK[1, 99, 22] := true;
  _YearOK[0, 3, 23] := true;
  _YearOK[1, 3, 23] := true;
  _YearOK[0, 6, 23] := true;
  _YearOK[1, 6, 23] := true;
  _YearOK[0, 9, 23] := true;
  _YearOK[1, 9, 23] := true;
  _YearOK[0, 12, 23] := true;
  _YearOK[1, 12, 23] := true;
  _YearOK[0, 15, 23] := true;
  _YearOK[1, 15, 23] := true;
  _YearOK[0, 18, 23] := true;
  _YearOK[1, 18, 23] := true;
  _YearOK[0, 21, 23] := true;
  _YearOK[1, 21, 23] := true;
  _YearOK[0, 24, 23] := true;
  _YearOK[1, 24, 23] := true;
  _YearOK[0, 27, 23] := true;
  _YearOK[1, 27, 23] := true;
```

```
  _YearOK[0, 30, 23] := true;
  _YearOK[1, 30, 23] := true;
  _YearOK[0, 33, 23] := true;
  _YearOK[1, 33, 23] := true;
  _YearOK[0, 36, 23] := true;
  _YearOK[1, 36, 23] := true;
  _YearOK[0, 39, 23] := true;
  _YearOK[1, 39, 23] := true;
  _YearOK[0, 42, 23] := true;
  _YearOK[1, 42, 23] := true;
  _YearOK[0, 45, 23] := true;
  _YearOK[1, 45, 23] := true;
  _YearOK[0, 48, 23] := true;
  _YearOK[1, 48, 23] := true;
  _YearOK[0, 51, 23] := true;
  _YearOK[1, 51, 23] := true;
  _YearOK[0, 54, 23] := true;
  _YearOK[1, 54, 23] := true;
  _YearOK[0, 57, 23] := true;
  _YearOK[1, 57, 23] := true;
  _YearOK[0, 60, 23] := true;
  _YearOK[1, 60, 23] := true;
  _YearOK[0, 63, 23] := true;
  _YearOK[1, 63, 23] := true;
  _YearOK[0, 66, 23] := true;
  _YearOK[1, 66, 23] := true;
  _YearOK[0, 69, 23] := true;
  _YearOK[1, 69, 23] := true;
  _YearOK[0, 72, 23] := true;
  _YearOK[1, 72, 23] := true;
  _YearOK[0, 75, 23] := true;
  _YearOK[1, 75, 23] := true;
  _YearOK[0, 78, 23] := true;
  _YearOK[1, 78, 23] := true;
  _YearOK[0, 81, 23] := true;
  _YearOK[1, 81, 23] := true;
  _YearOK[0, 84, 23] := true;
  _YearOK[1, 84, 23] := true;
  _YearOK[0, 87, 23] := true;
  _YearOK[1, 87, 23] := true;
  _YearOK[0, 90, 23] := true;
  _YearOK[1, 90, 23] := true;
  _YearOK[0, 93, 23] := true;
  _YearOK[1, 93, 23] := true;
  _YearOK[0, 96, 23] := true;
  _YearOK[1, 96, 23] := true;
  _YearOK[0, 99, 23] := true;
  _YearOK[1, 99, 23] := true;
  _YearOK[0, 3, 24] := true;
  _YearOK[1, 3, 24] := true;
  _YearOK[0, 4, 24] := true;
  _YearOK[1, 4, 24] := true;
  _YearOK[0, 5, 24] := true;
  _YearOK[1, 5, 24] := true;
  _YearOK[0, 6, 24] := true;
  _YearOK[1, 6, 24] := true;
  _YearOK[0, 7, 24] := true;
  _YearOK[1, 7, 24] := true;
  _YearOK[0, 8, 24] := true;
  _YearOK[1, 8, 24] := true;
  _YearOK[0, 9, 24] := true;
  _YearOK[1, 9, 24] := true;
  _YearOK[0, 10, 24] := true;
  _YearOK[1, 10, 24] := true;
  _YearOK[0, 11, 24] := true;
  _YearOK[1, 11, 24] := true;
  _YearOK[0, 12, 24] := true;
  _YearOK[1, 12, 24] := true;
  _YearOK[0, 13, 24] := true;
  _YearOK[1, 13, 24] := true;
  _YearOK[0, 14, 24] := true;
  _YearOK[1, 14, 24] := true;
  _YearOK[0, 15, 24] := true;
  _YearOK[1, 15, 24] := true;
```

```
  _YearOK[0, 16, 24] := true;
  _YearOK[1, 16, 24] := true;
  _YearOK[0, 17, 24] := true;
  _YearOK[1, 17, 24] := true;
  _YearOK[0, 18, 24] := true;
  _YearOK[1, 18, 24] := true;
  _YearOK[0, 19, 24] := true;
  _YearOK[1, 19, 24] := true;
  _YearOK[0, 20, 24] := true;
  _YearOK[1, 20, 24] := true;
  _YearOK[0, 21, 24] := true;
  _YearOK[1, 21, 24] := true;
  _YearOK[0, 22, 24] := true;
  _YearOK[1, 22, 24] := true;
  _YearOK[0, 23, 24] := true;
  _YearOK[1, 23, 24] := true;
  _YearOK[0, 24, 24] := true;
  _YearOK[1, 24, 24] := true;
  _YearOK[0, 25, 24] := true;
  _YearOK[1, 25, 24] := true;
  _YearOK[0, 26, 24] := true;
  _YearOK[1, 26, 24] := true;
  _YearOK[0, 27, 24] := true;
  _YearOK[1, 27, 24] := true;
  _YearOK[0, 28, 24] := true;
  _YearOK[1, 28, 24] := true;
  _YearOK[0, 29, 24] := true;
  _YearOK[1, 29, 24] := true;
  _YearOK[0, 30, 24] := true;
  _YearOK[1, 30, 24] := true;
  _YearOK[0, 31, 24] := true;
  _YearOK[1, 31, 24] := true;
  _YearOK[0, 32, 24] := true;
  _YearOK[1, 32, 24] := true;
  _YearOK[0, 33, 24] := true;
  _YearOK[1, 33, 24] := true;
  _YearOK[0, 34, 24] := true;
  _YearOK[1, 34, 24] := true;
  _YearOK[0, 35, 24] := true;
  _YearOK[1, 35, 24] := true;
  _YearOK[0, 36, 24] := true;
  _YearOK[1, 36, 24] := true;
  _YearOK[0, 37, 24] := true;
  _YearOK[1, 37, 24] := true;
  _YearOK[0, 38, 24] := true;
  _YearOK[1, 38, 24] := true;
  _YearOK[0, 39, 24] := true;
  _YearOK[1, 39, 24] := true;
  _YearOK[0, 40, 24] := true;
  _YearOK[1, 40, 24] := true;
  _YearOK[0, 41, 24] := true;
  _YearOK[1, 41, 24] := true;
  _YearOK[0, 42, 24] := true;
  _YearOK[1, 42, 24] := true;
  _YearOK[0, 43, 24] := true;
  _YearOK[1, 43, 24] := true;
  _YearOK[0, 44, 24] := true;
  _YearOK[1, 44, 24] := true;
  _YearOK[0, 45, 24] := true;
  _YearOK[1, 45, 24] := true;
  _YearOK[0, 46, 24] := true;
  _YearOK[1, 46, 24] := true;
  _YearOK[0, 47, 24] := true;
  _YearOK[1, 47, 24] := true;
  _YearOK[0, 48, 24] := true;
  _YearOK[1, 48, 24] := true;
  _YearOK[0, 49, 24] := true;
  _YearOK[1, 49, 24] := true;
  _YearOK[0, 50, 24] := true;
  _YearOK[1, 50, 24] := true;
  _YearOK[0, 51, 24] := true;
  _YearOK[1, 51, 24] := true;
  _YearOK[0, 52, 24] := true;
  _YearOK[1, 52, 24] := true;
  _YearOK[0, 53, 24] := true;
```

```
  _YearOK[1, 53, 24] := true;
  _YearOK[0, 54, 24] := true;
  _YearOK[1, 54, 24] := true;
  _YearOK[0, 55, 24] := true;
  _YearOK[1, 55, 24] := true;
  _YearOK[0, 56, 24] := true;
  _YearOK[1, 56, 24] := true;
  _YearOK[0, 57, 24] := true;
  _YearOK[1, 57, 24] := true;
  _YearOK[0, 58, 24] := true;
  _YearOK[1, 58, 24] := true;
  _YearOK[0, 59, 24] := true;
  _YearOK[1, 59, 24] := true;
  _YearOK[0, 60, 24] := true;
  _YearOK[1, 60, 24] := true;
  _YearOK[0, 61, 24] := true;
  _YearOK[1, 61, 24] := true;
  _YearOK[0, 62, 24] := true;
  _YearOK[1, 62, 24] := true;
  _YearOK[0, 63, 24] := true;
  _YearOK[1, 63, 24] := true;
  _YearOK[0, 64, 24] := true;
  _YearOK[1, 64, 24] := true;
  _YearOK[0, 65, 24] := true;
  _YearOK[1, 65, 24] := true;
  _YearOK[0, 66, 24] := true;
  _YearOK[1, 66, 24] := true;
  _YearOK[0, 67, 24] := true;
  _YearOK[1, 67, 24] := true;
  _YearOK[0, 68, 24] := true;
  _YearOK[1, 68, 24] := true;
  _YearOK[0, 69, 24] := true;
  _YearOK[1, 69, 24] := true;
  _YearOK[0, 70, 24] := true;
  _YearOK[1, 70, 24] := true;
  _YearOK[0, 71, 24] := true;
  _YearOK[1, 71, 24] := true;
  _YearOK[0, 72, 24] := true;
  _YearOK[1, 72, 24] := true;
  _YearOK[0, 73, 24] := true;
  _YearOK[1, 73, 24] := true;
  _YearOK[0, 74, 24] := true;
  _YearOK[1, 74, 24] := true;
  _YearOK[0, 75, 24] := true;
  _YearOK[1, 75, 24] := true;
  _YearOK[0, 76, 24] := true;
  _YearOK[1, 76, 24] := true;
  _YearOK[0, 77, 24] := true;
  _YearOK[1, 77, 24] := true;
  _YearOK[0, 78, 24] := true;
  _YearOK[1, 78, 24] := true;
  _YearOK[0, 79, 24] := true;
  _YearOK[1, 79, 24] := true;
  _YearOK[0, 80, 24] := true;
  _YearOK[1, 80, 24] := true;
  _YearOK[0, 81, 24] := true;
  _YearOK[1, 81, 24] := true;
  _YearOK[0, 82, 24] := true;
  _YearOK[1, 82, 24] := true;
  _YearOK[0, 83, 24] := true;
  _YearOK[1, 83, 24] := true;
  _YearOK[0, 84, 24] := true;
  _YearOK[1, 84, 24] := true;
  _YearOK[0, 85, 24] := true;
  _YearOK[1, 85, 24] := true;
  _YearOK[0, 86, 24] := true;
  _YearOK[1, 86, 24] := true;
  _YearOK[0, 87, 24] := true;
  _YearOK[1, 87, 24] := true;
  _YearOK[0, 88, 24] := true;
  _YearOK[1, 88, 24] := true;
  _YearOK[0, 89, 24] := true;
  _YearOK[1, 89, 24] := true;
  _YearOK[0, 90, 24] := true;
```

```
  _YearOK[1, 90, 24] := true;
  _YearOK[0, 91, 24] := true;
  _YearOK[1, 91, 24] := true;
  _YearOK[0, 92, 24] := true;
  _YearOK[1, 92, 24] := true;
  _YearOK[0, 93, 24] := true;
  _YearOK[1, 93, 24] := true;
  _YearOK[0, 94, 24] := true;
  _YearOK[1, 94, 24] := true;
  _YearOK[0, 95, 24] := true;
  _YearOK[1, 95, 24] := true;
  _YearOK[0, 96, 24] := true;
  _YearOK[1, 96, 24] := true;
  _YearOK[0, 97, 24] := true;
  _YearOK[1, 97, 24] := true;
  _YearOK[0, 98, 24] := true;
  _YearOK[1, 98, 24] := true;
  _YearOK[0, 99, 24] := true;
  _YearOK[1, 99, 24] := true;
  _YearOK[0, 100, 24] := true;
  _YearOK[1, 100, 24] := true;
  _YearOK[0, 3, 25] := true;
  _YearOK[1, 3, 25] := true;
  _YearOK[0, 4, 25] := true;
  _YearOK[1, 4, 25] := true;
  _YearOK[0, 5, 25] := true;
  _YearOK[1, 5, 25] := true;
  _YearOK[0, 6, 25] := true;
  _YearOK[1, 6, 25] := true;
  _YearOK[0, 7, 25] := true;
  _YearOK[1, 7, 25] := true;
  _YearOK[0, 8, 25] := true;
  _YearOK[1, 8, 25] := true;
  _YearOK[0, 9, 25] := true;
  _YearOK[1, 9, 25] := true;
  _YearOK[0, 10, 25] := true;
  _YearOK[1, 10, 25] := true;
  _YearOK[0, 11, 25] := true;
  _YearOK[1, 11, 25] := true;
  _YearOK[0, 12, 25] := true;
  _YearOK[1, 12, 25] := true;
  _YearOK[0, 13, 25] := true;
  _YearOK[1, 13, 25] := true;
  _YearOK[0, 14, 25] := true;
  _YearOK[1, 14, 25] := true;
  _YearOK[0, 15, 25] := true;
  _YearOK[1, 15, 25] := true;
  _YearOK[0, 16, 25] := true;
  _YearOK[1, 16, 25] := true;
  _YearOK[0, 17, 25] := true;
  _YearOK[1, 17, 25] := true;
  _YearOK[0, 18, 25] := true;
  _YearOK[1, 18, 25] := true;
  _YearOK[0, 19, 25] := true;
  _YearOK[1, 19, 25] := true;
  _YearOK[0, 20, 25] := true;
  _YearOK[1, 20, 25] := true;
  _YearOK[0, 21, 25] := true;
  _YearOK[1, 21, 25] := true;
  _YearOK[0, 22, 25] := true;
  _YearOK[1, 22, 25] := true;
  _YearOK[0, 23, 25] := true;
  _YearOK[1, 23, 25] := true;
  _YearOK[0, 24, 25] := true;
  _YearOK[1, 24, 25] := true;
  _YearOK[0, 25, 25] := true;
  _YearOK[1, 25, 25] := true;
  _YearOK[0, 26, 25] := true;
  _YearOK[1, 26, 25] := true;
  _YearOK[0, 27, 25] := true;
  _YearOK[1, 27, 25] := true;
  _YearOK[0, 28, 25] := true;
  _YearOK[1, 28, 25] := true;
  _YearOK[0, 29, 25] := true;
  _YearOK[1, 29, 25] := true;
```

```
  _YearOK[0, 30, 25] := true;
  _YearOK[1, 30, 25] := true;
  _YearOK[0, 31, 25] := true;
  _YearOK[1, 31, 25] := true;
  _YearOK[0, 32, 25] := true;
  _YearOK[1, 32, 25] := true;
  _YearOK[0, 33, 25] := true;
  _YearOK[1, 33, 25] := true;
  _YearOK[0, 34, 25] := true;
  _YearOK[1, 34, 25] := true;
  _YearOK[0, 35, 25] := true;
  _YearOK[1, 35, 25] := true;
  _YearOK[0, 36, 25] := true;
  _YearOK[1, 36, 25] := true;
  _YearOK[0, 37, 25] := true;
  _YearOK[1, 37, 25] := true;
  _YearOK[0, 38, 25] := true;
  _YearOK[1, 38, 25] := true;
  _YearOK[0, 39, 25] := true;
  _YearOK[1, 39, 25] := true;
  _YearOK[0, 40, 25] := true;
  _YearOK[1, 40, 25] := true;
  _YearOK[0, 41, 25] := true;
  _YearOK[1, 41, 25] := true;
  _YearOK[0, 42, 25] := true;
  _YearOK[1, 42, 25] := true;
  _YearOK[0, 43, 25] := true;
  _YearOK[1, 43, 25] := true;
  _YearOK[0, 44, 25] := true;
  _YearOK[1, 44, 25] := true;
  _YearOK[0, 45, 25] := true;
  _YearOK[1, 45, 25] := true;
  _YearOK[0, 46, 25] := true;
  _YearOK[1, 46, 25] := true;
  _YearOK[0, 47, 25] := true;
  _YearOK[1, 47, 25] := true;
  _YearOK[0, 48, 25] := true;
  _YearOK[1, 48, 25] := true;
  _YearOK[0, 49, 25] := true;
  _YearOK[1, 49, 25] := true;
  _YearOK[0, 50, 25] := true;
  _YearOK[1, 50, 25] := true;
  _YearOK[0, 51, 25] := true;
  _YearOK[1, 51, 25] := true;
  _YearOK[0, 52, 25] := true;
  _YearOK[1, 52, 25] := true;
  _YearOK[0, 53, 25] := true;
  _YearOK[1, 53, 25] := true;
  _YearOK[0, 54, 25] := true;
  _YearOK[1, 54, 25] := true;
  _YearOK[0, 55, 25] := true;
  _YearOK[1, 55, 25] := true;
  _YearOK[0, 56, 25] := true;
  _YearOK[1, 56, 25] := true;
  _YearOK[0, 57, 25] := true;
  _YearOK[1, 57, 25] := true;
  _YearOK[0, 58, 25] := true;
  _YearOK[1, 58, 25] := true;
  _YearOK[0, 59, 25] := true;
  _YearOK[1, 59, 25] := true;
  _YearOK[0, 60, 25] := true;
  _YearOK[1, 60, 25] := true;
  _YearOK[0, 61, 25] := true;
  _YearOK[1, 61, 25] := true;
  _YearOK[0, 62, 25] := true;
  _YearOK[1, 62, 25] := true;
  _YearOK[0, 63, 25] := true;
  _YearOK[1, 63, 25] := true;
  _YearOK[0, 64, 25] := true;
  _YearOK[1, 64, 25] := true;
  _YearOK[0, 65, 25] := true;
  _YearOK[1, 65, 25] := true;
  _YearOK[0, 66, 25] := true;
  _YearOK[1, 66, 25] := true;
```

```
_YearOK[0, 67, 25] := true;
_YearOK[1, 67, 25] := true;
_YearOK[0, 68, 25] := true;
_YearOK[1, 68, 25] := true;
_YearOK[0, 69, 25] := true;
_YearOK[1, 69, 25] := true;
_YearOK[0, 70, 25] := true;
_YearOK[1, 70, 25] := true;
_YearOK[0, 71, 25] := true;
_YearOK[1, 71, 25] := true;
_YearOK[0, 72, 25] := true;
_YearOK[1, 72, 25] := true;
_YearOK[0, 73, 25] := true;
_YearOK[1, 73, 25] := true;
_YearOK[0, 74, 25] := true;
_YearOK[1, 74, 25] := true;
_YearOK[0, 75, 25] := true;
_YearOK[1, 75, 25] := true;
_YearOK[0, 76, 25] := true;
_YearOK[1, 76, 25] := true;
_YearOK[0, 77, 25] := true;
_YearOK[1, 77, 25] := true;
_YearOK[0, 78, 25] := true;
_YearOK[1, 78, 25] := true;
_YearOK[0, 79, 25] := true;
_YearOK[1, 79, 25] := true;
_YearOK[0, 80, 25] := true;
_YearOK[1, 80, 25] := true;
_YearOK[0, 81, 25] := true;
_YearOK[1, 81, 25] := true;
_YearOK[0, 82, 25] := true;
_YearOK[1, 82, 25] := true;
_YearOK[0, 83, 25] := true;
_YearOK[1, 83, 25] := true;
_YearOK[0, 84, 25] := true;
_YearOK[1, 84, 25] := true;
_YearOK[0, 85, 25] := true;
_YearOK[1, 85, 25] := true;
_YearOK[0, 86, 25] := true;
_YearOK[1, 86, 25] := true;
_YearOK[0, 87, 25] := true;
_YearOK[1, 87, 25] := true;
_YearOK[0, 88, 25] := true;
_YearOK[1, 88, 25] := true;
_YearOK[0, 89, 25] := true;
_YearOK[1, 89, 25] := true;
_YearOK[0, 90, 25] := true;
_YearOK[1, 90, 25] := true;
_YearOK[0, 91, 25] := true;
_YearOK[1, 91, 25] := true;
_YearOK[0, 92, 25] := true;
_YearOK[1, 92, 25] := true;
_YearOK[0, 93, 25] := true;
_YearOK[1, 93, 25] := true;
_YearOK[0, 94, 25] := true;
_YearOK[1, 94, 25] := true;
_YearOK[0, 95, 25] := true;
_YearOK[1, 95, 25] := true;
_YearOK[0, 96, 25] := true;
_YearOK[1, 96, 25] := true;
_YearOK[0, 97, 25] := true;
_YearOK[1, 97, 25] := true;
_YearOK[0, 98, 25] := true;
_YearOK[1, 98, 25] := true;
_YearOK[0, 99, 25] := true;
_YearOK[1, 99, 25] := true;
_YearOK[0, 100, 25] := true;
_YearOK[1, 100, 25] := true;
_YearOK[0, 3, 26] := true;
_YearOK[1, 3, 26] := true;
_YearOK[0, 4, 26] := true;
_YearOK[1, 4, 26] := true;
_YearOK[0, 5, 26] := true;
_YearOK[1, 5, 26] := true;
_YearOK[0, 6, 26] := true;
```

```
_YearOK[1, 6, 26] := true;
_YearOK[0, 7, 26] := true;
_YearOK[1, 7, 26] := true;
_YearOK[0, 8, 26] := true;
_YearOK[1, 8, 26] := true;
_YearOK[0, 9, 26] := true;
_YearOK[1, 9, 26] := true;
_YearOK[0, 10, 26] := true;
_YearOK[1, 10, 26] := true;
_YearOK[0, 11, 26] := true;
_YearOK[1, 11, 26] := true;
_YearOK[0, 12, 26] := true;
_YearOK[1, 12, 26] := true;
_YearOK[0, 13, 26] := true;
_YearOK[1, 13, 26] := true;
_YearOK[0, 14, 26] := true;
_YearOK[1, 14, 26] := true;
_YearOK[0, 15, 26] := true;
_YearOK[1, 15, 26] := true;
_YearOK[0, 16, 26] := true;
_YearOK[1, 16, 26] := true;
_YearOK[0, 17, 26] := true;
_YearOK[1, 17, 26] := true;
_YearOK[0, 18, 26] := true;
_YearOK[1, 18, 26] := true;
_YearOK[0, 19, 26] := true;
_YearOK[1, 19, 26] := true;
_YearOK[0, 20, 26] := true;
_YearOK[1, 20, 26] := true;
_YearOK[0, 21, 26] := true;
_YearOK[1, 21, 26] := true;
_YearOK[0, 22, 26] := true;
_YearOK[1, 22, 26] := true;
_YearOK[0, 23, 26] := true;
_YearOK[1, 23, 26] := true;
_YearOK[0, 24, 26] := true;
_YearOK[1, 24, 26] := true;
_YearOK[0, 25, 26] := true;
_YearOK[1, 25, 26] := true;
_YearOK[0, 26, 26] := true;
_YearOK[1, 26, 26] := true;
_YearOK[0, 27, 26] := true;
_YearOK[1, 27, 26] := true;
_YearOK[0, 28, 26] := true;
_YearOK[1, 28, 26] := true;
_YearOK[0, 29, 26] := true;
_YearOK[1, 29, 26] := true;
_YearOK[0, 30, 26] := true;
_YearOK[1, 30, 26] := true;
_YearOK[0, 31, 26] := true;
_YearOK[1, 31, 26] := true;
_YearOK[0, 32, 26] := true;
_YearOK[1, 32, 26] := true;
_YearOK[0, 33, 26] := true;
_YearOK[1, 33, 26] := true;
_YearOK[0, 34, 26] := true;
_YearOK[1, 34, 26] := true;
_YearOK[0, 35, 26] := true;
_YearOK[1, 35, 26] := true;
_YearOK[0, 36, 26] := true;
_YearOK[1, 36, 26] := true;
_YearOK[0, 37, 26] := true;
_YearOK[1, 37, 26] := true;
_YearOK[0, 38, 26] := true;
_YearOK[1, 38, 26] := true;
_YearOK[0, 39, 26] := true;
_YearOK[1, 39, 26] := true;
_YearOK[0, 40, 26] := true;
_YearOK[1, 40, 26] := true;
_YearOK[0, 41, 26] := true;
_YearOK[1, 41, 26] := true;
_YearOK[0, 42, 26] := true;
_YearOK[1, 42, 26] := true;
_YearOK[0, 43, 26] := true;
```

```
  _YearOK[1, 43, 26] := true;
  _YearOK[0, 44, 26] := true;
  _YearOK[1, 44, 26] := true;
  _YearOK[0, 45, 26] := true;
  _YearOK[1, 45, 26] := true;
  _YearOK[0, 46, 26] := true;
  _YearOK[1, 46, 26] := true;
  _YearOK[0, 47, 26] := true;
  _YearOK[1, 47, 26] := true;
  _YearOK[0, 48, 26] := true;
  _YearOK[1, 48, 26] := true;
  _YearOK[0, 49, 26] := true;
  _YearOK[1, 49, 26] := true;
  _YearOK[0, 50, 26] := true;
  _YearOK[1, 50, 26] := true;
  _YearOK[0, 51, 26] := true;
  _YearOK[1, 51, 26] := true;
  _YearOK[0, 52, 26] := true;
  _YearOK[1, 52, 26] := true;
  _YearOK[0, 53, 26] := true;
  _YearOK[1, 53, 26] := true;
  _YearOK[0, 54, 26] := true;
  _YearOK[1, 54, 26] := true;
  _YearOK[0, 55, 26] := true;
  _YearOK[1, 55, 26] := true;
  _YearOK[0, 56, 26] := true;
  _YearOK[1, 56, 26] := true;
  _YearOK[0, 57, 26] := true;
  _YearOK[1, 57, 26] := true;
  _YearOK[0, 58, 26] := true;
  _YearOK[1, 58, 26] := true;
  _YearOK[0, 59, 26] := true;
  _YearOK[1, 59, 26] := true;
  _YearOK[0, 60, 26] := true;
  _YearOK[1, 60, 26] := true;
  _YearOK[0, 61, 26] := true;
  _YearOK[1, 61, 26] := true;
  _YearOK[0, 62, 26] := true;
  _YearOK[1, 62, 26] := true;
  _YearOK[0, 63, 26] := true;
  _YearOK[1, 63, 26] := true;
  _YearOK[0, 64, 26] := true;
  _YearOK[1, 64, 26] := true;
  _YearOK[0, 65, 26] := true;
  _YearOK[1, 65, 26] := true;
  _YearOK[0, 66, 26] := true;
  _YearOK[1, 66, 26] := true;
  _YearOK[0, 67, 26] := true;
  _YearOK[1, 67, 26] := true;
  _YearOK[0, 68, 26] := true;
  _YearOK[1, 68, 26] := true;
  _YearOK[0, 69, 26] := true;
  _YearOK[1, 69, 26] := true;
  _YearOK[0, 70, 26] := true;
  _YearOK[1, 70, 26] := true;
  _YearOK[0, 71, 26] := true;
  _YearOK[1, 71, 26] := true;
  _YearOK[0, 72, 26] := true;
  _YearOK[1, 72, 26] := true;
  _YearOK[0, 73, 26] := true;
  _YearOK[1, 73, 26] := true;
  _YearOK[0, 74, 26] := true;
  _YearOK[1, 74, 26] := true;
  _YearOK[0, 75, 26] := true;
  _YearOK[1, 75, 26] := true;
  _YearOK[0, 76, 26] := true;
  _YearOK[1, 76, 26] := true;
  _YearOK[0, 77, 26] := true;
  _YearOK[1, 77, 26] := true;
  _YearOK[0, 78, 26] := true;
  _YearOK[1, 78, 26] := true;
  _YearOK[0, 79, 26] := true;
  _YearOK[1, 79, 26] := true;
  _YearOK[0, 80, 26] := true;
  _YearOK[1, 80, 26] := true;
```

```
  _YearOK[0, 81, 26] := true;
  _YearOK[1, 81, 26] := true;
  _YearOK[0, 82, 26] := true;
  _YearOK[1, 82, 26] := true;
  _YearOK[0, 83, 26] := true;
  _YearOK[1, 83, 26] := true;
  _YearOK[0, 84, 26] := true;
  _YearOK[1, 84, 26] := true;
  _YearOK[0, 85, 26] := true;
  _YearOK[1, 85, 26] := true;
  _YearOK[0, 86, 26] := true;
  _YearOK[1, 86, 26] := true;
  _YearOK[0, 87, 26] := true;
  _YearOK[1, 87, 26] := true;
  _YearOK[0, 88, 26] := true;
  _YearOK[1, 88, 26] := true;
  _YearOK[0, 89, 26] := true;
  _YearOK[1, 89, 26] := true;
  _YearOK[0, 90, 26] := true;
  _YearOK[1, 90, 26] := true;
  _YearOK[0, 91, 26] := true;
  _YearOK[1, 91, 26] := true;
  _YearOK[0, 92, 26] := true;
  _YearOK[1, 92, 26] := true;
  _YearOK[0, 93, 26] := true;
  _YearOK[1, 93, 26] := true;
  _YearOK[0, 94, 26] := true;
  _YearOK[1, 94, 26] := true;
  _YearOK[0, 95, 26] := true;
  _YearOK[1, 95, 26] := true;
  _YearOK[0, 96, 26] := true;
  _YearOK[1, 96, 26] := true;
  _YearOK[0, 97, 26] := true;
  _YearOK[1, 97, 26] := true;
  _YearOK[0, 98, 26] := true;
  _YearOK[1, 98, 26] := true;
  _YearOK[0, 99, 26] := true;
  _YearOK[1, 99, 26] := true;
  _YearOK[0, 100, 26] := true;
  _YearOK[1, 100, 26] := true;
  _YearOK[0, 3, 27] := true;
  _YearOK[1, 3, 27] := true;
  _YearOK[0, 4, 27] := true;
  _YearOK[1, 4, 27] := true;
  _YearOK[0, 5, 27] := true;
  _YearOK[1, 5, 27] := true;
  _YearOK[0, 6, 27] := true;
  _YearOK[1, 6, 27] := true;
  _YearOK[0, 7, 27] := true;
  _YearOK[1, 7, 27] := true;
  _YearOK[0, 8, 27] := true;
  _YearOK[1, 8, 27] := true;
  _YearOK[0, 9, 27] := true;
  _YearOK[1, 9, 27] := true;
  _YearOK[0, 10, 27] := true;
  _YearOK[1, 10, 27] := true;
  _YearOK[0, 11, 27] := true;
  _YearOK[1, 11, 27] := true;
  _YearOK[0, 12, 27] := true;
  _YearOK[1, 12, 27] := true;
  _YearOK[0, 13, 27] := true;
  _YearOK[1, 13, 27] := true;
  _YearOK[0, 14, 27] := true;
  _YearOK[1, 14, 27] := true;
  _YearOK[0, 15, 27] := true;
  _YearOK[1, 15, 27] := true;
  _YearOK[0, 16, 27] := true;
  _YearOK[1, 16, 27] := true;
  _YearOK[0, 17, 27] := true;
  _YearOK[1, 17, 27] := true;
  _YearOK[0, 18, 27] := true;
  _YearOK[1, 18, 27] := true;
  _YearOK[0, 19, 27] := true;
  _YearOK[1, 19, 27] := true;
```

```
  _YearOK[0, 20, 27] := true;
  _YearOK[1, 20, 27] := true;
  _YearOK[0, 21, 27] := true;
  _YearOK[1, 21, 27] := true;
  _YearOK[0, 22, 27] := true;
  _YearOK[1, 22, 27] := true;
  _YearOK[0, 23, 27] := true;
  _YearOK[1, 23, 27] := true;
  _YearOK[0, 24, 27] := true;
  _YearOK[1, 24, 27] := true;
  _YearOK[0, 25, 27] := true;
  _YearOK[1, 25, 27] := true;
  _YearOK[0, 26, 27] := true;
  _YearOK[1, 26, 27] := true;
  _YearOK[0, 27, 27] := true;
  _YearOK[1, 27, 27] := true;
  _YearOK[0, 28, 27] := true;
  _YearOK[1, 28, 27] := true;
  _YearOK[0, 29, 27] := true;
  _YearOK[1, 29, 27] := true;
  _YearOK[0, 30, 27] := true;
  _YearOK[1, 30, 27] := true;
  _YearOK[0, 31, 27] := true;
  _YearOK[1, 31, 27] := true;
  _YearOK[0, 32, 27] := true;
  _YearOK[1, 32, 27] := true;
  _YearOK[0, 33, 27] := true;
  _YearOK[1, 33, 27] := true;
  _YearOK[0, 34, 27] := true;
  _YearOK[1, 34, 27] := true;
  _YearOK[0, 35, 27] := true;
  _YearOK[1, 35, 27] := true;
  _YearOK[0, 36, 27] := true;
  _YearOK[1, 36, 27] := true;
  _YearOK[0, 37, 27] := true;
  _YearOK[1, 37, 27] := true;
  _YearOK[0, 38, 27] := true;
  _YearOK[1, 38, 27] := true;
  _YearOK[0, 39, 27] := true;
  _YearOK[1, 39, 27] := true;
  _YearOK[0, 40, 27] := true;
  _YearOK[1, 40, 27] := true;
  _YearOK[0, 41, 27] := true;
  _YearOK[1, 41, 27] := true;
  _YearOK[0, 42, 27] := true;
  _YearOK[1, 42, 27] := true;
  _YearOK[0, 43, 27] := true;
  _YearOK[1, 43, 27] := true;
  _YearOK[0, 44, 27] := true;
  _YearOK[1, 44, 27] := true;
  _YearOK[0, 45, 27] := true;
  _YearOK[1, 45, 27] := true;
  _YearOK[0, 46, 27] := true;
  _YearOK[1, 46, 27] := true;
  _YearOK[0, 47, 27] := true;
  _YearOK[1, 47, 27] := true;
  _YearOK[0, 48, 27] := true;
  _YearOK[1, 48, 27] := true;
  _YearOK[0, 49, 27] := true;
  _YearOK[1, 49, 27] := true;
  _YearOK[0, 50, 27] := true;
  _YearOK[1, 50, 27] := true;
  _YearOK[0, 51, 27] := true;
  _YearOK[1, 51, 27] := true;
  _YearOK[0, 52, 27] := true;
  _YearOK[1, 52, 27] := true;
  _YearOK[0, 53, 27] := true;
  _YearOK[1, 53, 27] := true;
  _YearOK[0, 54, 27] := true;
  _YearOK[1, 54, 27] := true;
  _YearOK[0, 55, 27] := true;
  _YearOK[1, 55, 27] := true;
  _YearOK[0, 56, 27] := true;
  _YearOK[1, 56, 27] := true;
  _YearOK[0, 57, 27] := true;
```

```
  _YearOK[1, 57, 27] := true;
  _YearOK[0, 58, 27] := true;
  _YearOK[1, 58, 27] := true;
  _YearOK[0, 59, 27] := true;
  _YearOK[1, 59, 27] := true;
  _YearOK[0, 60, 27] := true;
  _YearOK[1, 60, 27] := true;
  _YearOK[0, 61, 27] := true;
  _YearOK[1, 61, 27] := true;
  _YearOK[0, 62, 27] := true;
  _YearOK[1, 62, 27] := true;
  _YearOK[0, 63, 27] := true;
  _YearOK[1, 63, 27] := true;
  _YearOK[0, 64, 27] := true;
  _YearOK[1, 64, 27] := true;
  _YearOK[0, 65, 27] := true;
  _YearOK[1, 65, 27] := true;
  _YearOK[0, 66, 27] := true;
  _YearOK[1, 66, 27] := true;
  _YearOK[0, 67, 27] := true;
  _YearOK[1, 67, 27] := true;
  _YearOK[0, 68, 27] := true;
  _YearOK[1, 68, 27] := true;
  _YearOK[0, 69, 27] := true;
  _YearOK[1, 69, 27] := true;
  _YearOK[0, 70, 27] := true;
  _YearOK[1, 70, 27] := true;
  _YearOK[0, 71, 27] := true;
  _YearOK[1, 71, 27] := true;
  _YearOK[0, 72, 27] := true;
  _YearOK[1, 72, 27] := true;
  _YearOK[0, 73, 27] := true;
  _YearOK[1, 73, 27] := true;
  _YearOK[0, 74, 27] := true;
  _YearOK[1, 74, 27] := true;
  _YearOK[0, 75, 27] := true;
  _YearOK[1, 75, 27] := true;
  _YearOK[0, 76, 27] := true;
  _YearOK[1, 76, 27] := true;
  _YearOK[0, 77, 27] := true;
  _YearOK[1, 77, 27] := true;
  _YearOK[0, 78, 27] := true;
  _YearOK[1, 78, 27] := true;
  _YearOK[0, 79, 27] := true;
  _YearOK[1, 79, 27] := true;
  _YearOK[0, 80, 27] := true;
  _YearOK[1, 80, 27] := true;
  _YearOK[0, 81, 27] := true;
  _YearOK[1, 81, 27] := true;
  _YearOK[0, 82, 27] := true;
  _YearOK[1, 82, 27] := true;
  _YearOK[0, 83, 27] := true;
  _YearOK[1, 83, 27] := true;
  _YearOK[0, 84, 27] := true;
  _YearOK[1, 84, 27] := true;
  _YearOK[0, 85, 27] := true;
  _YearOK[1, 85, 27] := true;
  _YearOK[0, 86, 27] := true;
  _YearOK[1, 86, 27] := true;
  _YearOK[0, 87, 27] := true;
  _YearOK[1, 87, 27] := true;
  _YearOK[0, 88, 27] := true;
  _YearOK[1, 88, 27] := true;
  _YearOK[0, 89, 27] := true;
  _YearOK[1, 89, 27] := true;
  _YearOK[0, 90, 27] := true;
  _YearOK[1, 90, 27] := true;
  _YearOK[0, 91, 27] := true;
  _YearOK[1, 91, 27] := true;
  _YearOK[0, 92, 27] := true;
  _YearOK[1, 92, 27] := true;
  _YearOK[0, 93, 27] := true;
  _YearOK[1, 93, 27] := true;
  _YearOK[0, 94, 27] := true;
```

```
  _YearOK[1, 94, 27] := true;
  _YearOK[0, 95, 27] := true;
  _YearOK[1, 95, 27] := true;
  _YearOK[0, 96, 27] := true;
  _YearOK[1, 96, 27] := true;
  _YearOK[0, 97, 27] := true;
  _YearOK[1, 97, 27] := true;
  _YearOK[0, 98, 27] := true;
  _YearOK[1, 98, 27] := true;
  _YearOK[0, 99, 27] := true;
  _YearOK[1, 99, 27] := true;
  _YearOK[0, 100, 27] := true;
  _YearOK[1, 100, 27] := true;
  _YearOK[0, 3, 28] := true;
  _YearOK[1, 3, 28] := true;
  _YearOK[0, 4, 28] := true;
  _YearOK[1, 4, 28] := true;
  _YearOK[0, 5, 28] := true;
  _YearOK[1, 5, 28] := true;
  _YearOK[0, 6, 28] := true;
  _YearOK[1, 6, 28] := true;
  _YearOK[0, 7, 28] := true;
  _YearOK[1, 7, 28] := true;
  _YearOK[0, 8, 28] := true;
  _YearOK[1, 8, 28] := true;
  _YearOK[0, 9, 28] := true;
  _YearOK[1, 9, 28] := true;
  _YearOK[0, 10, 28] := true;
  _YearOK[1, 10, 28] := true;
  _YearOK[0, 11, 28] := true;
  _YearOK[1, 11, 28] := true;
  _YearOK[0, 12, 28] := true;
  _YearOK[1, 12, 28] := true;
  _YearOK[0, 13, 28] := true;
  _YearOK[1, 13, 28] := true;
  _YearOK[0, 14, 28] := true;
  _YearOK[1, 14, 28] := true;
  _YearOK[0, 15, 28] := true;
  _YearOK[1, 15, 28] := true;
  _YearOK[0, 16, 28] := true;
  _YearOK[1, 16, 28] := true;
  _YearOK[0, 17, 28] := true;
  _YearOK[1, 17, 28] := true;
  _YearOK[0, 18, 28] := true;
  _YearOK[1, 18, 28] := true;
  _YearOK[0, 19, 28] := true;
  _YearOK[1, 19, 28] := true;
  _YearOK[0, 20, 28] := true;
  _YearOK[1, 20, 28] := true;
  _YearOK[0, 21, 28] := true;
  _YearOK[1, 21, 28] := true;
  _YearOK[0, 22, 28] := true;
  _YearOK[1, 22, 28] := true;
  _YearOK[0, 23, 28] := true;
  _YearOK[1, 23, 28] := true;
  _YearOK[0, 24, 28] := true;
  _YearOK[1, 24, 28] := true;
  _YearOK[0, 25, 28] := true;
  _YearOK[1, 25, 28] := true;
  _YearOK[0, 26, 28] := true;
  _YearOK[1, 26, 28] := true;
  _YearOK[0, 27, 28] := true;
  _YearOK[1, 27, 28] := true;
  _YearOK[0, 28, 28] := true;
  _YearOK[1, 28, 28] := true;
  _YearOK[0, 29, 28] := true;
  _YearOK[1, 29, 28] := true;
  _YearOK[0, 30, 28] := true;
  _YearOK[1, 30, 28] := true;
  _YearOK[0, 31, 28] := true;
  _YearOK[1, 31, 28] := true;
  _YearOK[0, 32, 28] := true;
  _YearOK[1, 32, 28] := true;
  _YearOK[0, 33, 28] := true;
  _YearOK[1, 33, 28] := true;
```

```
  _YearOK[0, 34, 28] := true;
  _YearOK[1, 34, 28] := true;
  _YearOK[0, 35, 28] := true;
  _YearOK[1, 35, 28] := true;
  _YearOK[0, 36, 28] := true;
  _YearOK[1, 36, 28] := true;
  _YearOK[0, 37, 28] := true;
  _YearOK[1, 37, 28] := true;
  _YearOK[0, 38, 28] := true;
  _YearOK[1, 38, 28] := true;
  _YearOK[0, 39, 28] := true;
  _YearOK[1, 39, 28] := true;
  _YearOK[0, 40, 28] := true;
  _YearOK[1, 40, 28] := true;
  _YearOK[0, 41, 28] := true;
  _YearOK[1, 41, 28] := true;
  _YearOK[0, 42, 28] := true;
  _YearOK[1, 42, 28] := true;
  _YearOK[0, 43, 28] := true;
  _YearOK[1, 43, 28] := true;
  _YearOK[0, 44, 28] := true;
  _YearOK[1, 44, 28] := true;
  _YearOK[0, 45, 28] := true;
  _YearOK[1, 45, 28] := true;
  _YearOK[0, 46, 28] := true;
  _YearOK[1, 46, 28] := true;
  _YearOK[0, 47, 28] := true;
  _YearOK[1, 47, 28] := true;
  _YearOK[0, 48, 28] := true;
  _YearOK[1, 48, 28] := true;
  _YearOK[0, 49, 28] := true;
  _YearOK[1, 49, 28] := true;
  _YearOK[0, 50, 28] := true;
  _YearOK[1, 50, 28] := true;
  _YearOK[0, 51, 28] := true;
  _YearOK[1, 51, 28] := true;
  _YearOK[0, 52, 28] := true;
  _YearOK[1, 52, 28] := true;
  _YearOK[0, 53, 28] := true;
  _YearOK[1, 53, 28] := true;
  _YearOK[0, 54, 28] := true;
  _YearOK[1, 54, 28] := true;
  _YearOK[0, 55, 28] := true;
  _YearOK[1, 55, 28] := true;
  _YearOK[0, 56, 28] := true;
  _YearOK[1, 56, 28] := true;
  _YearOK[0, 57, 28] := true;
  _YearOK[1, 57, 28] := true;
  _YearOK[0, 58, 28] := true;
  _YearOK[1, 58, 28] := true;
  _YearOK[0, 59, 28] := true;
  _YearOK[1, 59, 28] := true;
  _YearOK[0, 60, 28] := true;
  _YearOK[1, 60, 28] := true;
  _YearOK[0, 61, 28] := true;
  _YearOK[1, 61, 28] := true;
  _YearOK[0, 62, 28] := true;
  _YearOK[1, 62, 28] := true;
  _YearOK[0, 63, 28] := true;
  _YearOK[1, 63, 28] := true;
  _YearOK[0, 64, 28] := true;
  _YearOK[1, 64, 28] := true;
  _YearOK[0, 65, 28] := true;
  _YearOK[1, 65, 28] := true;
  _YearOK[0, 66, 28] := true;
  _YearOK[1, 66, 28] := true;
  _YearOK[0, 67, 28] := true;
  _YearOK[1, 67, 28] := true;
  _YearOK[0, 68, 28] := true;
  _YearOK[1, 68, 28] := true;
  _YearOK[0, 69, 28] := true;
  _YearOK[1, 69, 28] := true;
  _YearOK[0, 70, 28] := true;
  _YearOK[1, 70, 28] := true;
```

```
_YearOK[0, 71, 28] := true;
_YearOK[1, 71, 28] := true;
_YearOK[0, 72, 28] := true;
_YearOK[1, 72, 28] := true;
_YearOK[0, 73, 28] := true;
_YearOK[1, 73, 28] := true;
_YearOK[0, 74, 28] := true;
_YearOK[1, 74, 28] := true;
_YearOK[0, 75, 28] := true;
_YearOK[1, 75, 28] := true;
_YearOK[0, 76, 28] := true;
_YearOK[1, 76, 28] := true;
_YearOK[0, 77, 28] := true;
_YearOK[1, 77, 28] := true;
_YearOK[0, 78, 28] := true;
_YearOK[1, 78, 28] := true;
_YearOK[0, 79, 28] := true;
_YearOK[1, 79, 28] := true;
_YearOK[0, 80, 28] := true;
_YearOK[1, 80, 28] := true;
_YearOK[0, 81, 28] := true;
_YearOK[1, 81, 28] := true;
_YearOK[0, 82, 28] := true;
_YearOK[1, 82, 28] := true;
_YearOK[0, 83, 28] := true;
_YearOK[1, 83, 28] := true;
_YearOK[0, 84, 28] := true;
_YearOK[1, 84, 28] := true;
_YearOK[0, 85, 28] := true;
_YearOK[1, 85, 28] := true;
_YearOK[0, 86, 28] := true;
_YearOK[1, 86, 28] := true;
_YearOK[0, 87, 28] := true;
_YearOK[1, 87, 28] := true;
_YearOK[0, 88, 28] := true;
_YearOK[1, 88, 28] := true;
_YearOK[0, 89, 28] := true;
_YearOK[1, 89, 28] := true;
_YearOK[0, 90, 28] := true;
_YearOK[1, 90, 28] := true;
_YearOK[0, 91, 28] := true;
_YearOK[1, 91, 28] := true;
_YearOK[0, 92, 28] := true;
_YearOK[1, 92, 28] := true;
_YearOK[0, 93, 28] := true;
_YearOK[1, 93, 28] := true;
_YearOK[0, 94, 28] := true;
_YearOK[1, 94, 28] := true;
_YearOK[0, 95, 28] := true;
_YearOK[1, 95, 28] := true;
_YearOK[0, 96, 28] := true;
_YearOK[1, 96, 28] := true;
_YearOK[0, 97, 28] := true;
_YearOK[1, 97, 28] := true;
_YearOK[0, 98, 28] := true;
_YearOK[1, 98, 28] := true;
_YearOK[0, 99, 28] := true;
_YearOK[1, 99, 28] := true;
_YearOK[0, 100, 28] := true;
_YearOK[1, 100, 28] := true;
_YearOK[0, 3, 29] := true;
_YearOK[1, 3, 29] := true;
_YearOK[0, 4, 29] := true;
_YearOK[1, 4, 29] := true;
_YearOK[0, 5, 29] := true;
_YearOK[1, 5, 29] := true;
_YearOK[0, 6, 29] := true;
_YearOK[1, 6, 29] := true;
_YearOK[0, 7, 29] := true;
_YearOK[1, 7, 29] := true;
_YearOK[0, 8, 29] := true;
_YearOK[1, 8, 29] := true;
_YearOK[0, 9, 29] := true;
_YearOK[1, 9, 29] := true;
_YearOK[0, 10, 29] := true;
```

```
  _YearOK[1, 10, 29] := true;
  _YearOK[0, 11, 29] := true;
  _YearOK[1, 11, 29] := true;
  _YearOK[0, 12, 29] := true;
  _YearOK[1, 12, 29] := true;
  _YearOK[0, 13, 29] := true;
  _YearOK[1, 13, 29] := true;
  _YearOK[0, 14, 29] := true;
  _YearOK[1, 14, 29] := true;
  _YearOK[0, 15, 29] := true;
  _YearOK[1, 15, 29] := true;
  _YearOK[0, 16, 29] := true;
  _YearOK[1, 16, 29] := true;
  _YearOK[0, 17, 29] := true;
  _YearOK[1, 17, 29] := true;
  _YearOK[0, 18, 29] := true;
  _YearOK[1, 18, 29] := true;
  _YearOK[0, 19, 29] := true;
  _YearOK[1, 19, 29] := true;
  _YearOK[0, 20, 29] := true;
  _YearOK[1, 20, 29] := true;
  _YearOK[0, 21, 29] := true;
  _YearOK[1, 21, 29] := true;
  _YearOK[0, 22, 29] := true;
  _YearOK[1, 22, 29] := true;
  _YearOK[0, 23, 29] := true;
  _YearOK[1, 23, 29] := true;
  _YearOK[0, 24, 29] := true;
  _YearOK[1, 24, 29] := true;
  _YearOK[0, 25, 29] := true;
  _YearOK[1, 25, 29] := true;
  _YearOK[0, 26, 29] := true;
  _YearOK[1, 26, 29] := true;
  _YearOK[0, 27, 29] := true;
  _YearOK[1, 27, 29] := true;
  _YearOK[0, 28, 29] := true;
  _YearOK[1, 28, 29] := true;
  _YearOK[0, 29, 29] := true;
  _YearOK[1, 29, 29] := true;
  _YearOK[0, 30, 29] := true;
  _YearOK[1, 30, 29] := true;
  _YearOK[0, 31, 29] := true;
  _YearOK[1, 31, 29] := true;
  _YearOK[0, 32, 29] := true;
  _YearOK[1, 32, 29] := true;
  _YearOK[0, 33, 29] := true;
  _YearOK[1, 33, 29] := true;
  _YearOK[0, 34, 29] := true;
  _YearOK[1, 34, 29] := true;
  _YearOK[0, 35, 29] := true;
  _YearOK[1, 35, 29] := true;
  _YearOK[0, 36, 29] := true;
  _YearOK[1, 36, 29] := true;
  _YearOK[0, 37, 29] := true;
  _YearOK[1, 37, 29] := true;
  _YearOK[0, 38, 29] := true;
  _YearOK[1, 38, 29] := true;
  _YearOK[0, 39, 29] := true;
  _YearOK[1, 39, 29] := true;
  _YearOK[0, 40, 29] := true;
  _YearOK[1, 40, 29] := true;
  _YearOK[0, 41, 29] := true;
  _YearOK[1, 41, 29] := true;
  _YearOK[0, 42, 29] := true;
  _YearOK[1, 42, 29] := true;
  _YearOK[0, 43, 29] := true;
  _YearOK[1, 43, 29] := true;
  _YearOK[0, 44, 29] := true;
  _YearOK[1, 44, 29] := true;
  _YearOK[0, 45, 29] := true;
  _YearOK[1, 45, 29] := true;
  _YearOK[0, 46, 29] := true;
  _YearOK[1, 46, 29] := true;
  _YearOK[0, 47, 29] := true;
```

```
  _YearOK[1, 47, 29] := true;
  _YearOK[0, 48, 29] := true;
  _YearOK[1, 48, 29] := true;
  _YearOK[0, 49, 29] := true;
  _YearOK[1, 49, 29] := true;
  _YearOK[0, 50, 29] := true;
  _YearOK[1, 50, 29] := true;
  _YearOK[0, 51, 29] := true;
  _YearOK[1, 51, 29] := true;
  _YearOK[0, 52, 29] := true;
  _YearOK[1, 52, 29] := true;
  _YearOK[0, 53, 29] := true;
  _YearOK[1, 53, 29] := true;
  _YearOK[0, 54, 29] := true;
  _YearOK[1, 54, 29] := true;
  _YearOK[0, 55, 29] := true;
  _YearOK[1, 55, 29] := true;
  _YearOK[0, 56, 29] := true;
  _YearOK[1, 56, 29] := true;
  _YearOK[0, 57, 29] := true;
  _YearOK[1, 57, 29] := true;
  _YearOK[0, 58, 29] := true;
  _YearOK[1, 58, 29] := true;
  _YearOK[0, 59, 29] := true;
  _YearOK[1, 59, 29] := true;
  _YearOK[0, 60, 29] := true;
  _YearOK[1, 60, 29] := true;
  _YearOK[0, 61, 29] := true;
  _YearOK[1, 61, 29] := true;
  _YearOK[0, 62, 29] := true;
  _YearOK[1, 62, 29] := true;
  _YearOK[0, 63, 29] := true;
  _YearOK[1, 63, 29] := true;
  _YearOK[0, 64, 29] := true;
  _YearOK[1, 64, 29] := true;
  _YearOK[0, 65, 29] := true;
  _YearOK[1, 65, 29] := true;
  _YearOK[0, 66, 29] := true;
  _YearOK[1, 66, 29] := true;
  _YearOK[0, 67, 29] := true;
  _YearOK[1, 67, 29] := true;
  _YearOK[0, 68, 29] := true;
  _YearOK[1, 68, 29] := true;
  _YearOK[0, 69, 29] := true;
  _YearOK[1, 69, 29] := true;
  _YearOK[0, 70, 29] := true;
  _YearOK[1, 70, 29] := true;
  _YearOK[0, 71, 29] := true;
  _YearOK[1, 71, 29] := true;
  _YearOK[0, 72, 29] := true;
  _YearOK[1, 72, 29] := true;
  _YearOK[0, 73, 29] := true;
  _YearOK[1, 73, 29] := true;
  _YearOK[0, 74, 29] := true;
  _YearOK[1, 74, 29] := true;
  _YearOK[0, 75, 29] := true;
  _YearOK[1, 75, 29] := true;
  _YearOK[0, 76, 29] := true;
  _YearOK[1, 76, 29] := true;
  _YearOK[0, 77, 29] := true;
  _YearOK[1, 77, 29] := true;
  _YearOK[0, 78, 29] := true;
  _YearOK[1, 78, 29] := true;
  _YearOK[0, 79, 29] := true;
  _YearOK[1, 79, 29] := true;
  _YearOK[0, 80, 29] := true;
  _YearOK[1, 80, 29] := true;
  _YearOK[0, 81, 29] := true;
  _YearOK[1, 81, 29] := true;
  _YearOK[0, 82, 29] := true;
  _YearOK[1, 82, 29] := true;
  _YearOK[0, 83, 29] := true;
  _YearOK[1, 83, 29] := true;
  _YearOK[0, 84, 29] := true;
  _YearOK[1, 84, 29] := true;
```

```
  _YearOK[0, 85, 29] := true;
  _YearOK[1, 85, 29] := true;
  _YearOK[0, 86, 29] := true;
  _YearOK[1, 86, 29] := true;
  _YearOK[0, 87, 29] := true;
  _YearOK[1, 87, 29] := true;
  _YearOK[0, 88, 29] := true;
  _YearOK[1, 88, 29] := true;
  _YearOK[0, 89, 29] := true;
  _YearOK[1, 89, 29] := true;
  _YearOK[0, 90, 29] := true;
  _YearOK[1, 90, 29] := true;
  _YearOK[0, 91, 29] := true;
  _YearOK[1, 91, 29] := true;
  _YearOK[0, 92, 29] := true;
  _YearOK[1, 92, 29] := true;
  _YearOK[0, 93, 29] := true;
  _YearOK[1, 93, 29] := true;
  _YearOK[0, 94, 29] := true;
  _YearOK[1, 94, 29] := true;
  _YearOK[0, 95, 29] := true;
  _YearOK[1, 95, 29] := true;
  _YearOK[0, 96, 29] := true;
  _YearOK[1, 96, 29] := true;
  _YearOK[0, 97, 29] := true;
  _YearOK[1, 97, 29] := true;
  _YearOK[0, 98, 29] := true;
  _YearOK[1, 98, 29] := true;
  _YearOK[0, 99, 29] := true;
  _YearOK[1, 99, 29] := true;
  _YearOK[0, 100, 29] := true;
  _YearOK[1, 100, 29] := true;
  _YearOK[0, 3, 30] := true;
  _YearOK[1, 3, 30] := true;
  _YearOK[0, 4, 30] := true;
  _YearOK[1, 4, 30] := true;
  _YearOK[0, 5, 30] := true;
  _YearOK[1, 5, 30] := true;
  _YearOK[0, 6, 30] := true;
  _YearOK[1, 6, 30] := true;
  _YearOK[0, 7, 30] := true;
  _YearOK[1, 7, 30] := true;
  _YearOK[0, 8, 30] := true;
  _YearOK[1, 8, 30] := true;
  _YearOK[0, 9, 30] := true;
  _YearOK[1, 9, 30] := true;
  _YearOK[0, 10, 30] := true;
  _YearOK[1, 10, 30] := true;
  _YearOK[0, 11, 30] := true;
  _YearOK[1, 11, 30] := true;
  _YearOK[0, 12, 30] := true;
  _YearOK[1, 12, 30] := true;
  _YearOK[0, 13, 30] := true;
  _YearOK[1, 13, 30] := true;
  _YearOK[0, 14, 30] := true;
  _YearOK[1, 14, 30] := true;
  _YearOK[0, 15, 30] := true;
  _YearOK[1, 15, 30] := true;
  _YearOK[0, 16, 30] := true;
  _YearOK[1, 16, 30] := true;
  _YearOK[0, 17, 30] := true;
  _YearOK[1, 17, 30] := true;
  _YearOK[0, 18, 30] := true;
  _YearOK[1, 18, 30] := true;
  _YearOK[0, 19, 30] := true;
  _YearOK[1, 19, 30] := true;
  _YearOK[0, 20, 30] := true;
  _YearOK[1, 20, 30] := true;
  _YearOK[0, 21, 30] := true;
  _YearOK[1, 21, 30] := true;
  _YearOK[0, 22, 30] := true;
  _YearOK[1, 22, 30] := true;
  _YearOK[0, 23, 30] := true;
  _YearOK[1, 23, 30] := true;
```

```
  _YearOK[0, 24, 30] := true;
  _YearOK[1, 24, 30] := true;
  _YearOK[0, 25, 30] := true;
  _YearOK[1, 25, 30] := true;
  _YearOK[0, 26, 30] := true;
  _YearOK[1, 26, 30] := true;
  _YearOK[0, 27, 30] := true;
  _YearOK[1, 27, 30] := true;
  _YearOK[0, 28, 30] := true;
  _YearOK[1, 28, 30] := true;
  _YearOK[0, 29, 30] := true;
  _YearOK[1, 29, 30] := true;
  _YearOK[0, 30, 30] := true;
  _YearOK[1, 30, 30] := true;
  _YearOK[0, 31, 30] := true;
  _YearOK[1, 31, 30] := true;
  _YearOK[0, 32, 30] := true;
  _YearOK[1, 32, 30] := true;
  _YearOK[0, 33, 30] := true;
  _YearOK[1, 33, 30] := true;
  _YearOK[0, 34, 30] := true;
  _YearOK[1, 34, 30] := true;
  _YearOK[0, 35, 30] := true;
  _YearOK[1, 35, 30] := true;
  _YearOK[0, 36, 30] := true;
  _YearOK[1, 36, 30] := true;
  _YearOK[0, 37, 30] := true;
  _YearOK[1, 37, 30] := true;
  _YearOK[0, 38, 30] := true;
  _YearOK[1, 38, 30] := true;
  _YearOK[0, 39, 30] := true;
  _YearOK[1, 39, 30] := true;
  _YearOK[0, 40, 30] := true;
  _YearOK[1, 40, 30] := true;
  _YearOK[0, 41, 30] := true;
  _YearOK[1, 41, 30] := true;
  _YearOK[0, 42, 30] := true;
  _YearOK[1, 42, 30] := true;
  _YearOK[0, 43, 30] := true;
  _YearOK[1, 43, 30] := true;
  _YearOK[0, 44, 30] := true;
  _YearOK[1, 44, 30] := true;
  _YearOK[0, 45, 30] := true;
  _YearOK[1, 45, 30] := true;
  _YearOK[0, 46, 30] := true;
  _YearOK[1, 46, 30] := true;
  _YearOK[0, 47, 30] := true;
  _YearOK[1, 47, 30] := true;
  _YearOK[0, 48, 30] := true;
  _YearOK[1, 48, 30] := true;
  _YearOK[0, 49, 30] := true;
  _YearOK[1, 49, 30] := true;
  _YearOK[0, 50, 30] := true;
  _YearOK[1, 50, 30] := true;
  _YearOK[0, 51, 30] := true;
  _YearOK[1, 51, 30] := true;
  _YearOK[0, 52, 30] := true;
  _YearOK[1, 52, 30] := true;
  _YearOK[0, 53, 30] := true;
  _YearOK[1, 53, 30] := true;
  _YearOK[0, 54, 30] := true;
  _YearOK[1, 54, 30] := true;
  _YearOK[0, 55, 30] := true;
  _YearOK[1, 55, 30] := true;
  _YearOK[0, 56, 30] := true;
  _YearOK[1, 56, 30] := true;
  _YearOK[0, 57, 30] := true;
  _YearOK[1, 57, 30] := true;
  _YearOK[0, 58, 30] := true;
  _YearOK[1, 58, 30] := true;
  _YearOK[0, 59, 30] := true;
  _YearOK[1, 59, 30] := true;
  _YearOK[0, 60, 30] := true;
  _YearOK[1, 60, 30] := true;
  _YearOK[0, 61, 30] := true;
```

```
  _YearOK[1, 61, 30] := true;
  _YearOK[0, 62, 30] := true;
  _YearOK[1, 62, 30] := true;
  _YearOK[0, 63, 30] := true;
  _YearOK[1, 63, 30] := true;
  _YearOK[0, 64, 30] := true;
  _YearOK[1, 64, 30] := true;
  _YearOK[0, 65, 30] := true;
  _YearOK[1, 65, 30] := true;
  _YearOK[0, 66, 30] := true;
  _YearOK[1, 66, 30] := true;
  _YearOK[0, 67, 30] := true;
  _YearOK[1, 67, 30] := true;
  _YearOK[0, 68, 30] := true;
  _YearOK[1, 68, 30] := true;
  _YearOK[0, 69, 30] := true;
  _YearOK[1, 69, 30] := true;
  _YearOK[0, 70, 30] := true;
  _YearOK[1, 70, 30] := true;
  _YearOK[0, 71, 30] := true;
  _YearOK[1, 71, 30] := true;
  _YearOK[0, 72, 30] := true;
  _YearOK[1, 72, 30] := true;
  _YearOK[0, 73, 30] := true;
  _YearOK[1, 73, 30] := true;
  _YearOK[0, 74, 30] := true;
  _YearOK[1, 74, 30] := true;
  _YearOK[0, 75, 30] := true;
  _YearOK[1, 75, 30] := true;
  _YearOK[0, 76, 30] := true;
  _YearOK[1, 76, 30] := true;
  _YearOK[0, 77, 30] := true;
  _YearOK[1, 77, 30] := true;
  _YearOK[0, 78, 30] := true;
  _YearOK[1, 78, 30] := true;
  _YearOK[0, 79, 30] := true;
  _YearOK[1, 79, 30] := true;
  _YearOK[0, 80, 30] := true;
  _YearOK[1, 80, 30] := true;
  _YearOK[0, 81, 30] := true;
  _YearOK[1, 81, 30] := true;
  _YearOK[0, 82, 30] := true;
  _YearOK[1, 82, 30] := true;
  _YearOK[0, 83, 30] := true;
  _YearOK[1, 83, 30] := true;
  _YearOK[0, 84, 30] := true;
  _YearOK[1, 84, 30] := true;
  _YearOK[0, 85, 30] := true;
  _YearOK[1, 85, 30] := true;
  _YearOK[0, 86, 30] := true;
  _YearOK[1, 86, 30] := true;
  _YearOK[0, 87, 30] := true;
  _YearOK[1, 87, 30] := true;
  _YearOK[0, 88, 30] := true;
  _YearOK[1, 88, 30] := true;
  _YearOK[0, 89, 30] := true;
  _YearOK[1, 89, 30] := true;
  _YearOK[0, 90, 30] := true;
  _YearOK[1, 90, 30] := true;
  _YearOK[0, 91, 30] := true;
  _YearOK[1, 91, 30] := true;
  _YearOK[0, 92, 30] := true;
  _YearOK[1, 92, 30] := true;
  _YearOK[0, 93, 30] := true;
  _YearOK[1, 93, 30] := true;
  _YearOK[0, 94, 30] := true;
  _YearOK[1, 94, 30] := true;
  _YearOK[0, 95, 30] := true;
  _YearOK[1, 95, 30] := true;
  _YearOK[0, 96, 30] := true;
  _YearOK[1, 96, 30] := true;
  _YearOK[0, 97, 30] := true;
  _YearOK[1, 97, 30] := true;
  _YearOK[0, 98, 30] := true;
```

```
  _YearOK[1, 98, 30] := true;
  _YearOK[0, 99, 30] := true;
  _YearOK[1, 99, 30] := true;
  _YearOK[0, 100, 30] := true;
  _YearOK[1, 100, 30] := true;
  _YearOK[0, 3, 31] := true;
  _YearOK[1, 3, 31] := true;
  _YearOK[0, 4, 31] := true;
  _YearOK[1, 4, 31] := true;
  _YearOK[0, 5, 31] := true;
  _YearOK[1, 5, 31] := true;
  _YearOK[0, 6, 31] := true;
  _YearOK[1, 6, 31] := true;
  _YearOK[0, 7, 31] := true;
  _YearOK[1, 7, 31] := true;
  _YearOK[0, 8, 31] := true;
  _YearOK[1, 8, 31] := true;
  _YearOK[0, 9, 31] := true;
  _YearOK[1, 9, 31] := true;
  _YearOK[0, 10, 31] := true;
  _YearOK[1, 10, 31] := true;
  _YearOK[0, 11, 31] := true;
  _YearOK[1, 11, 31] := true;
  _YearOK[0, 12, 31] := true;
  _YearOK[1, 12, 31] := true;
  _YearOK[0, 13, 31] := true;
  _YearOK[1, 13, 31] := true;
  _YearOK[0, 14, 31] := true;
  _YearOK[1, 14, 31] := true;
  _YearOK[0, 15, 31] := true;
  _YearOK[1, 15, 31] := true;
  _YearOK[0, 16, 31] := true;
  _YearOK[1, 16, 31] := true;
  _YearOK[0, 17, 31] := true;
  _YearOK[1, 17, 31] := true;
  _YearOK[0, 18, 31] := true;
  _YearOK[1, 18, 31] := true;
  _YearOK[0, 19, 31] := true;
  _YearOK[1, 19, 31] := true;
  _YearOK[0, 20, 31] := true;
  _YearOK[1, 20, 31] := true;
  _YearOK[0, 21, 31] := true;
  _YearOK[1, 21, 31] := true;
  _YearOK[0, 22, 31] := true;
  _YearOK[1, 22, 31] := true;
  _YearOK[0, 23, 31] := true;
  _YearOK[1, 23, 31] := true;
  _YearOK[0, 24, 31] := true;
  _YearOK[1, 24, 31] := true;
  _YearOK[0, 25, 31] := true;
  _YearOK[1, 25, 31] := true;
  _YearOK[0, 26, 31] := true;
  _YearOK[1, 26, 31] := true;
  _YearOK[0, 27, 31] := true;
  _YearOK[1, 27, 31] := true;
  _YearOK[0, 28, 31] := true;
  _YearOK[1, 28, 31] := true;
  _YearOK[0, 29, 31] := true;
  _YearOK[1, 29, 31] := true;
  _YearOK[0, 30, 31] := true;
  _YearOK[1, 30, 31] := true;
  _YearOK[0, 31, 31] := true;
  _YearOK[1, 31, 31] := true;
  _YearOK[0, 32, 31] := true;
  _YearOK[1, 32, 31] := true;
  _YearOK[0, 33, 31] := true;
  _YearOK[1, 33, 31] := true;
  _YearOK[0, 34, 31] := true;
  _YearOK[1, 34, 31] := true;
  _YearOK[0, 35, 31] := true;
  _YearOK[1, 35, 31] := true;
  _YearOK[0, 36, 31] := true;
  _YearOK[1, 36, 31] := true;
  _YearOK[0, 37, 31] := true;
  _YearOK[1, 37, 31] := true;
```

```
  _YearOK[0, 38, 31] := true;
  _YearOK[1, 38, 31] := true;
  _YearOK[0, 39, 31] := true;
  _YearOK[1, 39, 31] := true;
  _YearOK[0, 40, 31] := true;
  _YearOK[1, 40, 31] := true;
  _YearOK[0, 41, 31] := true;
  _YearOK[1, 41, 31] := true;
  _YearOK[0, 42, 31] := true;
  _YearOK[1, 42, 31] := true;
  _YearOK[0, 43, 31] := true;
  _YearOK[1, 43, 31] := true;
  _YearOK[0, 44, 31] := true;
  _YearOK[1, 44, 31] := true;
  _YearOK[0, 45, 31] := true;
  _YearOK[1, 45, 31] := true;
  _YearOK[0, 46, 31] := true;
  _YearOK[1, 46, 31] := true;
  _YearOK[0, 47, 31] := true;
  _YearOK[1, 47, 31] := true;
  _YearOK[0, 48, 31] := true;
  _YearOK[1, 48, 31] := true;
  _YearOK[0, 49, 31] := true;
  _YearOK[1, 49, 31] := true;
  _YearOK[0, 50, 31] := true;
  _YearOK[1, 50, 31] := true;
  _YearOK[0, 51, 31] := true;
  _YearOK[1, 51, 31] := true;
  _YearOK[0, 52, 31] := true;
  _YearOK[1, 52, 31] := true;
  _YearOK[0, 53, 31] := true;
  _YearOK[1, 53, 31] := true;
  _YearOK[0, 54, 31] := true;
  _YearOK[1, 54, 31] := true;
  _YearOK[0, 55, 31] := true;
  _YearOK[1, 55, 31] := true;
  _YearOK[0, 56, 31] := true;
  _YearOK[1, 56, 31] := true;
  _YearOK[0, 57, 31] := true;
  _YearOK[1, 57, 31] := true;
  _YearOK[0, 58, 31] := true;
  _YearOK[1, 58, 31] := true;
  _YearOK[0, 59, 31] := true;
  _YearOK[1, 59, 31] := true;
  _YearOK[0, 60, 31] := true;
  _YearOK[1, 60, 31] := true;
  _YearOK[0, 61, 31] := true;
  _YearOK[1, 61, 31] := true;
  _YearOK[0, 62, 31] := true;
  _YearOK[1, 62, 31] := true;
  _YearOK[0, 63, 31] := true;
  _YearOK[1, 63, 31] := true;
  _YearOK[0, 64, 31] := true;
  _YearOK[1, 64, 31] := true;
  _YearOK[0, 65, 31] := true;
  _YearOK[1, 65, 31] := true;
  _YearOK[0, 66, 31] := true;
  _YearOK[1, 66, 31] := true;
  _YearOK[0, 67, 31] := true;
  _YearOK[1, 67, 31] := true;
  _YearOK[0, 68, 31] := true;
  _YearOK[1, 68, 31] := true;
  _YearOK[0, 69, 31] := true;
  _YearOK[1, 69, 31] := true;
  _YearOK[0, 70, 31] := true;
  _YearOK[1, 70, 31] := true;
  _YearOK[0, 71, 31] := true;
  _YearOK[1, 71, 31] := true;
  _YearOK[0, 72, 31] := true;
  _YearOK[1, 72, 31] := true;
  _YearOK[0, 73, 31] := true;
  _YearOK[1, 73, 31] := true;
  _YearOK[0, 74, 31] := true;
  _YearOK[1, 74, 31] := true;
```

```
_YearOK[0, 75, 31] := true;
_YearOK[1, 75, 31] := true;
_YearOK[0, 76, 31] := true;
_YearOK[1, 76, 31] := true;
_YearOK[0, 77, 31] := true;
_YearOK[1, 77, 31] := true;
_YearOK[0, 78, 31] := true;
_YearOK[1, 78, 31] := true;
_YearOK[0, 79, 31] := true;
_YearOK[1, 79, 31] := true;
_YearOK[0, 80, 31] := true;
_YearOK[1, 80, 31] := true;
_YearOK[0, 81, 31] := true;
_YearOK[1, 81, 31] := true;
_YearOK[0, 82, 31] := true;
_YearOK[1, 82, 31] := true;
_YearOK[0, 83, 31] := true;
_YearOK[1, 83, 31] := true;
_YearOK[0, 84, 31] := true;
_YearOK[1, 84, 31] := true;
_YearOK[0, 85, 31] := true;
_YearOK[1, 85, 31] := true;
_YearOK[0, 86, 31] := true;
_YearOK[1, 86, 31] := true;
_YearOK[0, 87, 31] := true;
_YearOK[1, 87, 31] := true;
_YearOK[0, 88, 31] := true;
_YearOK[1, 88, 31] := true;
_YearOK[0, 89, 31] := true;
_YearOK[1, 89, 31] := true;
_YearOK[0, 90, 31] := true;
_YearOK[1, 90, 31] := true;
_YearOK[0, 91, 31] := true;
_YearOK[1, 91, 31] := true;
_YearOK[0, 92, 31] := true;
_YearOK[1, 92, 31] := true;
_YearOK[0, 93, 31] := true;
_YearOK[1, 93, 31] := true;
_YearOK[0, 94, 31] := true;
_YearOK[1, 94, 31] := true;
_YearOK[0, 95, 31] := true;
_YearOK[1, 95, 31] := true;
_YearOK[0, 96, 31] := true;
_YearOK[1, 96, 31] := true;
_YearOK[0, 97, 31] := true;
_YearOK[1, 97, 31] := true;
_YearOK[0, 98, 31] := true;
_YearOK[1, 98, 31] := true;
_YearOK[0, 99, 31] := true;
_YearOK[1, 99, 31] := true;
_YearOK[0, 100, 31] := true;
_YearOK[1, 100, 31] := true;
_YearOK[0, 3, 32] := true;
_YearOK[1, 3, 32] := true;
_YearOK[0, 4, 32] := true;
_YearOK[1, 4, 32] := true;
_YearOK[0, 5, 32] := true;
_YearOK[1, 5, 32] := true;
_YearOK[0, 6, 32] := true;
_YearOK[1, 6, 32] := true;
_YearOK[0, 7, 32] := true;
_YearOK[1, 7, 32] := true;
_YearOK[0, 8, 32] := true;
_YearOK[1, 8, 32] := true;
_YearOK[0, 9, 32] := true;
_YearOK[1, 9, 32] := true;
_YearOK[0, 10, 32] := true;
_YearOK[1, 10, 32] := true;
_YearOK[0, 11, 32] := true;
_YearOK[1, 11, 32] := true;
_YearOK[0, 12, 32] := true;
_YearOK[1, 12, 32] := true;
_YearOK[0, 13, 32] := true;
_YearOK[1, 13, 32] := true;
_YearOK[0, 14, 32] := true;
```

```
  _YearOK[1, 14, 32] := true;
  _YearOK[0, 15, 32] := true;
  _YearOK[1, 15, 32] := true;
  _YearOK[0, 16, 32] := true;
  _YearOK[1, 16, 32] := true;
  _YearOK[0, 17, 32] := true;
  _YearOK[1, 17, 32] := true;
  _YearOK[0, 18, 32] := true;
  _YearOK[1, 18, 32] := true;
  _YearOK[0, 19, 32] := true;
  _YearOK[1, 19, 32] := true;
  _YearOK[0, 20, 32] := true;
  _YearOK[1, 20, 32] := true;
  _YearOK[0, 21, 32] := true;
  _YearOK[1, 21, 32] := true;
  _YearOK[0, 22, 32] := true;
  _YearOK[1, 22, 32] := true;
  _YearOK[0, 23, 32] := true;
  _YearOK[1, 23, 32] := true;
  _YearOK[0, 24, 32] := true;
  _YearOK[1, 24, 32] := true;
  _YearOK[0, 25, 32] := true;
  _YearOK[1, 25, 32] := true;
  _YearOK[0, 26, 32] := true;
  _YearOK[1, 26, 32] := true;
  _YearOK[0, 27, 32] := true;
  _YearOK[1, 27, 32] := true;
  _YearOK[0, 28, 32] := true;
  _YearOK[1, 28, 32] := true;
  _YearOK[0, 29, 32] := true;
  _YearOK[1, 29, 32] := true;
  _YearOK[0, 30, 32] := true;
  _YearOK[1, 30, 32] := true;
  _YearOK[0, 31, 32] := true;
  _YearOK[1, 31, 32] := true;
  _YearOK[0, 32, 32] := true;
  _YearOK[1, 32, 32] := true;
  _YearOK[0, 33, 32] := true;
  _YearOK[1, 33, 32] := true;
  _YearOK[0, 34, 32] := true;
  _YearOK[1, 34, 32] := true;
  _YearOK[0, 35, 32] := true;
  _YearOK[1, 35, 32] := true;
  _YearOK[0, 36, 32] := true;
  _YearOK[1, 36, 32] := true;
  _YearOK[0, 37, 32] := true;
  _YearOK[1, 37, 32] := true;
  _YearOK[0, 38, 32] := true;
  _YearOK[1, 38, 32] := true;
  _YearOK[0, 39, 32] := true;
  _YearOK[1, 39, 32] := true;
  _YearOK[0, 40, 32] := true;
  _YearOK[1, 40, 32] := true;
  _YearOK[0, 41, 32] := true;
  _YearOK[1, 41, 32] := true;
  _YearOK[0, 42, 32] := true;
  _YearOK[1, 42, 32] := true;
  _YearOK[0, 43, 32] := true;
  _YearOK[1, 43, 32] := true;
  _YearOK[0, 44, 32] := true;
  _YearOK[1, 44, 32] := true;
  _YearOK[0, 45, 32] := true;
  _YearOK[1, 45, 32] := true;
  _YearOK[0, 46, 32] := true;
  _YearOK[1, 46, 32] := true;
  _YearOK[0, 47, 32] := true;
  _YearOK[1, 47, 32] := true;
  _YearOK[0, 48, 32] := true;
  _YearOK[1, 48, 32] := true;
  _YearOK[0, 49, 32] := true;
  _YearOK[1, 49, 32] := true;
  _YearOK[0, 50, 32] := true;
  _YearOK[1, 50, 32] := true;
  _YearOK[0, 51, 32] := true;
```

```
  _YearOK[1, 51, 32] := true;
  _YearOK[0, 52, 32] := true;
  _YearOK[1, 52, 32] := true;
  _YearOK[0, 53, 32] := true;
  _YearOK[1, 53, 32] := true;
  _YearOK[0, 54, 32] := true;
  _YearOK[1, 54, 32] := true;
  _YearOK[0, 55, 32] := true;
  _YearOK[1, 55, 32] := true;
  _YearOK[0, 56, 32] := true;
  _YearOK[1, 56, 32] := true;
  _YearOK[0, 57, 32] := true;
  _YearOK[1, 57, 32] := true;
  _YearOK[0, 58, 32] := true;
  _YearOK[1, 58, 32] := true;
  _YearOK[0, 59, 32] := true;
  _YearOK[1, 59, 32] := true;
  _YearOK[0, 60, 32] := true;
  _YearOK[1, 60, 32] := true;
  _YearOK[0, 61, 32] := true;
  _YearOK[1, 61, 32] := true;
  _YearOK[0, 62, 32] := true;
  _YearOK[1, 62, 32] := true;
  _YearOK[0, 63, 32] := true;
  _YearOK[1, 63, 32] := true;
  _YearOK[0, 64, 32] := true;
  _YearOK[1, 64, 32] := true;
  _YearOK[0, 65, 32] := true;
  _YearOK[1, 65, 32] := true;
  _YearOK[0, 66, 32] := true;
  _YearOK[1, 66, 32] := true;
  _YearOK[0, 67, 32] := true;
  _YearOK[1, 67, 32] := true;
  _YearOK[0, 68, 32] := true;
  _YearOK[1, 68, 32] := true;
  _YearOK[0, 69, 32] := true;
  _YearOK[1, 69, 32] := true;
  _YearOK[0, 70, 32] := true;
  _YearOK[1, 70, 32] := true;
  _YearOK[0, 71, 32] := true;
  _YearOK[1, 71, 32] := true;
  _YearOK[0, 72, 32] := true;
  _YearOK[1, 72, 32] := true;
  _YearOK[0, 73, 32] := true;
  _YearOK[1, 73, 32] := true;
  _YearOK[0, 74, 32] := true;
  _YearOK[1, 74, 32] := true;
  _YearOK[0, 75, 32] := true;
  _YearOK[1, 75, 32] := true;
  _YearOK[0, 76, 32] := true;
  _YearOK[1, 76, 32] := true;
  _YearOK[0, 77, 32] := true;
  _YearOK[1, 77, 32] := true;
  _YearOK[0, 78, 32] := true;
  _YearOK[1, 78, 32] := true;
  _YearOK[0, 79, 32] := true;
  _YearOK[1, 79, 32] := true;
  _YearOK[0, 80, 32] := true;
  _YearOK[1, 80, 32] := true;
  _YearOK[0, 81, 32] := true;
  _YearOK[1, 81, 32] := true;
  _YearOK[0, 82, 32] := true;
  _YearOK[1, 82, 32] := true;
  _YearOK[0, 83, 32] := true;
  _YearOK[1, 83, 32] := true;
  _YearOK[0, 84, 32] := true;
  _YearOK[1, 84, 32] := true;
  _YearOK[0, 85, 32] := true;
  _YearOK[1, 85, 32] := true;
  _YearOK[0, 86, 32] := true;
  _YearOK[1, 86, 32] := true;
  _YearOK[0, 87, 32] := true;
  _YearOK[1, 87, 32] := true;
  _YearOK[0, 88, 32] := true;
  _YearOK[1, 88, 32] := true;
```

```
  _YearOK[0, 89, 32] := true;
  _YearOK[1, 89, 32] := true;
  _YearOK[0, 90, 32] := true;
  _YearOK[1, 90, 32] := true;
  _YearOK[0, 91, 32] := true;
  _YearOK[1, 91, 32] := true;
  _YearOK[0, 92, 32] := true;
  _YearOK[1, 92, 32] := true;
  _YearOK[0, 93, 32] := true;
  _YearOK[1, 93, 32] := true;
  _YearOK[0, 94, 32] := true;
  _YearOK[1, 94, 32] := true;
  _YearOK[0, 95, 32] := true;
  _YearOK[1, 95, 32] := true;
  _YearOK[0, 96, 32] := true;
  _YearOK[1, 96, 32] := true;
  _YearOK[0, 97, 32] := true;
  _YearOK[1, 97, 32] := true;
  _YearOK[0, 98, 32] := true;
  _YearOK[1, 98, 32] := true;
  _YearOK[0, 99, 32] := true;
  _YearOK[1, 99, 32] := true;
  _YearOK[0, 100, 32] := true;
  _YearOK[1, 100, 32] := true;
  _YearOK[0, 3, 33] := true;
  _YearOK[1, 3, 33] := true;
  _YearOK[0, 4, 33] := true;
  _YearOK[1, 4, 33] := true;
  _YearOK[0, 5, 33] := true;
  _YearOK[1, 5, 33] := true;
  _YearOK[0, 6, 33] := true;
  _YearOK[1, 6, 33] := true;
  _YearOK[0, 7, 33] := true;
  _YearOK[1, 7, 33] := true;
  _YearOK[0, 8, 33] := true;
  _YearOK[1, 8, 33] := true;
  _YearOK[0, 9, 33] := true;
  _YearOK[1, 9, 33] := true;
  _YearOK[0, 10, 33] := true;
  _YearOK[1, 10, 33] := true;
  _YearOK[0, 11, 33] := true;
  _YearOK[1, 11, 33] := true;
  _YearOK[0, 12, 33] := true;
  _YearOK[1, 12, 33] := true;
  _YearOK[0, 13, 33] := true;
  _YearOK[1, 13, 33] := true;
  _YearOK[0, 14, 33] := true;
  _YearOK[1, 14, 33] := true;
  _YearOK[0, 15, 33] := true;
  _YearOK[1, 15, 33] := true;
  _YearOK[0, 16, 33] := true;
  _YearOK[1, 16, 33] := true;
  _YearOK[0, 17, 33] := true;
  _YearOK[1, 17, 33] := true;
  _YearOK[0, 18, 33] := true;
  _YearOK[1, 18, 33] := true;
  _YearOK[0, 19, 33] := true;
  _YearOK[1, 19, 33] := true;
  _YearOK[0, 20, 33] := true;
  _YearOK[1, 20, 33] := true;
  _YearOK[0, 21, 33] := true;
  _YearOK[1, 21, 33] := true;
  _YearOK[0, 22, 33] := true;
  _YearOK[1, 22, 33] := true;
  _YearOK[0, 23, 33] := true;
  _YearOK[1, 23, 33] := true;
  _YearOK[0, 24, 33] := true;
  _YearOK[1, 24, 33] := true;
  _YearOK[0, 25, 33] := true;
  _YearOK[1, 25, 33] := true;
  _YearOK[0, 26, 33] := true;
  _YearOK[1, 26, 33] := true;
  _YearOK[0, 27, 33] := true;
  _YearOK[1, 27, 33] := true;
```

```
_YearOK[0, 28, 33] := true;
_YearOK[1, 28, 33] := true;
_YearOK[0, 29, 33] := true;
_YearOK[1, 29, 33] := true;
_YearOK[0, 30, 33] := true;
_YearOK[1, 30, 33] := true;
_YearOK[0, 31, 33] := true;
_YearOK[1, 31, 33] := true;
_YearOK[0, 32, 33] := true;
_YearOK[1, 32, 33] := true;
_YearOK[0, 33, 33] := true;
_YearOK[1, 33, 33] := true;
_YearOK[0, 34, 33] := true;
_YearOK[1, 34, 33] := true;
_YearOK[0, 35, 33] := true;
_YearOK[1, 35, 33] := true;
_YearOK[0, 36, 33] := true;
_YearOK[1, 36, 33] := true;
_YearOK[0, 37, 33] := true;
_YearOK[1, 37, 33] := true;
_YearOK[0, 38, 33] := true;
_YearOK[1, 38, 33] := true;
_YearOK[0, 39, 33] := true;
_YearOK[1, 39, 33] := true;
_YearOK[0, 40, 33] := true;
_YearOK[1, 40, 33] := true;
_YearOK[0, 41, 33] := true;
_YearOK[1, 41, 33] := true;
_YearOK[0, 42, 33] := true;
_YearOK[1, 42, 33] := true;
_YearOK[0, 43, 33] := true;
_YearOK[1, 43, 33] := true;
_YearOK[0, 44, 33] := true;
_YearOK[1, 44, 33] := true;
_YearOK[0, 45, 33] := true;
_YearOK[1, 45, 33] := true;
_YearOK[0, 46, 33] := true;
_YearOK[1, 46, 33] := true;
_YearOK[0, 47, 33] := true;
_YearOK[1, 47, 33] := true;
_YearOK[0, 48, 33] := true;
_YearOK[1, 48, 33] := true;
_YearOK[0, 49, 33] := true;
_YearOK[1, 49, 33] := true;
_YearOK[0, 50, 33] := true;
_YearOK[1, 50, 33] := true;
_YearOK[0, 51, 33] := true;
_YearOK[1, 51, 33] := true;
_YearOK[0, 52, 33] := true;
_YearOK[1, 52, 33] := true;
_YearOK[0, 53, 33] := true;
_YearOK[1, 53, 33] := true;
_YearOK[0, 54, 33] := true;
_YearOK[1, 54, 33] := true;
_YearOK[0, 55, 33] := true;
_YearOK[1, 55, 33] := true;
_YearOK[0, 56, 33] := true;
_YearOK[1, 56, 33] := true;
_YearOK[0, 57, 33] := true;
_YearOK[1, 57, 33] := true;
_YearOK[0, 58, 33] := true;
_YearOK[1, 58, 33] := true;
_YearOK[0, 59, 33] := true;
_YearOK[1, 59, 33] := true;
_YearOK[0, 60, 33] := true;
_YearOK[1, 60, 33] := true;
_YearOK[0, 61, 33] := true;
_YearOK[1, 61, 33] := true;
_YearOK[0, 62, 33] := true;
_YearOK[1, 62, 33] := true;
_YearOK[0, 63, 33] := true;
_YearOK[1, 63, 33] := true;
_YearOK[0, 64, 33] := true;
_YearOK[1, 64, 33] := true;
_YearOK[0, 65, 33] := true;
```

```
_YearOK[1, 65, 33] := true;
_YearOK[0, 66, 33] := true;
_YearOK[1, 66, 33] := true;
_YearOK[0, 67, 33] := true;
_YearOK[1, 67, 33] := true;
_YearOK[0, 68, 33] := true;
_YearOK[1, 68, 33] := true;
_YearOK[0, 69, 33] := true;
_YearOK[1, 69, 33] := true;
_YearOK[0, 70, 33] := true;
_YearOK[1, 70, 33] := true;
_YearOK[0, 71, 33] := true;
_YearOK[1, 71, 33] := true;
_YearOK[0, 72, 33] := true;
_YearOK[1, 72, 33] := true;
_YearOK[0, 73, 33] := true;
_YearOK[1, 73, 33] := true;
_YearOK[0, 74, 33] := true;
_YearOK[1, 74, 33] := true;
_YearOK[0, 75, 33] := true;
_YearOK[1, 75, 33] := true;
_YearOK[0, 76, 33] := true;
_YearOK[1, 76, 33] := true;
_YearOK[0, 77, 33] := true;
_YearOK[1, 77, 33] := true;
_YearOK[0, 78, 33] := true;
_YearOK[1, 78, 33] := true;
_YearOK[0, 79, 33] := true;
_YearOK[1, 79, 33] := true;
_YearOK[0, 80, 33] := true;
_YearOK[1, 80, 33] := true;
_YearOK[0, 81, 33] := true;
_YearOK[1, 81, 33] := true;
_YearOK[0, 82, 33] := true;
_YearOK[1, 82, 33] := true;
_YearOK[0, 83, 33] := true;
_YearOK[1, 83, 33] := true;
_YearOK[0, 84, 33] := true;
_YearOK[1, 84, 33] := true;
_YearOK[0, 85, 33] := true;
_YearOK[1, 85, 33] := true;
_YearOK[0, 86, 33] := true;
_YearOK[1, 86, 33] := true;
_YearOK[0, 87, 33] := true;
_YearOK[1, 87, 33] := true;
_YearOK[0, 88, 33] := true;
_YearOK[1, 88, 33] := true;
_YearOK[0, 89, 33] := true;
_YearOK[1, 89, 33] := true;
_YearOK[0, 90, 33] := true;
_YearOK[1, 90, 33] := true;
_YearOK[0, 91, 33] := true;
_YearOK[1, 91, 33] := true;
_YearOK[0, 92, 33] := true;
_YearOK[1, 92, 33] := true;
_YearOK[0, 93, 33] := true;
_YearOK[1, 93, 33] := true;
_YearOK[0, 94, 33] := true;
_YearOK[1, 94, 33] := true;
_YearOK[0, 95, 33] := true;
_YearOK[1, 95, 33] := true;
_YearOK[0, 96, 33] := true;
_YearOK[1, 96, 33] := true;
_YearOK[0, 97, 33] := true;
_YearOK[1, 97, 33] := true;
_YearOK[0, 98, 33] := true;
_YearOK[1, 98, 33] := true;
_YearOK[0, 99, 33] := true;
_YearOK[1, 99, 33] := true;
_YearOK[0, 100, 33] := true;
_YearOK[1, 100, 33] := true;
_YearOK[0, 3, 34] := true;
_YearOK[1, 3, 34] := true;
_YearOK[0, 4, 34] := true;
```

```
_YearOK[1, 4, 34] := true;
_YearOK[0, 5, 34] := true;
_YearOK[1, 5, 34] := true;
_YearOK[0, 6, 34] := true;
_YearOK[1, 6, 34] := true;
_YearOK[0, 7, 34] := true;
_YearOK[1, 7, 34] := true;
_YearOK[0, 8, 34] := true;
_YearOK[1, 8, 34] := true;
_YearOK[0, 9, 34] := true;
_YearOK[1, 9, 34] := true;
_YearOK[0, 10, 34] := true;
_YearOK[1, 10, 34] := true;
_YearOK[0, 11, 34] := true;
_YearOK[1, 11, 34] := true;
_YearOK[0, 12, 34] := true;
_YearOK[1, 12, 34] := true;
_YearOK[0, 13, 34] := true;
_YearOK[1, 13, 34] := true;
_YearOK[0, 14, 34] := true;
_YearOK[1, 14, 34] := true;
_YearOK[0, 15, 34] := true;
_YearOK[1, 15, 34] := true;
_YearOK[0, 16, 34] := true;
_YearOK[1, 16, 34] := true;
_YearOK[0, 17, 34] := true;
_YearOK[1, 17, 34] := true;
_YearOK[0, 18, 34] := true;
_YearOK[1, 18, 34] := true;
_YearOK[0, 19, 34] := true;
_YearOK[1, 19, 34] := true;
_YearOK[0, 20, 34] := true;
_YearOK[1, 20, 34] := true;
_YearOK[0, 21, 34] := true;
_YearOK[1, 21, 34] := true;
_YearOK[0, 22, 34] := true;
_YearOK[1, 22, 34] := true;
_YearOK[0, 23, 34] := true;
_YearOK[1, 23, 34] := true;
_YearOK[0, 24, 34] := true;
_YearOK[1, 24, 34] := true;
_YearOK[0, 25, 34] := true;
_YearOK[1, 25, 34] := true;
_YearOK[0, 26, 34] := true;
_YearOK[1, 26, 34] := true;
_YearOK[0, 27, 34] := true;
_YearOK[1, 27, 34] := true;
_YearOK[0, 28, 34] := true;
_YearOK[1, 28, 34] := true;
_YearOK[0, 29, 34] := true;
_YearOK[1, 29, 34] := true;
_YearOK[0, 30, 34] := true;
_YearOK[1, 30, 34] := true;
_YearOK[0, 31, 34] := true;
_YearOK[1, 31, 34] := true;
_YearOK[0, 32, 34] := true;
_YearOK[1, 32, 34] := true;
_YearOK[0, 33, 34] := true;
_YearOK[1, 33, 34] := true;
_YearOK[0, 34, 34] := true;
_YearOK[1, 34, 34] := true;
_YearOK[0, 35, 34] := true;
_YearOK[1, 35, 34] := true;
_YearOK[0, 36, 34] := true;
_YearOK[1, 36, 34] := true;
_YearOK[0, 37, 34] := true;
_YearOK[1, 37, 34] := true;
_YearOK[0, 38, 34] := true;
_YearOK[1, 38, 34] := true;
_YearOK[0, 39, 34] := true;
_YearOK[1, 39, 34] := true;
_YearOK[0, 40, 34] := true;
_YearOK[1, 40, 34] := true;
_YearOK[0, 41, 34] := true;
_YearOK[1, 41, 34] := true;
```

```
_YearOK[0, 42, 34] := true;
_YearOK[1, 42, 34] := true;
_YearOK[0, 43, 34] := true;
_YearOK[1, 43, 34] := true;
_YearOK[0, 44, 34] := true;
_YearOK[1, 44, 34] := true;
_YearOK[0, 45, 34] := true;
_YearOK[1, 45, 34] := true;
_YearOK[0, 46, 34] := true;
_YearOK[1, 46, 34] := true;
_YearOK[0, 47, 34] := true;
_YearOK[1, 47, 34] := true;
_YearOK[0, 48, 34] := true;
_YearOK[1, 48, 34] := true;
_YearOK[0, 49, 34] := true;
_YearOK[1, 49, 34] := true;
_YearOK[0, 50, 34] := true;
_YearOK[1, 50, 34] := true;
_YearOK[0, 51, 34] := true;
_YearOK[1, 51, 34] := true;
_YearOK[0, 52, 34] := true;
_YearOK[1, 52, 34] := true;
_YearOK[0, 53, 34] := true;
_YearOK[1, 53, 34] := true;
_YearOK[0, 54, 34] := true;
_YearOK[1, 54, 34] := true;
_YearOK[0, 55, 34] := true;
_YearOK[1, 55, 34] := true;
_YearOK[0, 56, 34] := true;
_YearOK[1, 56, 34] := true;
_YearOK[0, 57, 34] := true;
_YearOK[1, 57, 34] := true;
_YearOK[0, 58, 34] := true;
_YearOK[1, 58, 34] := true;
_YearOK[0, 59, 34] := true;
_YearOK[1, 59, 34] := true;
_YearOK[0, 60, 34] := true;
_YearOK[1, 60, 34] := true;
_YearOK[0, 61, 34] := true;
_YearOK[1, 61, 34] := true;
_YearOK[0, 62, 34] := true;
_YearOK[1, 62, 34] := true;
_YearOK[0, 63, 34] := true;
_YearOK[1, 63, 34] := true;
_YearOK[0, 64, 34] := true;
_YearOK[1, 64, 34] := true;
_YearOK[0, 65, 34] := true;
_YearOK[1, 65, 34] := true;
_YearOK[0, 66, 34] := true;
_YearOK[1, 66, 34] := true;
_YearOK[0, 67, 34] := true;
_YearOK[1, 67, 34] := true;
_YearOK[0, 68, 34] := true;
_YearOK[1, 68, 34] := true;
_YearOK[0, 69, 34] := true;
_YearOK[1, 69, 34] := true;
_YearOK[0, 70, 34] := true;
_YearOK[1, 70, 34] := true;
_YearOK[0, 71, 34] := true;
_YearOK[1, 71, 34] := true;
_YearOK[0, 72, 34] := true;
_YearOK[1, 72, 34] := true;
_YearOK[0, 73, 34] := true;
_YearOK[1, 73, 34] := true;
_YearOK[0, 74, 34] := true;
_YearOK[1, 74, 34] := true;
_YearOK[0, 75, 34] := true;
_YearOK[1, 75, 34] := true;
_YearOK[0, 76, 34] := true;
_YearOK[1, 76, 34] := true;
_YearOK[0, 77, 34] := true;
_YearOK[1, 77, 34] := true;
_YearOK[0, 78, 34] := true;
_YearOK[1, 78, 34] := true;
```

```
  _YearOK[0, 79, 34] := true;
  _YearOK[1, 79, 34] := true;
  _YearOK[0, 80, 34] := true;
  _YearOK[1, 80, 34] := true;
  _YearOK[0, 81, 34] := true;
  _YearOK[1, 81, 34] := true;
  _YearOK[0, 82, 34] := true;
  _YearOK[1, 82, 34] := true;
  _YearOK[0, 83, 34] := true;
  _YearOK[1, 83, 34] := true;
  _YearOK[0, 84, 34] := true;
  _YearOK[1, 84, 34] := true;
  _YearOK[0, 85, 34] := true;
  _YearOK[1, 85, 34] := true;
  _YearOK[0, 86, 34] := true;
  _YearOK[1, 86, 34] := true;
  _YearOK[0, 87, 34] := true;
  _YearOK[1, 87, 34] := true;
  _YearOK[0, 88, 34] := true;
  _YearOK[1, 88, 34] := true;
  _YearOK[0, 89, 34] := true;
  _YearOK[1, 89, 34] := true;
  _YearOK[0, 90, 34] := true;
  _YearOK[1, 90, 34] := true;
  _YearOK[0, 91, 34] := true;
  _YearOK[1, 91, 34] := true;
  _YearOK[0, 92, 34] := true;
  _YearOK[1, 92, 34] := true;
  _YearOK[0, 93, 34] := true;
  _YearOK[1, 93, 34] := true;
  _YearOK[0, 94, 34] := true;
  _YearOK[1, 94, 34] := true;
  _YearOK[0, 95, 34] := true;
  _YearOK[1, 95, 34] := true;
  _YearOK[0, 96, 34] := true;
  _YearOK[1, 96, 34] := true;
  _YearOK[0, 97, 34] := true;
  _YearOK[1, 97, 34] := true;
  _YearOK[0, 98, 34] := true;
  _YearOK[1, 98, 34] := true;
  _YearOK[0, 99, 34] := true;
  _YearOK[1, 99, 34] := true;
  _YearOK[0, 100, 34] := true;
  _YearOK[1, 100, 34] := true;
  _YearOK[0, 3, 35] := true;
  _YearOK[1, 3, 35] := true;
  _YearOK[0, 4, 35] := true;
  _YearOK[1, 4, 35] := true;
  _YearOK[0, 5, 35] := true;
  _YearOK[1, 5, 35] := true;
  _YearOK[0, 6, 35] := true;
  _YearOK[1, 6, 35] := true;
  _YearOK[0, 7, 35] := true;
  _YearOK[1, 7, 35] := true;
  _YearOK[0, 8, 35] := true;
  _YearOK[1, 8, 35] := true;
  _YearOK[0, 9, 35] := true;
  _YearOK[1, 9, 35] := true;
  _YearOK[0, 10, 35] := true;
  _YearOK[1, 10, 35] := true;
  _YearOK[0, 11, 35] := true;
  _YearOK[1, 11, 35] := true;
  _YearOK[0, 12, 35] := true;
  _YearOK[1, 12, 35] := true;
  _YearOK[0, 13, 35] := true;
  _YearOK[1, 13, 35] := true;
  _YearOK[0, 14, 35] := true;
  _YearOK[1, 14, 35] := true;
  _YearOK[0, 15, 35] := true;
  _YearOK[1, 15, 35] := true;
  _YearOK[0, 16, 35] := true;
  _YearOK[1, 16, 35] := true;
  _YearOK[0, 17, 35] := true;
  _YearOK[1, 17, 35] := true;
  _YearOK[0, 18, 35] := true;
```

```
  _YearOK[1, 18, 35] := true;
  _YearOK[0, 19, 35] := true;
  _YearOK[1, 19, 35] := true;
  _YearOK[0, 20, 35] := true;
  _YearOK[1, 20, 35] := true;
  _YearOK[0, 21, 35] := true;
  _YearOK[1, 21, 35] := true;
  _YearOK[0, 22, 35] := true;
  _YearOK[1, 22, 35] := true;
  _YearOK[0, 23, 35] := true;
  _YearOK[1, 23, 35] := true;
  _YearOK[0, 24, 35] := true;
  _YearOK[1, 24, 35] := true;
  _YearOK[0, 25, 35] := true;
  _YearOK[1, 25, 35] := true;
  _YearOK[0, 26, 35] := true;
  _YearOK[1, 26, 35] := true;
  _YearOK[0, 27, 35] := true;
  _YearOK[1, 27, 35] := true;
  _YearOK[0, 28, 35] := true;
  _YearOK[1, 28, 35] := true;
  _YearOK[0, 29, 35] := true;
  _YearOK[1, 29, 35] := true;
  _YearOK[0, 30, 35] := true;
  _YearOK[1, 30, 35] := true;
  _YearOK[0, 31, 35] := true;
  _YearOK[1, 31, 35] := true;
  _YearOK[0, 32, 35] := true;
  _YearOK[1, 32, 35] := true;
  _YearOK[0, 33, 35] := true;
  _YearOK[1, 33, 35] := true;
  _YearOK[0, 34, 35] := true;
  _YearOK[1, 34, 35] := true;
  _YearOK[0, 35, 35] := true;
  _YearOK[1, 35, 35] := true;
  _YearOK[0, 36, 35] := true;
  _YearOK[1, 36, 35] := true;
  _YearOK[0, 37, 35] := true;
  _YearOK[1, 37, 35] := true;
  _YearOK[0, 38, 35] := true;
  _YearOK[1, 38, 35] := true;
  _YearOK[0, 39, 35] := true;
  _YearOK[1, 39, 35] := true;
  _YearOK[0, 40, 35] := true;
  _YearOK[1, 40, 35] := true;
  _YearOK[0, 41, 35] := true;
  _YearOK[1, 41, 35] := true;
  _YearOK[0, 42, 35] := true;
  _YearOK[1, 42, 35] := true;
  _YearOK[0, 43, 35] := true;
  _YearOK[1, 43, 35] := true;
  _YearOK[0, 44, 35] := true;
  _YearOK[1, 44, 35] := true;
  _YearOK[0, 45, 35] := true;
  _YearOK[1, 45, 35] := true;
  _YearOK[0, 46, 35] := true;
  _YearOK[1, 46, 35] := true;
  _YearOK[0, 47, 35] := true;
  _YearOK[1, 47, 35] := true;
  _YearOK[0, 48, 35] := true;
  _YearOK[1, 48, 35] := true;
  _YearOK[0, 49, 35] := true;
  _YearOK[1, 49, 35] := true;
  _YearOK[0, 50, 35] := true;
  _YearOK[1, 50, 35] := true;
  _YearOK[0, 51, 35] := true;
  _YearOK[1, 51, 35] := true;
  _YearOK[0, 52, 35] := true;
  _YearOK[1, 52, 35] := true;
  _YearOK[0, 53, 35] := true;
  _YearOK[1, 53, 35] := true;
  _YearOK[0, 54, 35] := true;
  _YearOK[1, 54, 35] := true;
  _YearOK[0, 55, 35] := true;
```

```
  _YearOK[1, 55, 35] := true;
  _YearOK[0, 56, 35] := true;
  _YearOK[1, 56, 35] := true;
  _YearOK[0, 57, 35] := true;
  _YearOK[1, 57, 35] := true;
  _YearOK[0, 58, 35] := true;
  _YearOK[1, 58, 35] := true;
  _YearOK[0, 59, 35] := true;
  _YearOK[1, 59, 35] := true;
  _YearOK[0, 60, 35] := true;
  _YearOK[1, 60, 35] := true;
  _YearOK[0, 61, 35] := true;
  _YearOK[1, 61, 35] := true;
  _YearOK[0, 62, 35] := true;
  _YearOK[1, 62, 35] := true;
  _YearOK[0, 63, 35] := true;
  _YearOK[1, 63, 35] := true;
  _YearOK[0, 64, 35] := true;
  _YearOK[1, 64, 35] := true;
  _YearOK[0, 65, 35] := true;
  _YearOK[1, 65, 35] := true;
  _YearOK[0, 66, 35] := true;
  _YearOK[1, 66, 35] := true;
  _YearOK[0, 67, 35] := true;
  _YearOK[1, 67, 35] := true;
  _YearOK[0, 68, 35] := true;
  _YearOK[1, 68, 35] := true;
  _YearOK[0, 69, 35] := true;
  _YearOK[1, 69, 35] := true;
  _YearOK[0, 70, 35] := true;
  _YearOK[1, 70, 35] := true;
  _YearOK[0, 71, 35] := true;
  _YearOK[1, 71, 35] := true;
  _YearOK[0, 72, 35] := true;
  _YearOK[1, 72, 35] := true;
  _YearOK[0, 73, 35] := true;
  _YearOK[1, 73, 35] := true;
  _YearOK[0, 74, 35] := true;
  _YearOK[1, 74, 35] := true;
  _YearOK[0, 75, 35] := true;
  _YearOK[1, 75, 35] := true;
  _YearOK[0, 76, 35] := true;
  _YearOK[1, 76, 35] := true;
  _YearOK[0, 77, 35] := true;
  _YearOK[1, 77, 35] := true;
  _YearOK[0, 78, 35] := true;
  _YearOK[1, 78, 35] := true;
  _YearOK[0, 79, 35] := true;
  _YearOK[1, 79, 35] := true;
  _YearOK[0, 80, 35] := true;
  _YearOK[1, 80, 35] := true;
  _YearOK[0, 81, 35] := true;
  _YearOK[1, 81, 35] := true;
  _YearOK[0, 82, 35] := true;
  _YearOK[1, 82, 35] := true;
  _YearOK[0, 83, 35] := true;
  _YearOK[1, 83, 35] := true;
  _YearOK[0, 84, 35] := true;
  _YearOK[1, 84, 35] := true;
  _YearOK[0, 85, 35] := true;
  _YearOK[1, 85, 35] := true;
  _YearOK[0, 86, 35] := true;
  _YearOK[1, 86, 35] := true;
  _YearOK[0, 87, 35] := true;
  _YearOK[1, 87, 35] := true;
  _YearOK[0, 88, 35] := true;
  _YearOK[1, 88, 35] := true;
  _YearOK[0, 89, 35] := true;
  _YearOK[1, 89, 35] := true;
  _YearOK[0, 90, 35] := true;
  _YearOK[1, 90, 35] := true;
  _YearOK[0, 91, 35] := true;
  _YearOK[1, 91, 35] := true;
  _YearOK[0, 92, 35] := true;
  _YearOK[1, 92, 35] := true;
```

```
_YearOK[0, 93, 35] := true;
_YearOK[1, 93, 35] := true;
_YearOK[0, 94, 35] := true;
_YearOK[1, 94, 35] := true;
_YearOK[0, 95, 35] := true;
_YearOK[1, 95, 35] := true;
_YearOK[0, 96, 35] := true;
_YearOK[1, 96, 35] := true;
_YearOK[0, 97, 35] := true;
_YearOK[1, 97, 35] := true;
_YearOK[0, 98, 35] := true;
_YearOK[1, 98, 35] := true;
_YearOK[0, 99, 35] := true;
_YearOK[1, 99, 35] := true;
_YearOK[0, 100, 35] := true;
_YearOK[1, 100, 35] := true;
_YearOK[0, 3, 36] := true;
_YearOK[1, 3, 36] := true;
_YearOK[0, 12, 36] := true;
_YearOK[1, 12, 36] := true;
_YearOK[0, 21, 36] := true;
_YearOK[1, 21, 36] := true;
_YearOK[0, 30, 36] := true;
_YearOK[1, 30, 36] := true;
_YearOK[0, 39, 36] := true;
_YearOK[1, 39, 36] := true;
_YearOK[0, 48, 36] := true;
_YearOK[1, 48, 36] := true;
_YearOK[0, 57, 36] := true;
_YearOK[1, 57, 36] := true;
_YearOK[0, 66, 36] := true;
_YearOK[1, 66, 36] := true;
_YearOK[0, 75, 36] := true;
_YearOK[1, 75, 36] := true;
_YearOK[0, 84, 36] := true;
_YearOK[1, 84, 36] := true;
_YearOK[0, 93, 36] := true;
_YearOK[1, 93, 36] := true;
_YearOK[0, 3, 37] := true;
_YearOK[1, 3, 37] := true;
_YearOK[0, 12, 37] := true;
_YearOK[1, 12, 37] := true;
_YearOK[0, 21, 37] := true;
_YearOK[1, 21, 37] := true;
_YearOK[0, 30, 37] := true;
_YearOK[1, 30, 37] := true;
_YearOK[0, 39, 37] := true;
_YearOK[1, 39, 37] := true;
_YearOK[0, 48, 37] := true;
_YearOK[1, 48, 37] := true;
_YearOK[0, 57, 37] := true;
_YearOK[1, 57, 37] := true;
_YearOK[0, 66, 37] := true;
_YearOK[1, 66, 37] := true;
_YearOK[0, 75, 37] := true;
_YearOK[1, 75, 37] := true;
_YearOK[0, 84, 37] := true;
_YearOK[1, 84, 37] := true;
_YearOK[0, 93, 37] := true;
_YearOK[1, 93, 37] := true;
_YearOK[0, 3, 38] := true;
_YearOK[1, 3, 38] := true;
_YearOK[0, 12, 38] := true;
_YearOK[1, 12, 38] := true;
_YearOK[0, 21, 38] := true;
_YearOK[1, 21, 38] := true;
_YearOK[0, 30, 38] := true;
_YearOK[1, 30, 38] := true;
_YearOK[0, 39, 38] := true;
_YearOK[1, 39, 38] := true;
_YearOK[0, 48, 38] := true;
_YearOK[1, 48, 38] := true;
_YearOK[0, 57, 38] := true;
_YearOK[1, 57, 38] := true;
```

```
_YearOK[0, 66, 38] := true;
_YearOK[1, 66, 38] := true;
_YearOK[0, 75, 38] := true;
_YearOK[1, 75, 38] := true;
_YearOK[0, 84, 38] := true;
_YearOK[1, 84, 38] := true;
_YearOK[0, 93, 38] := true;
_YearOK[1, 93, 38] := true;
_YearOK[0, 3, 39] := true;
_YearOK[1, 3, 39] := true;
_YearOK[0, 12, 39] := true;
_YearOK[1, 12, 39] := true;
_YearOK[0, 21, 39] := true;
_YearOK[1, 21, 39] := true;
_YearOK[0, 30, 39] := true;
_YearOK[1, 30, 39] := true;
_YearOK[0, 39, 39] := true;
_YearOK[1, 39, 39] := true;
_YearOK[0, 48, 39] := true;
_YearOK[1, 48, 39] := true;
_YearOK[0, 57, 39] := true;
_YearOK[1, 57, 39] := true;
_YearOK[0, 66, 39] := true;
_YearOK[1, 66, 39] := true;
_YearOK[0, 75, 39] := true;
_YearOK[1, 75, 39] := true;
_YearOK[0, 84, 39] := true;
_YearOK[1, 84, 39] := true;
_YearOK[0, 93, 39] := true;
_YearOK[1, 93, 39] := true;
_YearOK[0, 3, 40] := true;
_YearOK[1, 3, 40] := true;
_YearOK[0, 12, 40] := true;
_YearOK[1, 12, 40] := true;
_YearOK[0, 21, 40] := true;
_YearOK[1, 21, 40] := true;
_YearOK[0, 30, 40] := true;
_YearOK[1, 30, 40] := true;
_YearOK[0, 39, 40] := true;
_YearOK[1, 39, 40] := true;
_YearOK[0, 48, 40] := true;
_YearOK[1, 48, 40] := true;
_YearOK[0, 57, 40] := true;
_YearOK[1, 57, 40] := true;
_YearOK[0, 66, 40] := true;
_YearOK[1, 66, 40] := true;
_YearOK[0, 75, 40] := true;
_YearOK[1, 75, 40] := true;
_YearOK[0, 84, 40] := true;
_YearOK[1, 84, 40] := true;
_YearOK[0, 93, 40] := true;
_YearOK[1, 93, 40] := true;
_YearOK[0, 3, 41] := true;
_YearOK[1, 3, 41] := true;
_YearOK[0, 12, 41] := true;
_YearOK[1, 12, 41] := true;
_YearOK[0, 21, 41] := true;
_YearOK[1, 21, 41] := true;
_YearOK[0, 30, 41] := true;
_YearOK[1, 30, 41] := true;
_YearOK[0, 39, 41] := true;
_YearOK[1, 39, 41] := true;
_YearOK[0, 48, 41] := true;
_YearOK[1, 48, 41] := true;
_YearOK[0, 57, 41] := true;
_YearOK[1, 57, 41] := true;
_YearOK[0, 66, 41] := true;
_YearOK[1, 66, 41] := true;
_YearOK[0, 75, 41] := true;
_YearOK[1, 75, 41] := true;
_YearOK[0, 84, 41] := true;
_YearOK[1, 84, 41] := true;
_YearOK[0, 93, 41] := true;
_YearOK[1, 93, 41] := true;
_YearOK[0, 4, 42] := true;
```

```
_YearOK[1, 4, 42] := true;
_YearOK[0, 4, 43] := true;
_YearOK[1, 4, 43] := true;
_YearOK[0, 4, 44] := true;
_YearOK[1, 4, 44] := true;
_YearOK[0, 4, 45] := true;
_YearOK[1, 4, 45] := true;
_YearOK[0, 3, 46] := true;
_YearOK[1, 3, 46] := true;
_YearOK[0, 3, 47] := true;
_YearOK[1, 3, 47] := true;
_YearOK[0, 3, 48] := true;
_YearOK[1, 3, 48] := true;
_YearOK[0, 3, 49] := true;
_YearOK[1, 3, 49] := true;
_YearOK[0, 4, 49] := true;
_YearOK[1, 4, 49] := true;
_YearOK[0, 5, 49] := true;
_YearOK[1, 5, 49] := true;
_YearOK[0, 6, 49] := true;
_YearOK[1, 6, 49] := true;
_YearOK[0, 7, 49] := true;
_YearOK[1, 7, 49] := true;
_YearOK[0, 8, 49] := true;
_YearOK[1, 8, 49] := true;
_YearOK[0, 9, 49] := true;
_YearOK[1, 9, 49] := true;
_YearOK[0, 10, 49] := true;
_YearOK[1, 10, 49] := true;
_YearOK[0, 11, 49] := true;
_YearOK[1, 11, 49] := true;
_YearOK[0, 12, 49] := true;
_YearOK[1, 12, 49] := true;
_YearOK[0, 13, 49] := true;
_YearOK[1, 13, 49] := true;
_YearOK[0, 14, 49] := true;
_YearOK[1, 14, 49] := true;
_YearOK[0, 15, 49] := true;
_YearOK[1, 15, 49] := true;
_YearOK[0, 16, 49] := true;
_YearOK[1, 16, 49] := true;
_YearOK[0, 17, 49] := true;
_YearOK[1, 17, 49] := true;
_YearOK[0, 18, 49] := true;
_YearOK[1, 18, 49] := true;
_YearOK[0, 19, 49] := true;
_YearOK[1, 19, 49] := true;
_YearOK[0, 20, 49] := true;
_YearOK[1, 20, 49] := true;
_YearOK[0, 21, 49] := true;
_YearOK[1, 21, 49] := true;
_YearOK[0, 22, 49] := true;
_YearOK[1, 22, 49] := true;
_YearOK[0, 23, 49] := true;
_YearOK[1, 23, 49] := true;
_YearOK[0, 24, 49] := true;
_YearOK[1, 24, 49] := true;
_YearOK[0, 25, 49] := true;
_YearOK[1, 25, 49] := true;
_YearOK[0, 26, 49] := true;
_YearOK[1, 26, 49] := true;
_YearOK[0, 27, 49] := true;
_YearOK[1, 27, 49] := true;
_YearOK[0, 28, 49] := true;
_YearOK[1, 28, 49] := true;
_YearOK[0, 29, 49] := true;
_YearOK[1, 29, 49] := true;
_YearOK[0, 30, 49] := true;
_YearOK[1, 30, 49] := true;
_YearOK[0, 31, 49] := true;
_YearOK[1, 31, 49] := true;
_YearOK[0, 32, 49] := true;
_YearOK[1, 32, 49] := true;
_YearOK[0, 33, 49] := true;
```

```
  _YearOK[1, 33, 49] := true;
  _YearOK[0, 34, 49] := true;
  _YearOK[1, 34, 49] := true;
  _YearOK[0, 35, 49] := true;
  _YearOK[1, 35, 49] := true;
  _YearOK[0, 36, 49] := true;
  _YearOK[1, 36, 49] := true;
  _YearOK[0, 37, 49] := true;
  _YearOK[1, 37, 49] := true;
  _YearOK[0, 38, 49] := true;
  _YearOK[1, 38, 49] := true;
  _YearOK[0, 39, 49] := true;
  _YearOK[1, 39, 49] := true;
  _YearOK[0, 40, 49] := true;
  _YearOK[1, 40, 49] := true;
  _YearOK[0, 41, 49] := true;
  _YearOK[1, 41, 49] := true;
  _YearOK[0, 42, 49] := true;
  _YearOK[1, 42, 49] := true;
  _YearOK[0, 43, 49] := true;
  _YearOK[1, 43, 49] := true;
  _YearOK[0, 44, 49] := true;
  _YearOK[1, 44, 49] := true;
  _YearOK[0, 45, 49] := true;
  _YearOK[1, 45, 49] := true;
  _YearOK[0, 46, 49] := true;
  _YearOK[1, 46, 49] := true;
  _YearOK[0, 47, 49] := true;
  _YearOK[1, 47, 49] := true;
  _YearOK[0, 48, 49] := true;
  _YearOK[1, 48, 49] := true;
  _YearOK[0, 49, 49] := true;
  _YearOK[1, 49, 49] := true;
  _YearOK[0, 50, 49] := true;
  _YearOK[1, 50, 49] := true;
  _YearOK[0, 51, 49] := true;
  _YearOK[1, 51, 49] := true;
  _YearOK[0, 52, 49] := true;
  _YearOK[1, 52, 49] := true;
  _YearOK[0, 53, 49] := true;
  _YearOK[1, 53, 49] := true;
  _YearOK[0, 54, 49] := true;
  _YearOK[1, 54, 49] := true;
  _YearOK[0, 55, 49] := true;
  _YearOK[1, 55, 49] := true;
  _YearOK[0, 56, 49] := true;
  _YearOK[1, 56, 49] := true;
  _YearOK[0, 57, 49] := true;
  _YearOK[1, 57, 49] := true;
  _YearOK[0, 58, 49] := true;
  _YearOK[1, 58, 49] := true;
  _YearOK[0, 59, 49] := true;
  _YearOK[1, 59, 49] := true;
  _YearOK[0, 60, 49] := true;
  _YearOK[1, 60, 49] := true;
  _YearOK[0, 61, 49] := true;
  _YearOK[1, 61, 49] := true;
  _YearOK[0, 62, 49] := true;
  _YearOK[1, 62, 49] := true;
  _YearOK[0, 63, 49] := true;
  _YearOK[1, 63, 49] := true;
  _YearOK[0, 64, 49] := true;
  _YearOK[1, 64, 49] := true;
  _YearOK[0, 65, 49] := true;
  _YearOK[1, 65, 49] := true;
  _YearOK[0, 66, 49] := true;
  _YearOK[1, 66, 49] := true;
  _YearOK[0, 67, 49] := true;
  _YearOK[1, 67, 49] := true;
  _YearOK[0, 68, 49] := true;
  _YearOK[1, 68, 49] := true;
  _YearOK[0, 69, 49] := true;
  _YearOK[1, 69, 49] := true;
  _YearOK[0, 70, 49] := true;
  _YearOK[1, 70, 49] := true;
```

```
  _YearOK[0, 71, 49] := true;
  _YearOK[1, 71, 49] := true;
  _YearOK[0, 72, 49] := true;
  _YearOK[1, 72, 49] := true;
  _YearOK[0, 73, 49] := true;
  _YearOK[1, 73, 49] := true;
  _YearOK[0, 74, 49] := true;
  _YearOK[1, 74, 49] := true;
  _YearOK[0, 75, 49] := true;
  _YearOK[1, 75, 49] := true;
  _YearOK[0, 76, 49] := true;
  _YearOK[1, 76, 49] := true;
  _YearOK[0, 77, 49] := true;
  _YearOK[1, 77, 49] := true;
  _YearOK[0, 78, 49] := true;
  _YearOK[1, 78, 49] := true;
  _YearOK[0, 79, 49] := true;
  _YearOK[1, 79, 49] := true;
  _YearOK[0, 80, 49] := true;
  _YearOK[1, 80, 49] := true;
  _YearOK[0, 81, 49] := true;
  _YearOK[1, 81, 49] := true;
  _YearOK[0, 82, 49] := true;
  _YearOK[1, 82, 49] := true;
  _YearOK[0, 83, 49] := true;
  _YearOK[1, 83, 49] := true;
  _YearOK[0, 84, 49] := true;
  _YearOK[1, 84, 49] := true;
  _YearOK[0, 85, 49] := true;
  _YearOK[1, 85, 49] := true;
  _YearOK[0, 86, 49] := true;
  _YearOK[1, 86, 49] := true;
  _YearOK[0, 87, 49] := true;
  _YearOK[1, 87, 49] := true;
  _YearOK[0, 88, 49] := true;
  _YearOK[1, 88, 49] := true;
  _YearOK[0, 89, 49] := true;
  _YearOK[1, 89, 49] := true;
  _YearOK[0, 90, 49] := true;
  _YearOK[1, 90, 49] := true;
  _YearOK[0, 91, 49] := true;
  _YearOK[1, 91, 49] := true;
  _YearOK[0, 92, 49] := true;
  _YearOK[1, 92, 49] := true;
  _YearOK[0, 93, 49] := true;
  _YearOK[1, 93, 49] := true;
  _YearOK[0, 94, 49] := true;
  _YearOK[1, 94, 49] := true;
  _YearOK[0, 95, 49] := true;
  _YearOK[1, 95, 49] := true;
  _YearOK[0, 96, 49] := true;
  _YearOK[1, 96, 49] := true;
  _YearOK[0, 97, 49] := true;
  _YearOK[1, 97, 49] := true;
  _YearOK[0, 98, 49] := true;
  _YearOK[1, 98, 49] := true;
  _YearOK[0, 99, 49] := true;
  _YearOK[1, 99, 49] := true;
  _YearOK[0, 100, 49] := true;
  _YearOK[1, 100, 49] := true;
  _YearOK[0, 3, 50] := true;
  _YearOK[1, 3, 50] := true;
  _YearOK[0, 4, 50] := true;
  _YearOK[1, 4, 50] := true;
  _YearOK[0, 5, 50] := true;
  _YearOK[1, 5, 50] := true;
  _YearOK[0, 6, 50] := true;
  _YearOK[1, 6, 50] := true;
  _YearOK[0, 7, 50] := true;
  _YearOK[1, 7, 50] := true;
  _YearOK[0, 8, 50] := true;
  _YearOK[1, 8, 50] := true;
  _YearOK[0, 9, 50] := true;
  _YearOK[1, 9, 50] := true;
```

```
  _YearOK[0, 10, 50] := true;
  _YearOK[1, 10, 50] := true;
  _YearOK[0, 11, 50] := true;
  _YearOK[1, 11, 50] := true;
  _YearOK[0, 12, 50] := true;
  _YearOK[1, 12, 50] := true;
  _YearOK[0, 13, 50] := true;
  _YearOK[1, 13, 50] := true;
  _YearOK[0, 14, 50] := true;
  _YearOK[1, 14, 50] := true;
  _YearOK[0, 15, 50] := true;
  _YearOK[1, 15, 50] := true;
  _YearOK[0, 16, 50] := true;
  _YearOK[1, 16, 50] := true;
  _YearOK[0, 17, 50] := true;
  _YearOK[1, 17, 50] := true;
  _YearOK[0, 18, 50] := true;
  _YearOK[1, 18, 50] := true;
  _YearOK[0, 19, 50] := true;
  _YearOK[1, 19, 50] := true;
  _YearOK[0, 20, 50] := true;
  _YearOK[1, 20, 50] := true;
  _YearOK[0, 21, 50] := true;
  _YearOK[1, 21, 50] := true;
  _YearOK[0, 22, 50] := true;
  _YearOK[1, 22, 50] := true;
  _YearOK[0, 23, 50] := true;
  _YearOK[1, 23, 50] := true;
  _YearOK[0, 24, 50] := true;
  _YearOK[1, 24, 50] := true;
  _YearOK[0, 25, 50] := true;
  _YearOK[1, 25, 50] := true;
  _YearOK[0, 26, 50] := true;
  _YearOK[1, 26, 50] := true;
  _YearOK[0, 27, 50] := true;
  _YearOK[1, 27, 50] := true;
  _YearOK[0, 28, 50] := true;
  _YearOK[1, 28, 50] := true;
  _YearOK[0, 29, 50] := true;
  _YearOK[1, 29, 50] := true;
  _YearOK[0, 30, 50] := true;
  _YearOK[1, 30, 50] := true;
  _YearOK[0, 31, 50] := true;
  _YearOK[1, 31, 50] := true;
  _YearOK[0, 32, 50] := true;
  _YearOK[1, 32, 50] := true;
  _YearOK[0, 33, 50] := true;
  _YearOK[1, 33, 50] := true;
  _YearOK[0, 34, 50] := true;
  _YearOK[1, 34, 50] := true;
  _YearOK[0, 35, 50] := true;
  _YearOK[1, 35, 50] := true;
  _YearOK[0, 36, 50] := true;
  _YearOK[1, 36, 50] := true;
  _YearOK[0, 37, 50] := true;
  _YearOK[1, 37, 50] := true;
  _YearOK[0, 38, 50] := true;
  _YearOK[1, 38, 50] := true;
  _YearOK[0, 39, 50] := true;
  _YearOK[1, 39, 50] := true;
  _YearOK[0, 40, 50] := true;
  _YearOK[1, 40, 50] := true;
  _YearOK[0, 41, 50] := true;
  _YearOK[1, 41, 50] := true;
  _YearOK[0, 42, 50] := true;
  _YearOK[1, 42, 50] := true;
  _YearOK[0, 43, 50] := true;
  _YearOK[1, 43, 50] := true;
  _YearOK[0, 44, 50] := true;
  _YearOK[1, 44, 50] := true;
  _YearOK[0, 45, 50] := true;
  _YearOK[1, 45, 50] := true;
  _YearOK[0, 46, 50] := true;
  _YearOK[1, 46, 50] := true;
  _YearOK[0, 47, 50] := true;
```

```
_YearOK[1, 47, 50] := true;
_YearOK[0, 48, 50] := true;
_YearOK[1, 48, 50] := true;
_YearOK[0, 49, 50] := true;
_YearOK[1, 49, 50] := true;
_YearOK[0, 50, 50] := true;
_YearOK[1, 50, 50] := true;
_YearOK[0, 51, 50] := true;
_YearOK[1, 51, 50] := true;
_YearOK[0, 52, 50] := true;
_YearOK[1, 52, 50] := true;
_YearOK[0, 53, 50] := true;
_YearOK[1, 53, 50] := true;
_YearOK[0, 54, 50] := true;
_YearOK[1, 54, 50] := true;
_YearOK[0, 55, 50] := true;
_YearOK[1, 55, 50] := true;
_YearOK[0, 56, 50] := true;
_YearOK[1, 56, 50] := true;
_YearOK[0, 57, 50] := true;
_YearOK[1, 57, 50] := true;
_YearOK[0, 58, 50] := true;
_YearOK[1, 58, 50] := true;
_YearOK[0, 59, 50] := true;
_YearOK[1, 59, 50] := true;
_YearOK[0, 60, 50] := true;
_YearOK[1, 60, 50] := true;
_YearOK[0, 61, 50] := true;
_YearOK[1, 61, 50] := true;
_YearOK[0, 62, 50] := true;
_YearOK[1, 62, 50] := true;
_YearOK[0, 63, 50] := true;
_YearOK[1, 63, 50] := true;
_YearOK[0, 64, 50] := true;
_YearOK[1, 64, 50] := true;
_YearOK[0, 65, 50] := true;
_YearOK[1, 65, 50] := true;
_YearOK[0, 66, 50] := true;
_YearOK[1, 66, 50] := true;
_YearOK[0, 67, 50] := true;
_YearOK[1, 67, 50] := true;
_YearOK[0, 68, 50] := true;
_YearOK[1, 68, 50] := true;
_YearOK[0, 69, 50] := true;
_YearOK[1, 69, 50] := true;
_YearOK[0, 70, 50] := true;
_YearOK[1, 70, 50] := true;
_YearOK[0, 71, 50] := true;
_YearOK[1, 71, 50] := true;
_YearOK[0, 72, 50] := true;
_YearOK[1, 72, 50] := true;
_YearOK[0, 73, 50] := true;
_YearOK[1, 73, 50] := true;
_YearOK[0, 74, 50] := true;
_YearOK[1, 74, 50] := true;
_YearOK[0, 75, 50] := true;
_YearOK[1, 75, 50] := true;
_YearOK[0, 76, 50] := true;
_YearOK[1, 76, 50] := true;
_YearOK[0, 77, 50] := true;
_YearOK[1, 77, 50] := true;
_YearOK[0, 78, 50] := true;
_YearOK[1, 78, 50] := true;
_YearOK[0, 79, 50] := true;
_YearOK[1, 79, 50] := true;
_YearOK[0, 80, 50] := true;
_YearOK[1, 80, 50] := true;
_YearOK[0, 81, 50] := true;
_YearOK[1, 81, 50] := true;
_YearOK[0, 82, 50] := true;
_YearOK[1, 82, 50] := true;
_YearOK[0, 83, 50] := true;
_YearOK[1, 83, 50] := true;
_YearOK[0, 84, 50] := true;
```

```
_YearOK[1, 84, 50] := true;
_YearOK[0, 85, 50] := true;
_YearOK[1, 85, 50] := true;
_YearOK[0, 86, 50] := true;
_YearOK[1, 86, 50] := true;
_YearOK[0, 87, 50] := true;
_YearOK[1, 87, 50] := true;
_YearOK[0, 88, 50] := true;
_YearOK[1, 88, 50] := true;
_YearOK[0, 89, 50] := true;
_YearOK[1, 89, 50] := true;
_YearOK[0, 90, 50] := true;
_YearOK[1, 90, 50] := true;
_YearOK[0, 91, 50] := true;
_YearOK[1, 91, 50] := true;
_YearOK[0, 92, 50] := true;
_YearOK[1, 92, 50] := true;
_YearOK[0, 93, 50] := true;
_YearOK[1, 93, 50] := true;
_YearOK[0, 94, 50] := true;
_YearOK[1, 94, 50] := true;
_YearOK[0, 95, 50] := true;
_YearOK[1, 95, 50] := true;
_YearOK[0, 96, 50] := true;
_YearOK[1, 96, 50] := true;
_YearOK[0, 97, 50] := true;
_YearOK[1, 97, 50] := true;
_YearOK[0, 98, 50] := true;
_YearOK[1, 98, 50] := true;
_YearOK[0, 99, 50] := true;
_YearOK[1, 99, 50] := true;
_YearOK[0, 100, 50] := true;
_YearOK[1, 100, 50] := true;
_YearOK[0, 3, 51] := true;
_YearOK[1, 3, 51] := true;
_YearOK[0, 4, 51] := true;
_YearOK[1, 4, 51] := true;
_YearOK[0, 5, 51] := true;
_YearOK[1, 5, 51] := true;
_YearOK[0, 6, 51] := true;
_YearOK[1, 6, 51] := true;
_YearOK[0, 7, 51] := true;
_YearOK[1, 7, 51] := true;
_YearOK[0, 8, 51] := true;
_YearOK[1, 8, 51] := true;
_YearOK[0, 9, 51] := true;
_YearOK[1, 9, 51] := true;
_YearOK[0, 10, 51] := true;
_YearOK[1, 10, 51] := true;
_YearOK[0, 11, 51] := true;
_YearOK[1, 11, 51] := true;
_YearOK[0, 12, 51] := true;
_YearOK[1, 12, 51] := true;
_YearOK[0, 13, 51] := true;
_YearOK[1, 13, 51] := true;
_YearOK[0, 14, 51] := true;
_YearOK[1, 14, 51] := true;
_YearOK[0, 15, 51] := true;
_YearOK[1, 15, 51] := true;
_YearOK[0, 16, 51] := true;
_YearOK[1, 16, 51] := true;
_YearOK[0, 17, 51] := true;
_YearOK[1, 17, 51] := true;
_YearOK[0, 18, 51] := true;
_YearOK[1, 18, 51] := true;
_YearOK[0, 19, 51] := true;
_YearOK[1, 19, 51] := true;
_YearOK[0, 20, 51] := true;
_YearOK[1, 20, 51] := true;
_YearOK[0, 21, 51] := true;
_YearOK[1, 21, 51] := true;
_YearOK[0, 22, 51] := true;
_YearOK[1, 22, 51] := true;
_YearOK[0, 23, 51] := true;
_YearOK[1, 23, 51] := true;
```

```
_YearOK[0, 24, 51] := true;
_YearOK[1, 24, 51] := true;
_YearOK[0, 25, 51] := true;
_YearOK[1, 25, 51] := true;
_YearOK[0, 26, 51] := true;
_YearOK[1, 26, 51] := true;
_YearOK[0, 27, 51] := true;
_YearOK[1, 27, 51] := true;
_YearOK[0, 28, 51] := true;
_YearOK[1, 28, 51] := true;
_YearOK[0, 29, 51] := true;
_YearOK[1, 29, 51] := true;
_YearOK[0, 30, 51] := true;
_YearOK[1, 30, 51] := true;
_YearOK[0, 31, 51] := true;
_YearOK[1, 31, 51] := true;
_YearOK[0, 32, 51] := true;
_YearOK[1, 32, 51] := true;
_YearOK[0, 33, 51] := true;
_YearOK[1, 33, 51] := true;
_YearOK[0, 34, 51] := true;
_YearOK[1, 34, 51] := true;
_YearOK[0, 35, 51] := true;
_YearOK[1, 35, 51] := true;
_YearOK[0, 36, 51] := true;
_YearOK[1, 36, 51] := true;
_YearOK[0, 37, 51] := true;
_YearOK[1, 37, 51] := true;
_YearOK[0, 38, 51] := true;
_YearOK[1, 38, 51] := true;
_YearOK[0, 39, 51] := true;
_YearOK[1, 39, 51] := true;
_YearOK[0, 40, 51] := true;
_YearOK[1, 40, 51] := true;
_YearOK[0, 41, 51] := true;
_YearOK[1, 41, 51] := true;
_YearOK[0, 42, 51] := true;
_YearOK[1, 42, 51] := true;
_YearOK[0, 43, 51] := true;
_YearOK[1, 43, 51] := true;
_YearOK[0, 44, 51] := true;
_YearOK[1, 44, 51] := true;
_YearOK[0, 45, 51] := true;
_YearOK[1, 45, 51] := true;
_YearOK[0, 46, 51] := true;
_YearOK[1, 46, 51] := true;
_YearOK[0, 47, 51] := true;
_YearOK[1, 47, 51] := true;
_YearOK[0, 48, 51] := true;
_YearOK[1, 48, 51] := true;
_YearOK[0, 49, 51] := true;
_YearOK[1, 49, 51] := true;
_YearOK[0, 50, 51] := true;
_YearOK[1, 50, 51] := true;
_YearOK[0, 51, 51] := true;
_YearOK[1, 51, 51] := true;
_YearOK[0, 52, 51] := true;
_YearOK[1, 52, 51] := true;
_YearOK[0, 53, 51] := true;
_YearOK[1, 53, 51] := true;
_YearOK[0, 54, 51] := true;
_YearOK[1, 54, 51] := true;
_YearOK[0, 55, 51] := true;
_YearOK[1, 55, 51] := true;
_YearOK[0, 56, 51] := true;
_YearOK[1, 56, 51] := true;
_YearOK[0, 57, 51] := true;
_YearOK[1, 57, 51] := true;
_YearOK[0, 58, 51] := true;
_YearOK[1, 58, 51] := true;
_YearOK[0, 59, 51] := true;
_YearOK[1, 59, 51] := true;
_YearOK[0, 60, 51] := true;
_YearOK[1, 60, 51] := true;
```

```
  _YearOK[0, 61, 51] := true;
  _YearOK[1, 61, 51] := true;
  _YearOK[0, 62, 51] := true;
  _YearOK[1, 62, 51] := true;
  _YearOK[0, 63, 51] := true;
  _YearOK[1, 63, 51] := true;
  _YearOK[0, 64, 51] := true;
  _YearOK[1, 64, 51] := true;
  _YearOK[0, 65, 51] := true;
  _YearOK[1, 65, 51] := true;
  _YearOK[0, 66, 51] := true;
  _YearOK[1, 66, 51] := true;
  _YearOK[0, 67, 51] := true;
  _YearOK[1, 67, 51] := true;
  _YearOK[0, 68, 51] := true;
  _YearOK[1, 68, 51] := true;
  _YearOK[0, 69, 51] := true;
  _YearOK[1, 69, 51] := true;
  _YearOK[0, 70, 51] := true;
  _YearOK[1, 70, 51] := true;
  _YearOK[0, 71, 51] := true;
  _YearOK[1, 71, 51] := true;
  _YearOK[0, 72, 51] := true;
  _YearOK[1, 72, 51] := true;
  _YearOK[0, 73, 51] := true;
  _YearOK[1, 73, 51] := true;
  _YearOK[0, 74, 51] := true;
  _YearOK[1, 74, 51] := true;
  _YearOK[0, 75, 51] := true;
  _YearOK[1, 75, 51] := true;
  _YearOK[0, 76, 51] := true;
  _YearOK[1, 76, 51] := true;
  _YearOK[0, 77, 51] := true;
  _YearOK[1, 77, 51] := true;
  _YearOK[0, 78, 51] := true;
  _YearOK[1, 78, 51] := true;
  _YearOK[0, 79, 51] := true;
  _YearOK[1, 79, 51] := true;
  _YearOK[0, 80, 51] := true;
  _YearOK[1, 80, 51] := true;
  _YearOK[0, 81, 51] := true;
  _YearOK[1, 81, 51] := true;
  _YearOK[0, 82, 51] := true;
  _YearOK[1, 82, 51] := true;
  _YearOK[0, 83, 51] := true;
  _YearOK[1, 83, 51] := true;
  _YearOK[0, 84, 51] := true;
  _YearOK[1, 84, 51] := true;
  _YearOK[0, 85, 51] := true;
  _YearOK[1, 85, 51] := true;
  _YearOK[0, 86, 51] := true;
  _YearOK[1, 86, 51] := true;
  _YearOK[0, 87, 51] := true;
  _YearOK[1, 87, 51] := true;
  _YearOK[0, 88, 51] := true;
  _YearOK[1, 88, 51] := true;
  _YearOK[0, 89, 51] := true;
  _YearOK[1, 89, 51] := true;
  _YearOK[0, 90, 51] := true;
  _YearOK[1, 90, 51] := true;
  _YearOK[0, 91, 51] := true;
  _YearOK[1, 91, 51] := true;
  _YearOK[0, 92, 51] := true;
  _YearOK[1, 92, 51] := true;
  _YearOK[0, 93, 51] := true;
  _YearOK[1, 93, 51] := true;
  _YearOK[0, 94, 51] := true;
  _YearOK[1, 94, 51] := true;
  _YearOK[0, 95, 51] := true;
  _YearOK[1, 95, 51] := true;
  _YearOK[0, 96, 51] := true;
  _YearOK[1, 96, 51] := true;
  _YearOK[0, 97, 51] := true;
  _YearOK[1, 97, 51] := true;
  _YearOK[0, 98, 51] := true;
```

```
_YearOK[1, 98, 51] := true;
_YearOK[0, 99, 51] := true;
_YearOK[1, 99, 51] := true;
_YearOK[0, 100, 51] := true;
_YearOK[1, 100, 51] := true;
_YearOK[0, 3, 52] := true;
_YearOK[1, 3, 52] := true;
_YearOK[0, 3, 53] := true;
_YearOK[1, 3, 53] := true;
_YearOK[0, 3, 54] := true;
_YearOK[1, 3, 54] := true;
_YearOK[0, 3, 55] := true;
_YearOK[1, 3, 55] := true;
_YearOK[0, 4, 55] := true;
_YearOK[1, 4, 55] := true;
_YearOK[0, 5, 55] := true;
_YearOK[1, 5, 55] := true;
_YearOK[0, 6, 55] := true;
_YearOK[1, 6, 55] := true;
_YearOK[0, 7, 55] := true;
_YearOK[1, 7, 55] := true;
_YearOK[0, 8, 55] := true;
_YearOK[1, 8, 55] := true;
_YearOK[0, 9, 55] := true;
_YearOK[1, 9, 55] := true;
_YearOK[0, 10, 55] := true;
_YearOK[1, 10, 55] := true;
_YearOK[0, 11, 55] := true;
_YearOK[1, 11, 55] := true;
_YearOK[0, 12, 55] := true;
_YearOK[1, 12, 55] := true;
_YearOK[0, 13, 55] := true;
_YearOK[1, 13, 55] := true;
_YearOK[0, 14, 55] := true;
_YearOK[1, 14, 55] := true;
_YearOK[0, 15, 55] := true;
_YearOK[1, 15, 55] := true;
_YearOK[0, 16, 55] := true;
_YearOK[1, 16, 55] := true;
_YearOK[0, 17, 55] := true;
_YearOK[1, 17, 55] := true;
_YearOK[0, 18, 55] := true;
_YearOK[1, 18, 55] := true;
_YearOK[0, 19, 55] := true;
_YearOK[1, 19, 55] := true;
_YearOK[0, 20, 55] := true;
_YearOK[1, 20, 55] := true;
_YearOK[0, 21, 55] := true;
_YearOK[1, 21, 55] := true;
_YearOK[0, 22, 55] := true;
_YearOK[1, 22, 55] := true;
_YearOK[0, 23, 55] := true;
_YearOK[1, 23, 55] := true;
_YearOK[0, 24, 55] := true;
_YearOK[1, 24, 55] := true;
_YearOK[0, 25, 55] := true;
_YearOK[1, 25, 55] := true;
_YearOK[0, 26, 55] := true;
_YearOK[1, 26, 55] := true;
_YearOK[0, 27, 55] := true;
_YearOK[1, 27, 55] := true;
_YearOK[0, 28, 55] := true;
_YearOK[1, 28, 55] := true;
_YearOK[0, 29, 55] := true;
_YearOK[1, 29, 55] := true;
_YearOK[0, 30, 55] := true;
_YearOK[1, 30, 55] := true;
_YearOK[0, 31, 55] := true;
_YearOK[1, 31, 55] := true;
_YearOK[0, 32, 55] := true;
_YearOK[1, 32, 55] := true;
_YearOK[0, 33, 55] := true;
_YearOK[1, 33, 55] := true;
_YearOK[0, 34, 55] := true;
```

```
  _YearOK[1, 34, 55] := true;
  _YearOK[0, 35, 55] := true;
  _YearOK[1, 35, 55] := true;
  _YearOK[0, 36, 55] := true;
  _YearOK[1, 36, 55] := true;
  _YearOK[0, 37, 55] := true;
  _YearOK[1, 37, 55] := true;
  _YearOK[0, 38, 55] := true;
  _YearOK[1, 38, 55] := true;
  _YearOK[0, 39, 55] := true;
  _YearOK[1, 39, 55] := true;
  _YearOK[0, 40, 55] := true;
  _YearOK[1, 40, 55] := true;
  _YearOK[0, 41, 55] := true;
  _YearOK[1, 41, 55] := true;
  _YearOK[0, 42, 55] := true;
  _YearOK[1, 42, 55] := true;
  _YearOK[0, 43, 55] := true;
  _YearOK[1, 43, 55] := true;
  _YearOK[0, 44, 55] := true;
  _YearOK[1, 44, 55] := true;
  _YearOK[0, 45, 55] := true;
  _YearOK[1, 45, 55] := true;
  _YearOK[0, 46, 55] := true;
  _YearOK[1, 46, 55] := true;
  _YearOK[0, 47, 55] := true;
  _YearOK[1, 47, 55] := true;
  _YearOK[0, 48, 55] := true;
  _YearOK[1, 48, 55] := true;
  _YearOK[0, 49, 55] := true;
  _YearOK[1, 49, 55] := true;
  _YearOK[0, 50, 55] := true;
  _YearOK[1, 50, 55] := true;
  _YearOK[0, 51, 55] := true;
  _YearOK[1, 51, 55] := true;
  _YearOK[0, 52, 55] := true;
  _YearOK[1, 52, 55] := true;
  _YearOK[0, 53, 55] := true;
  _YearOK[1, 53, 55] := true;
  _YearOK[0, 54, 55] := true;
  _YearOK[1, 54, 55] := true;
  _YearOK[0, 55, 55] := true;
  _YearOK[1, 55, 55] := true;
  _YearOK[0, 56, 55] := true;
  _YearOK[1, 56, 55] := true;
  _YearOK[0, 57, 55] := true;
  _YearOK[1, 57, 55] := true;
  _YearOK[0, 58, 55] := true;
  _YearOK[1, 58, 55] := true;
  _YearOK[0, 59, 55] := true;
  _YearOK[1, 59, 55] := true;
  _YearOK[0, 60, 55] := true;
  _YearOK[1, 60, 55] := true;
  _YearOK[0, 61, 55] := true;
  _YearOK[1, 61, 55] := true;
  _YearOK[0, 62, 55] := true;
  _YearOK[1, 62, 55] := true;
  _YearOK[0, 63, 55] := true;
  _YearOK[1, 63, 55] := true;
  _YearOK[0, 64, 55] := true;
  _YearOK[1, 64, 55] := true;
  _YearOK[0, 65, 55] := true;
  _YearOK[1, 65, 55] := true;
  _YearOK[0, 66, 55] := true;
  _YearOK[1, 66, 55] := true;
  _YearOK[0, 67, 55] := true;
  _YearOK[1, 67, 55] := true;
  _YearOK[0, 68, 55] := true;
  _YearOK[1, 68, 55] := true;
  _YearOK[0, 69, 55] := true;
  _YearOK[1, 69, 55] := true;
  _YearOK[0, 70, 55] := true;
  _YearOK[1, 70, 55] := true;
  _YearOK[0, 71, 55] := true;
  _YearOK[1, 71, 55] := true;
```

```
_YearOK[0, 72, 55] := true;
_YearOK[1, 72, 55] := true;
_YearOK[0, 73, 55] := true;
_YearOK[1, 73, 55] := true;
_YearOK[0, 74, 55] := true;
_YearOK[1, 74, 55] := true;
_YearOK[0, 75, 55] := true;
_YearOK[1, 75, 55] := true;
_YearOK[0, 76, 55] := true;
_YearOK[1, 76, 55] := true;
_YearOK[0, 77, 55] := true;
_YearOK[1, 77, 55] := true;
_YearOK[0, 78, 55] := true;
_YearOK[1, 78, 55] := true;
_YearOK[0, 79, 55] := true;
_YearOK[1, 79, 55] := true;
_YearOK[0, 80, 55] := true;
_YearOK[1, 80, 55] := true;
_YearOK[0, 81, 55] := true;
_YearOK[1, 81, 55] := true;
_YearOK[0, 82, 55] := true;
_YearOK[1, 82, 55] := true;
_YearOK[0, 83, 55] := true;
_YearOK[1, 83, 55] := true;
_YearOK[0, 84, 55] := true;
_YearOK[1, 84, 55] := true;
_YearOK[0, 85, 55] := true;
_YearOK[1, 85, 55] := true;
_YearOK[0, 86, 55] := true;
_YearOK[1, 86, 55] := true;
_YearOK[0, 87, 55] := true;
_YearOK[1, 87, 55] := true;
_YearOK[0, 88, 55] := true;
_YearOK[1, 88, 55] := true;
_YearOK[0, 89, 55] := true;
_YearOK[1, 89, 55] := true;
_YearOK[0, 90, 55] := true;
_YearOK[1, 90, 55] := true;
_YearOK[0, 91, 55] := true;
_YearOK[1, 91, 55] := true;
_YearOK[0, 92, 55] := true;
_YearOK[1, 92, 55] := true;
_YearOK[0, 93, 55] := true;
_YearOK[1, 93, 55] := true;
_YearOK[0, 94, 55] := true;
_YearOK[1, 94, 55] := true;
_YearOK[0, 95, 55] := true;
_YearOK[1, 95, 55] := true;
_YearOK[0, 96, 55] := true;
_YearOK[1, 96, 55] := true;
_YearOK[0, 97, 55] := true;
_YearOK[1, 97, 55] := true;
_YearOK[0, 98, 55] := true;
_YearOK[1, 98, 55] := true;
_YearOK[0, 99, 55] := true;
_YearOK[1, 99, 55] := true;
_YearOK[0, 100, 55] := true;
_YearOK[1, 100, 55] := true;
_YearOK[0, 3, 56] := true;
_YearOK[1, 3, 56] := true;
_YearOK[0, 4, 56] := true;
_YearOK[1, 4, 56] := true;
_YearOK[0, 5, 56] := true;
_YearOK[1, 5, 56] := true;
_YearOK[0, 6, 56] := true;
_YearOK[1, 6, 56] := true;
_YearOK[0, 7, 56] := true;
_YearOK[1, 7, 56] := true;
_YearOK[0, 8, 56] := true;
_YearOK[1, 8, 56] := true;
_YearOK[0, 9, 56] := true;
_YearOK[1, 9, 56] := true;
_YearOK[0, 10, 56] := true;
_YearOK[1, 10, 56] := true;
```

```
  _YearOK[0, 11, 56] := true;
  _YearOK[1, 11, 56] := true;
  _YearOK[0, 12, 56] := true;
  _YearOK[1, 12, 56] := true;
  _YearOK[0, 13, 56] := true;
  _YearOK[1, 13, 56] := true;
  _YearOK[0, 14, 56] := true;
  _YearOK[1, 14, 56] := true;
  _YearOK[0, 15, 56] := true;
  _YearOK[1, 15, 56] := true;
  _YearOK[0, 16, 56] := true;
  _YearOK[1, 16, 56] := true;
  _YearOK[0, 17, 56] := true;
  _YearOK[1, 17, 56] := true;
  _YearOK[0, 18, 56] := true;
  _YearOK[1, 18, 56] := true;
  _YearOK[0, 19, 56] := true;
  _YearOK[1, 19, 56] := true;
  _YearOK[0, 20, 56] := true;
  _YearOK[1, 20, 56] := true;
  _YearOK[0, 21, 56] := true;
  _YearOK[1, 21, 56] := true;
  _YearOK[0, 22, 56] := true;
  _YearOK[1, 22, 56] := true;
  _YearOK[0, 23, 56] := true;
  _YearOK[1, 23, 56] := true;
  _YearOK[0, 24, 56] := true;
  _YearOK[1, 24, 56] := true;
  _YearOK[0, 25, 56] := true;
  _YearOK[1, 25, 56] := true;
  _YearOK[0, 26, 56] := true;
  _YearOK[1, 26, 56] := true;
  _YearOK[0, 27, 56] := true;
  _YearOK[1, 27, 56] := true;
  _YearOK[0, 28, 56] := true;
  _YearOK[1, 28, 56] := true;
  _YearOK[0, 29, 56] := true;
  _YearOK[1, 29, 56] := true;
  _YearOK[0, 30, 56] := true;
  _YearOK[1, 30, 56] := true;
  _YearOK[0, 31, 56] := true;
  _YearOK[1, 31, 56] := true;
  _YearOK[0, 32, 56] := true;
  _YearOK[1, 32, 56] := true;
  _YearOK[0, 33, 56] := true;
  _YearOK[1, 33, 56] := true;
  _YearOK[0, 34, 56] := true;
  _YearOK[1, 34, 56] := true;
  _YearOK[0, 35, 56] := true;
  _YearOK[1, 35, 56] := true;
  _YearOK[0, 36, 56] := true;
  _YearOK[1, 36, 56] := true;
  _YearOK[0, 37, 56] := true;
  _YearOK[1, 37, 56] := true;
  _YearOK[0, 38, 56] := true;
  _YearOK[1, 38, 56] := true;
  _YearOK[0, 39, 56] := true;
  _YearOK[1, 39, 56] := true;
  _YearOK[0, 40, 56] := true;
  _YearOK[1, 40, 56] := true;
  _YearOK[0, 41, 56] := true;
  _YearOK[1, 41, 56] := true;
  _YearOK[0, 42, 56] := true;
  _YearOK[1, 42, 56] := true;
  _YearOK[0, 43, 56] := true;
  _YearOK[1, 43, 56] := true;
  _YearOK[0, 44, 56] := true;
  _YearOK[1, 44, 56] := true;
  _YearOK[0, 45, 56] := true;
  _YearOK[1, 45, 56] := true;
  _YearOK[0, 46, 56] := true;
  _YearOK[1, 46, 56] := true;
  _YearOK[0, 47, 56] := true;
  _YearOK[1, 47, 56] := true;
  _YearOK[0, 48, 56] := true;
```

```
_YearOK[1, 48, 56] := true;
_YearOK[0, 49, 56] := true;
_YearOK[1, 49, 56] := true;
_YearOK[0, 50, 56] := true;
_YearOK[1, 50, 56] := true;
_YearOK[0, 51, 56] := true;
_YearOK[1, 51, 56] := true;
_YearOK[0, 52, 56] := true;
_YearOK[1, 52, 56] := true;
_YearOK[0, 53, 56] := true;
_YearOK[1, 53, 56] := true;
_YearOK[0, 54, 56] := true;
_YearOK[1, 54, 56] := true;
_YearOK[0, 55, 56] := true;
_YearOK[1, 55, 56] := true;
_YearOK[0, 56, 56] := true;
_YearOK[1, 56, 56] := true;
_YearOK[0, 57, 56] := true;
_YearOK[1, 57, 56] := true;
_YearOK[0, 58, 56] := true;
_YearOK[1, 58, 56] := true;
_YearOK[0, 59, 56] := true;
_YearOK[1, 59, 56] := true;
_YearOK[0, 60, 56] := true;
_YearOK[1, 60, 56] := true;
_YearOK[0, 61, 56] := true;
_YearOK[1, 61, 56] := true;
_YearOK[0, 62, 56] := true;
_YearOK[1, 62, 56] := true;
_YearOK[0, 63, 56] := true;
_YearOK[1, 63, 56] := true;
_YearOK[0, 64, 56] := true;
_YearOK[1, 64, 56] := true;
_YearOK[0, 65, 56] := true;
_YearOK[1, 65, 56] := true;
_YearOK[0, 66, 56] := true;
_YearOK[1, 66, 56] := true;
_YearOK[0, 67, 56] := true;
_YearOK[1, 67, 56] := true;
_YearOK[0, 68, 56] := true;
_YearOK[1, 68, 56] := true;
_YearOK[0, 69, 56] := true;
_YearOK[1, 69, 56] := true;
_YearOK[0, 70, 56] := true;
_YearOK[1, 70, 56] := true;
_YearOK[0, 71, 56] := true;
_YearOK[1, 71, 56] := true;
_YearOK[0, 72, 56] := true;
_YearOK[1, 72, 56] := true;
_YearOK[0, 73, 56] := true;
_YearOK[1, 73, 56] := true;
_YearOK[0, 74, 56] := true;
_YearOK[1, 74, 56] := true;
_YearOK[0, 75, 56] := true;
_YearOK[1, 75, 56] := true;
_YearOK[0, 76, 56] := true;
_YearOK[1, 76, 56] := true;
_YearOK[0, 77, 56] := true;
_YearOK[1, 77, 56] := true;
_YearOK[0, 78, 56] := true;
_YearOK[1, 78, 56] := true;
_YearOK[0, 79, 56] := true;
_YearOK[1, 79, 56] := true;
_YearOK[0, 80, 56] := true;
_YearOK[1, 80, 56] := true;
_YearOK[0, 81, 56] := true;
_YearOK[1, 81, 56] := true;
_YearOK[0, 82, 56] := true;
_YearOK[1, 82, 56] := true;
_YearOK[0, 83, 56] := true;
_YearOK[1, 83, 56] := true;
_YearOK[0, 84, 56] := true;
_YearOK[1, 84, 56] := true;
_YearOK[0, 85, 56] := true;
```

```
_YearOK[1, 85, 56] := true;
_YearOK[0, 86, 56] := true;
_YearOK[1, 86, 56] := true;
_YearOK[0, 87, 56] := true;
_YearOK[1, 87, 56] := true;
_YearOK[0, 88, 56] := true;
_YearOK[1, 88, 56] := true;
_YearOK[0, 89, 56] := true;
_YearOK[1, 89, 56] := true;
_YearOK[0, 90, 56] := true;
_YearOK[1, 90, 56] := true;
_YearOK[0, 91, 56] := true;
_YearOK[1, 91, 56] := true;
_YearOK[0, 92, 56] := true;
_YearOK[1, 92, 56] := true;
_YearOK[0, 93, 56] := true;
_YearOK[1, 93, 56] := true;
_YearOK[0, 94, 56] := true;
_YearOK[1, 94, 56] := true;
_YearOK[0, 95, 56] := true;
_YearOK[1, 95, 56] := true;
_YearOK[0, 96, 56] := true;
_YearOK[1, 96, 56] := true;
_YearOK[0, 97, 56] := true;
_YearOK[1, 97, 56] := true;
_YearOK[0, 98, 56] := true;
_YearOK[1, 98, 56] := true;
_YearOK[0, 99, 56] := true;
_YearOK[1, 99, 56] := true;
_YearOK[0, 100, 56] := true;
_YearOK[1, 100, 56] := true;
_YearOK[0, 3, 57] := true;
_YearOK[1, 3, 57] := true;
_YearOK[0, 4, 57] := true;
_YearOK[1, 4, 57] := true;
_YearOK[0, 5, 57] := true;
_YearOK[1, 5, 57] := true;
_YearOK[0, 6, 57] := true;
_YearOK[1, 6, 57] := true;
_YearOK[0, 7, 57] := true;
_YearOK[1, 7, 57] := true;
_YearOK[0, 8, 57] := true;
_YearOK[1, 8, 57] := true;
_YearOK[0, 9, 57] := true;
_YearOK[1, 9, 57] := true;
_YearOK[0, 10, 57] := true;
_YearOK[1, 10, 57] := true;
_YearOK[0, 11, 57] := true;
_YearOK[1, 11, 57] := true;
_YearOK[0, 12, 57] := true;
_YearOK[1, 12, 57] := true;
_YearOK[0, 13, 57] := true;
_YearOK[1, 13, 57] := true;
_YearOK[0, 14, 57] := true;
_YearOK[1, 14, 57] := true;
_YearOK[0, 15, 57] := true;
_YearOK[1, 15, 57] := true;
_YearOK[0, 16, 57] := true;
_YearOK[1, 16, 57] := true;
_YearOK[0, 17, 57] := true;
_YearOK[1, 17, 57] := true;
_YearOK[0, 18, 57] := true;
_YearOK[1, 18, 57] := true;
_YearOK[0, 19, 57] := true;
_YearOK[1, 19, 57] := true;
_YearOK[0, 20, 57] := true;
_YearOK[1, 20, 57] := true;
_YearOK[0, 21, 57] := true;
_YearOK[1, 21, 57] := true;
_YearOK[0, 22, 57] := true;
_YearOK[1, 22, 57] := true;
_YearOK[0, 23, 57] := true;
_YearOK[1, 23, 57] := true;
_YearOK[0, 24, 57] := true;
_YearOK[1, 24, 57] := true;
```

```
  _YearOK[0, 25, 57] := true;
  _YearOK[1, 25, 57] := true;
  _YearOK[0, 26, 57] := true;
  _YearOK[1, 26, 57] := true;
  _YearOK[0, 27, 57] := true;
  _YearOK[1, 27, 57] := true;
  _YearOK[0, 28, 57] := true;
  _YearOK[1, 28, 57] := true;
  _YearOK[0, 29, 57] := true;
  _YearOK[1, 29, 57] := true;
  _YearOK[0, 30, 57] := true;
  _YearOK[1, 30, 57] := true;
  _YearOK[0, 31, 57] := true;
  _YearOK[1, 31, 57] := true;
  _YearOK[0, 32, 57] := true;
  _YearOK[1, 32, 57] := true;
  _YearOK[0, 33, 57] := true;
  _YearOK[1, 33, 57] := true;
  _YearOK[0, 34, 57] := true;
  _YearOK[1, 34, 57] := true;
  _YearOK[0, 35, 57] := true;
  _YearOK[1, 35, 57] := true;
  _YearOK[0, 36, 57] := true;
  _YearOK[1, 36, 57] := true;
  _YearOK[0, 37, 57] := true;
  _YearOK[1, 37, 57] := true;
  _YearOK[0, 38, 57] := true;
  _YearOK[1, 38, 57] := true;
  _YearOK[0, 39, 57] := true;
  _YearOK[1, 39, 57] := true;
  _YearOK[0, 40, 57] := true;
  _YearOK[1, 40, 57] := true;
  _YearOK[0, 41, 57] := true;
  _YearOK[1, 41, 57] := true;
  _YearOK[0, 42, 57] := true;
  _YearOK[1, 42, 57] := true;
  _YearOK[0, 43, 57] := true;
  _YearOK[1, 43, 57] := true;
  _YearOK[0, 44, 57] := true;
  _YearOK[1, 44, 57] := true;
  _YearOK[0, 45, 57] := true;
  _YearOK[1, 45, 57] := true;
  _YearOK[0, 46, 57] := true;
  _YearOK[1, 46, 57] := true;
  _YearOK[0, 47, 57] := true;
  _YearOK[1, 47, 57] := true;
  _YearOK[0, 48, 57] := true;
  _YearOK[1, 48, 57] := true;
  _YearOK[0, 49, 57] := true;
  _YearOK[1, 49, 57] := true;
  _YearOK[0, 50, 57] := true;
  _YearOK[1, 50, 57] := true;
  _YearOK[0, 51, 57] := true;
  _YearOK[1, 51, 57] := true;
  _YearOK[0, 52, 57] := true;
  _YearOK[1, 52, 57] := true;
  _YearOK[0, 53, 57] := true;
  _YearOK[1, 53, 57] := true;
  _YearOK[0, 54, 57] := true;
  _YearOK[1, 54, 57] := true;
  _YearOK[0, 55, 57] := true;
  _YearOK[1, 55, 57] := true;
  _YearOK[0, 56, 57] := true;
  _YearOK[1, 56, 57] := true;
  _YearOK[0, 57, 57] := true;
  _YearOK[1, 57, 57] := true;
  _YearOK[0, 58, 57] := true;
  _YearOK[1, 58, 57] := true;
  _YearOK[0, 59, 57] := true;
  _YearOK[1, 59, 57] := true;
  _YearOK[0, 60, 57] := true;
  _YearOK[1, 60, 57] := true;
  _YearOK[0, 61, 57] := true;
  _YearOK[1, 61, 57] := true;
```

```
  _YearOK[0, 62, 57] := true;
  _YearOK[1, 62, 57] := true;
  _YearOK[0, 63, 57] := true;
  _YearOK[1, 63, 57] := true;
  _YearOK[0, 64, 57] := true;
  _YearOK[1, 64, 57] := true;
  _YearOK[0, 65, 57] := true;
  _YearOK[1, 65, 57] := true;
  _YearOK[0, 66, 57] := true;
  _YearOK[1, 66, 57] := true;
  _YearOK[0, 67, 57] := true;
  _YearOK[1, 67, 57] := true;
  _YearOK[0, 68, 57] := true;
  _YearOK[1, 68, 57] := true;
  _YearOK[0, 69, 57] := true;
  _YearOK[1, 69, 57] := true;
  _YearOK[0, 70, 57] := true;
  _YearOK[1, 70, 57] := true;
  _YearOK[0, 71, 57] := true;
  _YearOK[1, 71, 57] := true;
  _YearOK[0, 72, 57] := true;
  _YearOK[1, 72, 57] := true;
  _YearOK[0, 73, 57] := true;
  _YearOK[1, 73, 57] := true;
  _YearOK[0, 74, 57] := true;
  _YearOK[1, 74, 57] := true;
  _YearOK[0, 75, 57] := true;
  _YearOK[1, 75, 57] := true;
  _YearOK[0, 76, 57] := true;
  _YearOK[1, 76, 57] := true;
  _YearOK[0, 77, 57] := true;
  _YearOK[1, 77, 57] := true;
  _YearOK[0, 78, 57] := true;
  _YearOK[1, 78, 57] := true;
  _YearOK[0, 79, 57] := true;
  _YearOK[1, 79, 57] := true;
  _YearOK[0, 80, 57] := true;
  _YearOK[1, 80, 57] := true;
  _YearOK[0, 81, 57] := true;
  _YearOK[1, 81, 57] := true;
  _YearOK[0, 82, 57] := true;
  _YearOK[1, 82, 57] := true;
  _YearOK[0, 83, 57] := true;
  _YearOK[1, 83, 57] := true;
  _YearOK[0, 84, 57] := true;
  _YearOK[1, 84, 57] := true;
  _YearOK[0, 85, 57] := true;
  _YearOK[1, 85, 57] := true;
  _YearOK[0, 86, 57] := true;
  _YearOK[1, 86, 57] := true;
  _YearOK[0, 87, 57] := true;
  _YearOK[1, 87, 57] := true;
  _YearOK[0, 88, 57] := true;
  _YearOK[1, 88, 57] := true;
  _YearOK[0, 89, 57] := true;
  _YearOK[1, 89, 57] := true;
  _YearOK[0, 90, 57] := true;
  _YearOK[1, 90, 57] := true;
  _YearOK[0, 91, 57] := true;
  _YearOK[1, 91, 57] := true;
  _YearOK[0, 92, 57] := true;
  _YearOK[1, 92, 57] := true;
  _YearOK[0, 93, 57] := true;
  _YearOK[1, 93, 57] := true;
  _YearOK[0, 94, 57] := true;
  _YearOK[1, 94, 57] := true;
  _YearOK[0, 95, 57] := true;
  _YearOK[1, 95, 57] := true;
  _YearOK[0, 96, 57] := true;
  _YearOK[1, 96, 57] := true;
  _YearOK[0, 97, 57] := true;
  _YearOK[1, 97, 57] := true;
  _YearOK[0, 98, 57] := true;
  _YearOK[1, 98, 57] := true;
  _YearOK[0, 99, 57] := true;
```

```
  _YearOK[1, 99, 57] := true;
  _YearOK[0, 100, 57] := true;
  _YearOK[1, 100, 57] := true;

end;

function TCCCompiledCostOption._DumpVars: string;
var s: string;
begin
  s := '';
  s := s + 'aw: ' + _Variables.aw^.ToString + #13#10;
  s := s + 'b_annual: ' + _Variables.b_annual^.ToString + #13#10;
  s := s + 'b_exceed: ' + _Variables.b_exceed^.ToString + #13#10;
  s := s + 'b_nondetect: ' + _Variables.b_nondetect^.ToString + #13#10;
  s := s + 'b_quarterly: ' + _Variables.b_quarterly^.ToString + #13#10;
  s := s + 'b_source: ' + _Variables.b_source^.ToString + #13#10;
  s := s + 'b_treat: ' + _Variables.b_treat^.ToString + #13#10;
  s := s + 'b_triennial: ' + _Variables.b_triennial^.ToString + #13#10;
  s := s + 'epa533: ' + _Variables.epa533^.ToString + #13#10;
  s := s + 'epa537: ' + _Variables.epa537^.ToString + #13#10;
  s := s + 'statepfas: ' + _Variables.statepfas^.ToString + #13#10;

  Result := s;
end;

constructor TCCCompiledCostOption.create;
begin
  inherited create;

end;

end.

```

code\CostModules\CompiledCostTemplate.pas

```
unit CompiledCostTemplate;

//Autogenerated unit - DO NOT ALTER
//Generated - (*DATE*)
//Baseline Workbook - (*WORKBOOKBASELINE*)
//Found variables in workbook - (*NUMVARSBASELINE*)
//Found costing steps - 9999
//Option Workbook - (*WORKBOOKOPTION*)
//Found variables in workbook - (*NUMVARSOPTION*)
//Found costing steps - 9998

interface

uses Math, Classes, SysUtils, SafewaterGlobals, SafewaterUtility, SafewaterPWSRecords,
     SafewaterModelData;

type
  TCCValueStore = array[0..9997] of double;
  //max 6 disc rates including 0
  TCCValueAccum = array[0..5,0..9997] of pdouble;
  TCCValueYearlyAccum = array[0..100,0..5,0..9997] of pdouble;
  TCCNameStore = array[0..9997] of string;
  TCCValueStoreFlag = array[0..9997] of boolean;
  //added large,small system split
  TCCCalcYearStore = array[0..1,0..100,0..9997] of boolean;

  _VariablesRecBASELINE = record
(*VARIABLESBASELINE*)
  end;
  _VariablesRecOPTION = record
(*VARIABLESOPTION*)
  end;

  TCCCompiledCost = class
    _CalcCost,_Cost,_Hours,_Labor  : TCCValueStore;
    _pCost,_pCost_pws: TCCValueAccum;
    _pCostY,_pCostY_pws: TCCValueYearlyAccum;
```

```
38      _Names, _Cat : TCCNameStore;
39      _EPCost: TCCValueStoreFlag;
40      _YearOK : TCCCalcYearStore;
41      _HighDRIdx: integer;
42      TotEval : int64;
43      constructor create();
44      procedure _Reset(ResetSystem: boolean);
45      procedure _SetSingleYear(Yr: integer);
46      procedure _SetYears; virtual; abstract;
47      procedure _SetNames; virtual; abstract;
48      procedure _SetEPCosts; virtual; abstract;
49      procedure _LoadVars; virtual; abstract;
50      function _LoadVarsOK : boolean; virtual; abstract;
51      procedure _SetVarPointer(const name : string; const pd : PDouble); virtual; abstract;
52      procedure _SetOutPointer(var O: TMDRCostOutputs; var Y : TMDRYearlyCostOutputs); virtual;
53      procedure _Evaluate(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer); virtual; abstract;
54      procedure _EvaluateSystem(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer); virtual; abstract;
55      function _DumpVars : string; virtual; abstract;
56      function _DumpYearTable : TStringList;
57      function _DumpCalcs : TStringList;
58    end;
59
60    TCCCompiledCostBaseline = class(TCCCompiledCost)
61      _Variables : _VariablesRecBaseline;
62
63      constructor create();
64      procedure _SetYears; override;
65      procedure _SetNames; override;
66      procedure _SetEPCosts; override;
67      procedure _LoadVars; override;
68      function _LoadVarsOk: boolean; override;
69      procedure _SetVarPointer(const name : string; const pd : PDouble); override;
70      procedure _Evaluate(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer); override;
71      procedure _EvaluateSystem(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer); override;
72      function _DumpVars : string; override;
73    end;
74
75    TCCCompiledCostOption = class(TCCCompiledCost)
76      _Variables : _VariablesRecOption;
77
78      constructor create();
79      procedure _SetYears; override;
80      procedure _SetNames; override;
81      procedure _SetEPCosts; override;
82      procedure _LoadVars; override;
83      function _LoadVarsOk: boolean; override;
84      procedure _SetVarPointer(const name : string; const pd : PDouble); override;
85      procedure _Evaluate(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer); override;
86      procedure _EvaluateSystem(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer); override;
87      function _DumpVars : string; override;
88    end;
89
90
91  const
92    _CWorkBookBASELINE = '(*WORKBOOKBASELINE*)';
93    _CWorkBookOPTION = '(*WORKBOOKOPTION*)';
94    _CWorkBookDate = '(*DATE*)';
95
96  var
97    _UseCompiled : boolean = true;
98
99  implementation
100 type
101   TSSArray = array[swGroundWater..swSurfaceWater,ss0_100..ss1Mp] of double;
102
103 const
104   Dummy=1;
105 (*CONSTANTS*)
106
107
108 { TCCCCompiledCost }
109
110 constructor TCCCompiledCost.create;
111 begin
```

```
  TotEval:=0;
  _SetYears;
  _SetNames;
  _SetEPCosts;
  for var i:=0 to high(_pCost) do begin
    for var j:=0 to high(_pCost[i]) do begin
      _pCost[i,j] := nil;
      _pCost_pws[i,j] := nil;
    end;
  end;
end;

function TCCCompiledCost._DumpCalcs: TStringList;
begin
  Result := TStringList.Create;
  for var i :=0 to high(_Cost) do begin
    Result.Add(_Names[i]+': '+floattostr(_Cost[i]));
  end;
end;

function TCCCompiledCost._DumpYearTable: TStringList;
begin
  Result := TStringList.Create;
  var tmps:string;
  for var sz := 0 to 1 do begin
    for var c := Low(_Names) to High(_Names) do begin
       tmps := _Names[c] +','+sz.ToString+',';
       for var y := 0 to 100 do begin
         var b :='';
         if _YearOK[0,y,c] then b:='1';
         tmps := tmps + b+',';
       end;
       Result.Add(tmps);
    end;
  end;

end;

procedure TCCCompiledCost._Reset(ResetSystem: boolean);
var i : integer;
begin
  for i:=0 to high(_CalcCost) do begin
    if (ResetSystem and not _EPCOst[i]) or (not ResetSystem and _EPCOst[i]) then begin
      _CalcCost[i]:=0;
      _Cost[i]:=0;
      //_Hours[i]:=0;
      //_OM[i]:=0;
      //_Labor[i]:=0;
    end;
  end;
end;

procedure TCCCompiledCost._SetOutPointer(var O: TMDRCostOutputs; var Y : TMDRYearlyCostOutputs);
//O is indexed by discount rate - Y is indexe by year for discount rate 0 only
begin
  //TODO make these more general
  _HighDRIdx := high(o);
  var iy := high(Y);
  var yy: integer;
  for var i:= 0 to high(_Cat) do begin
    var s := _Cat[i];
    for var dr := 0 to _HighDRIdx do begin
      if s = 'Administration - System' then begin
        _pCost[dr,i] := @O[dr].PWSImpAdmin;
        _pCost_pws[dr,i] := @O[dr].PWSImpAdmin_pws;
        for yy := 0 to iy do begin
          _pCostY[yy,dr,i] := @Y[yy,dr].PWSImpAdmin;
          _pCostY_pws[yy,dr,i] := @Y[yy,dr].PWSImpAdmin_pws;
        end;
      end else
      if s = 'Administration - Primacy Agency' then begin
        _pCost[dr,i] := @O[dr].AgencyImpAdmin;
        _pCost_pws[dr,i] := @O[dr].DontCare;
        for yy := 0 to iy do begin
          _pCostY[yy,dr,i] := @Y[yy,dr].AgencyImpAdmin;
```

```
187           _pCostY_pws[yy,dr,i] := @Y[yy,dr].DontCare;
188         end;
189       end else
190       if s = 'Treatment Administration - System' then begin
191         _pCost[dr,i] := @O[dr].PWSTreatmentAdmin;
192         _pCost_pws[dr,i] := @O[dr].PWSTreatmentAdmin_pws;
193         for yy := 0 to iy do begin
194           _pCostY[yy,dr,i] := @Y[yy,dr].PWSTreatmentAdmin;
195           _pCostY_pws[yy,dr,i] := @Y[yy,dr].PWSTreatmentAdmin_pws;
196         end;
197       end else
198       if s = 'Treatment Administration - Primacy Agency' then begin
199         _pCost[dr,i] := @O[dr].AgencyTreatmentAdmin;
200         _pCost_pws[dr,i] := @O[dr].DontCare;
201         for yy := 0 to iy do begin
202           _pCostY[yy,dr,i] := @Y[yy,dr].AgencyTreatmentAdmin;
203           _pCostY_pws[yy,dr,i] := @Y[yy,dr].DontCare;
204         end;
205       end else
206       if s = 'Monitoring - System' then begin
207         _pCost[dr,i] := @O[dr].PWSSampling;
208         _pCost_pws[dr,i] := @O[dr].PWSSampling_pws;
209         for yy := 0 to iy do begin
210           _pCostY[yy,dr,i] := @Y[yy,dr].PWSSampling;
211           _pCostY_pws[yy,dr,i] := @Y[yy,dr].PWSSampling_pws;
212         end;
213       end else
214       if s = 'Treatment - System' then begin
215         //ignore - handled elswhere with WBS curves
216       end else
217       if s = 'Monitoring - Primacy Agency' then begin
218         _pCost[dr,i] := @O[dr].AgencyImpSampling;
219         _pCost_pws[dr,i] := @O[dr].DontCare;
220         for yy := 0 to iy do begin
221           _pCostY[yy,dr,i] := @Y[yy,dr].AgencyImpSampling;
222           _pCostY_pws[yy,dr,i] := @Y[yy,dr].DontCare;
223         end;
224
225       end else
226         raise Exception.Create('Aggregation Category Not Supported: '+s);
227     end;
228   end;
229 end;
230
231 procedure TCCCompiledCost._SetSingleYear(Yr: integer);
232 begin
233   for var s := 0 to 1 do begin
234     for var y := 0 to 100 do begin
235       if y=Yr then continue;
236       for var c := 0 to high(_YearOK[s,y]) do
237         _YearOK[s,y,c] := false;
238     end;
239   end;
240 end;
241
242 { TCCCompiledCostBaseline }
243
244 procedure TCCCompiledCostBaseline._LoadVars;
245 begin
246 (*LoadVarsBASELINE*)
247 end;
248
249 function TCCCompiledCostBaseline._LoadVarsOk: boolean;
250 begin
251 (*LoadVarsOkBASELINE*)
252 end;
253
254 procedure TCCCompiledCostBaseline._Evaluate(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer);
255 begin
256   _Reset(False);
257 //  _LoadVars();
258   var sz := 0;
259   var v:double;
260   var DRIdx: integer;
```

```
261   if P.SystemSize >= ss3301_10k then sz:=1;
262
263   with _Variables do begin
264 (*EVALUATEBASELINE*)
265   end;
266 end;
267
268 procedure TCCCompiledCostBaseline._EvaluateSystem(const P: TPWSRecord;
269   const Yr: integer; const Data: PSWModelInputData; const PeriodAdjust: integer);
270 begin
271   _Reset(True);
272 //  _LoadVars();
273   var sz := 0;
274   var v:double;
275   var DRIdx: integer;
276   if P.SystemSize >= ss3301_10k then sz:=1;
277
278   with _Variables do begin
279 (*EVALUATESYSTEMBASELINE*)
280   end;
281 end;
282
283 procedure TCCCompiledCostBaseline._SetEPCosts;
284 begin
285   for var i := low(_EPCost) to high(_EPCost) do _EPCost[i] := true;
286 (*SetEPCostsBASELINE*)
287 end;
288
289 procedure TCCCompiledCostBaseline._SetNames;
290 begin
291 (*SetNamesBASELINE*)
292 end;
293
294 procedure TCCCompiledCostBaseline._SetVarPointer(const name : string; const pd : PDouble);
295 var s : string;
296 begin
297   s := lowercase(name);
298 (*_SetVarPointerBASELINE*)
299 end;
300
301 procedure TCCCompiledCostBaseline._SetYears;
302 begin
303   //Size, Year, CostID
304 (*SetYearsBASELINE*)
305 end;
306
307 function TCCCompiledCostBaseline._DumpVars: string;
308 var s : string;
309 begin
310   s:='';
311 (*DumpVarsBASELINE*)
312   result:=s;
313 end;
314
315 constructor TCCCompiledCostBaseline.create;
316 begin
317   inherited create;
318 (*SetStateBASELINE*)
319 end;
320
321 { TCCCompiledCostOption }
322
323 procedure TCCCompiledCostOption._LoadVars;
324 begin
325 (*LoadVarsOPTION*)
326 end;
327
328 function TCCCompiledCostOption._LoadVarsOk: boolean;
329 begin
330 (*LoadVarsOkOPTION*)
331 end;
332
333 procedure TCCCompiledCostOption._Evaluate(const P: TPWSRecord; const Yr : integer; const Data: PSWModelInputData; const PeriodAdjust: integer);
334 begin
335   _Reset(False);
```

```
//  _LoadVars();
  var sz := 0;
  var DRIdx: integer;
  var v:double;
  if P.SystemSize >= ss3301_10k then sz:=1;

  with _Variables do begin
(*EVALUATEOPTION*)
  end;
end;

procedure TCCCompiledCostOption._EvaluateSystem(const P: TPWSRecord;
  const Yr: integer; const Data: PSWModelInputData; const PeriodAdjust: integer);
begin
  _Reset(True);
//  _LoadVars();
  var sz := 0;
  var v:double;
  var DRIdx: integer;
  if P.SystemSize >= ss3301_10k then sz:=1;

  with _Variables do begin
(*EVALUATESYSTEMOPTION*)
  end;
end;

procedure TCCCompiledCostOption._SetEPCosts;
begin
  for var i := low(_EPCost) to high(_EPCost) do _EPCost[i] := true;
(*SetEPCostsOPTION*)
end;

procedure TCCCompiledCostOption._SetNames;
begin
(*SetNamesOPTION*)
end;

procedure TCCCompiledCostOption._SetVarPointer(const name : string; const pd : PDouble);
var s : string;
begin
  s := lowercase(name);
(*_SetVarPointerOPTION*)
end;

procedure TCCCompiledCostOption._SetYears;
begin
  //Size, Year, CostID
(*SetYearsOPTION*)
end;

function TCCCompiledCostOption._DumpVars: string;
var s : string;
begin
  s:='';
(*DumpVarsOPTION*)
  result:=s;
end;

constructor TCCCompiledCostOption.create;
begin
  inherited create;
(*SetStateOPTION*)
end;



end.

```

code\CostModules\SafeWaterCostBase.pas

```
unit SafeWaterCostBase;

```

```
interface

uses Classes, SafeWaterModelData, StrUtils, SafewaterPWSRecords, SafeWaterOccurrence,
     SafeWaterUncertBucket, SafewaterMetrics, SysUtils, SafewaterUtility, SafewaterCalcDist,
     Math, SafewaterWBSCostCurves, SafewaterGlobals;

type

  TCostOutputsEJ = record
    HHCostBlack, HHCostWhite,HHCostHisp,HHCostOther, HhCostAll: double;
  end;
  TCostPopsEJ = record
    Black, White,Hisp,Other, All: double;
  end;

  TSafeWaterCostBase=class(Tobject)
  protected
    EJProps: array of double;  // for free form EJ output
    EJHHC, EJHHC_act : array of array of double;
    EJAddpop, EJAddpop_act: array of double;
    FDebugList: TstringLIst;
    FOptionName: string;
    FActive, FInitialized: boolean;
    FNeededContams: string;
    FYears: integer; // covenience - end - start + 1

    FData: PSWModelData;
    FGlb: TGlobalUncertainty;
    //
    O,_T: TMDRCostOutputs;
    OYearly: TMDRYearlyCostOutputs;
    Inf : TCostInfo;
    //gates for quantile collectors
    PWSEJ_act, EJInd, PWS_Act: double;
    EJO, EJOAct: array of TCostOutputsEJ; //by DR
    EJOPopAct: TCostPopsEJ;
    CurConc : TContamArray;
    CurPop : double;
    NumContams: integer;
    EPNo, EPCostNo: integer; //current EP number
    fAvgFlowA,fAvgFlowB,fDesignFlowA,fDesignFlowB : double;

    {$ifdef DEBUG}
    _d: boolean; //debug output indicator
    {$endif}

    CompLevel: integer;
    FIteration : integer;  //in case modules need to know iteration

    //TODO decide which to promote to TCostOutputs
    FlowThousandGY, NumHouseholds, PWSAnnualRevenue, PWSCapitalCostRate: double;
    TookAction: boolean;  //set to sum up _act variables..
    //This record will be sent to Benefits AddEntryPointBenefits method
    FCostInfo: TCostInfo2Benefit;

    procedure SetupModule; virtual;
    procedure BreakdownModule; virtual;
    procedure ParseConc; virtual;
    procedure SetupYearlyOutputs(Metrics: TCategoryMetrics);
  public
    constructor Create(aData : PSWModelData; GlbUnc: TGlobalUncertainty;  aNeededContams: string; aOptionName: string; aIteration: integer); virtual;
    destructor Destroy; override;

    function Init: boolean;
    function GetMaxMCL: double; virtual;
    procedure SetupUncertainty(Unc: TUncertaintyStudy); virtual;
    procedure SetupOutputs(Metrics: TCategoryMetrics); virtual;

    procedure CapInputConc(const P: TPWSRecord; var Conc: TContamArray; const EPNo: integer; var UpdatePop: double); virtual;
    procedure ForceOutputConc(const P: TPWSRecord; var FinalConc: TContamArray; const EPNo: integer); virtual;
    procedure ResetCosts();
    procedure SetPWSWideEPValues(const P: TPWSRecord); virtual;
    procedure CollectPWSWideEPValues(const P: TPWSRecord); virtual;

```

```
77      procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); virtual;
78      procedure AddEntryPointCostsPostProcess(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); virtual;
79
80      property OptionName: string read FOptionName write FOptionName;
81      property Active: boolean read FActive write FActive;
82      property NeededContams: string read FNeededContams write FNeededContams;
83      property Initialized: boolean read FInitialized;
84      property Output: TMDRCostOutputs read O;
85      property CostInfo: TCostInfo2Benefit read FCostInfo;
86    end;
87
88  implementation
89
90  { TSafeWaterCostBase }
91
92  procedure TSafeWaterCostBase.AddEntryPointCosts(const P: TPWSRecord;
93    const Conc: TContamArray; var FinalConc: TContamArray);
94  begin
95    for var i := 0 to length(CurConc)-1 do begin
96      CurConc[i] := Conc[i];
97      //set final conc to starting conc initially
98      FinalConc[i] := Conc[i];
99    end;
100   _i(Inf.EntryPointCount,1);
101   ParseConc();
102   for var i := 0 to length(CurConc)-1 do begin
103     _i(Inf.SystemMeanConc[i],CurConc[i] / P.EPCount);
104   end;
105   var E := FData.EJProps.GetAllProportionData(P.PWSId);
106   var i: integer;
107   if Assigned(E) then
108     for i:=0 to length(EJProps) - 1 do  EJProps[i] := E.FValues[i]
109   else
110     for i:=0 to length(EJProps) - 1 do  EJProps[i] := 0;
111   inc(EPNo);
112
113   //reset values to accomodate VLS shim
114   CurPop := P.Population / P.EPCount;
115   Inf.AverageFlow := fAvgFlowA * Power(P.Population,fAvgFlowB) * 0.001;
116   var dflow := fDesignFlowA * Power(P.Population,fDesignFlowB) * 0.001;
117   Inf.DesignFlow := Max(2 * Inf.AverageFlow,dflow);
118   //set to EP specific... EPCount is 1 for custom VLS conc...
119   Inf.AverageFlow := Inf.AverageFlow / P.EPCount;
120   Inf.DesignFlow := Inf.DesignFlow / P.EPCount;
121 end;
122
123 procedure TSafeWaterCostBase.AddEntryPointCostsPostProcess(const P: TPWSRecord;
124   const Conc: TContamArray; var FinalConc: TContamArray);
125 begin
126   for var i := 0 to length(CurConc)-1 do begin
127     _i(Inf.SystemMeanConcFinal[i], FinalConc[i] / P.EPCount);
128   end;
129 end;
130
131 procedure TSafeWaterCostBase.BreakdownModule;
132 begin
133
134 end;
135
136 procedure TSafeWaterCostBase.CapInputConc(const P: TPWSRecord;
137   var Conc: TContamArray; const EPNo: integer; var UpdatePop: double);
138 begin
139
140 end;
141
142 procedure TSafeWaterCostBase.ForceOutputConc(const P: TPWSRecord;
143   var FinalConc: TContamArray; const EPNo: integer);
144 begin
145
146 end;
147
148 procedure TSafeWaterCostBase.CollectPWSWideEPValues(const P: TPWSRecord);
149 begin
150   //annualize values
151   for var i  := 0 to fData.Inp.RateCount-1 do begin
```

```
152     var r := FData.Inp.PreCalcAnnRate[i, fYears];
153     var a := FData.Inp.PWSPreCalcAnnRate[P.SourceWater, P.SystemSize, i, fYears];
154
155     O[i].AgencyImpAdmin := O[i].AgencyImpAdmin * r;
156     O[i].AgencyImpSampling := O[i].AgencyImpSampling * r;
157     O[i].AgencyTreatmentAdmin := O[i].AgencyTreatmentAdmin * r;
158
159     O[i].PWSImpAdmin := O[i].PWSImpAdmin * r;
160     O[i].PWSSampling := O[i].PWSSampling * r;
161     O[i].PWSTreatment := O[i].PWSTreatment * r;
162     O[i].PWSTreatmentAdmin := O[i].PWSTreatmentAdmin * r;
163     O[i].CapitalCost := O[i].CapitalCost * r;
164     O[i].OMCost := O[i].OMCost * r;
165
166     O[i].PWSImpAdmin_pws := O[i].PWSImpAdmin_pws * a;
167     O[i].PWSSampling_pws := O[i].PWSSampling_pws * a;
168     O[i].PWSTreatment_pws := O[i].PWSTreatment_pws * a;
169     O[i].PWSTreatmentAdmin_pws := O[i].PWSTreatmentAdmin_pws * a;
170     O[i].CapitalCost_pws := O[i].CapitalCost_pws * a;
171     O[i].OMCost_pws := O[i].OMCost_pws * a;
172
173     O[i].PWSAnnualCosts := O[i].PWSImpAdmin + O[i].PWSSampling + O[i].PWSTreatment + O[i].PWSTreatmentAdmin;
174     O[i].PWSAnnualCosts_pws := O[i].PWSImpAdmin_pws + O[i].PWSSampling_pws + O[i].PWSTreatment_pws + O[i].PWSTreatmentAdmin_pws;
175
176     O[i].AgencyAnnualCosts := O[i].AgencyImpAdmin + O[i].AgencyImpSampling + O[i].AgencyTreatmentAdmin;
177     O[i].TotalAnnualCosts := O[i].PWSAnnualCosts + O[i].AgencyImpAdmin + O[i].AgencyImpSampling + O[i].AgencyTreatmentAdmin;
178
179
180     var v := O[i].PWSAnnualCosts_pws / FlowThousandGY;
181     O[i].HHAnnualCosts_pws := v * FData.Inp.HouseholdConsumption[P.Ownership,P.SystemSize];
182     v := O[i].PWSAnnualCosts / FlowThousandGY;
183     O[i].HHAnnualCosts := v * FData.Inp.HouseholdConsumption[P.Ownership,P.SystemSize];
184     O[i].HHAll:=O[i].HHAnnualCosts_pws * P.Population;
185
186     if TookAction then begin
187       O[i].PWSAnnualCosts_pws_act :=  O[i].PWSAnnualCosts_pws;
188       O[i].HHAnnualCosts_pws_act := O[i].HHAnnualCosts_pws;
189     end;
190
191     if PWSAnnualRevenue>0 then begin
192       var crg := O[i].PWSAnnualCosts/PWSAnnualRevenue;
193       if crg>0.03 then O[i].CRG3 := 1 else O[i].CRG3:=0;
194       if crg>0.01 then O[i].CRG1 := 1 else O[i].CRG1:=0;
195     end;
196
197
198     if P.EJInd=1 then begin
199       EJO[i].HHCostBlack := O[i].HHAnnualCosts_pws * P.TotalRacePop[reBlack];
200       EJO[i].HHCostWhite := O[i].HHAnnualCosts_pws * P.TotalRacePop[reWhite];
201       EJO[i].HHCostOther := O[i].HHAnnualCosts_pws * P.TotalRacePop[reOther];
202       EJO[i].HHCostHisp := O[i].HHAnnualCosts_pws * P.TotalRacePop[reHispanic];
203       EJO[i].HHCostAll := O[i].HHAnnualCosts_pws * P.Population;
204       for var j := 1 to length(EJHHC[0]) -1 do begin
205         EJHHC[i,j] := O[i].HHAnnualCosts_pws * P.Population * EJProps[j];
206         EJAddPop[j] := P.Population * EJProps[j];
207       end;
208
209       if TookAction then begin
210         EJOAct[i].HHCostBlack := O[i].HHAnnualCosts_pws * P.TotalRacePop[reBlack];
211         EJOAct[i].HHCostWhite := O[i].HHAnnualCosts_pws * P.TotalRacePop[reWhite];
212         EJOAct[i].HHCostOther := O[i].HHAnnualCosts_pws * P.TotalRacePop[reOther];
213         EJOAct[i].HHCostHisp := O[i].HHAnnualCosts_pws * P.TotalRacePop[reHispanic];
214         EJOAct[i].HHCostAll := O[i].HHAnnualCosts_pws * P.Population;
215         EJOPopAct.Black := P.TotalRacePop[reBlack];
216         EJOPopAct.White := P.TotalRacePop[reWhite];
217         EJOPopAct.Other := P.TotalRacePop[reOther];
218         EJOPopAct.Hisp := P.TotalRacePop[reHispanic];
219         EJOPopAct.All := P.Population;
220         for var j := 1 to length(EJHHC[0]) -1 do begin
221           EJHHC_act[i,j] := O[i].HHAnnualCosts_pws * P.Population * EJProps[j];
222           EJAddPop_act[j] := P.Population * EJProps[j];
223         end;
224
225       end;
```

```
    end;

  end;

  PWS_Act := 0;
  if TookAction then PWS_Act := 1;
  PWSEJ_act:=0;
  EJInd := P.EJInd;
  if TookAction and (EJInd = 1) then PWSEJ_Act := 1;
  Inf.PWSCount := 1;

  {$ifdef DEBUG}
  (*
  if _d then begin
    fData.DebugLog.L('costs','FINAL COSTS');
    fData.DebugLog.L('costs', O.DebugLine);
    fData.DebugLog.L('pws', Format('%d, %s, %g',[FIteration, P.PWSId, O.PWSTreatment]));
  end;
  *)
  {$endif}
end;

constructor TSafeWaterCostBase.Create(aData: PSWModelData; GlbUnc: TGlobalUncertainty; aNeededContams: string; aOptionName: string; aIteration:
integer);
begin
  FData := aData;
  FGlb := GlbUnc;
  FOptionName := aOptionName;
  FActive := False;
  FNeededContams := uppercase(aNeededContams);
  FInitialized := False;
  CompLevel := wbsCompMid;
  FIteration := aIteration;
  FDebugList := TStringList.Create;
  if FData.Cfg.DebugOut then begin
    FDebugList.CommaText := FData.Cfg.DebugList;
    FDebugList.Sorted := True;
  end;
  setlength(_T,fData.Inp.RateCount);
  setlength(O,fData.Inp.RateCount);
  FYears := FData.Cfg.EndYear - FData.Cfg.StartYear + 1;
  setlength(OYearly,FYears);
  for var i := 0 to High(OYearly) do
    setlength(OYearly[i],fData.Inp.RateCount);
end;

destructor TSafeWaterCostBase.Destroy;
begin
  BreakdownModule;
  FDebugList.Free;
  inherited;
end;

function TSafeWaterCostBase.GetMaxMCL: double;
begin
  Result := 1e30;
end;

function TSafeWaterCostBase.Init: boolean;
begin
  result := true;
  NumContams := FData.Occ.ContamList.Count;
  setlength(CurConc,NumContams);
  var S := SplitString(FNeededContams,',');
  result := FData.Occ.DataAvailable(FNeededContams);
  if Result then
    SetupModule
  else
    if FIteration<0 then
      FData.ErrLog.L(FOptionName +' missing needed contaminants');
  FInitialized := result;
end;

procedure TSafeWaterCostBase.ParseConc;
begin
```

```
300   //do nothing in the abstract - convenience function for direct access to conc data
301 end;
302
303 procedure TSafeWaterCostBase.ResetCosts;
304 begin
305   EJOPopAct := Default(TCostPopsEJ);
306   Inf := Default(TCostInfo);
307   for var y := 0 to high(OYearly) do
308     for var i := 0 to fData.Inp.RateCount-1 do begin
309       OYearly[y,i] := Default(TCostOutputs);
310     end;
311
312   for var i := 0 to fData.Inp.RateCount-1 do begin
313     _T[i] := Default(TCostOutputs);
314     O[i] := Default(TCostOutputs);
315     EJO[i] := Default(TCostOutputsEJ);
316     EJOAct[i] := Default(TCostOutputsEJ);
317     for var j := 1 to length(EJHHC[0]) - 1 do begin
318       EJHHC[i,j] := 0;
319       EJHHC_act[i,j] := 0;
320     end;
321   end;
322   FCostInfo := Default(TCostInfo2Benefit);
323   for var i := 1 to length(EJAddPop) -1 do begin
324     EJAddPop[i] := 0;
325     EJAddPop_act[i] := 0;
326   end;
327 end;
328
329 procedure TSafeWaterCostBase.SetPWSWideEPValues(const P: TPWSRecord);
330 begin
331   EPNo:=0;
332   EPCostNo:=0;
333   TookAction := False;
334   CurPop := P.Population / P.EPCount;
335   if (P.Ownership=oPublic) and (P.SourceWater=swGroundwater) then begin
336     fAvgFlowA :=  FGlb.FlowParameters.PubGroundA;
337     fAvgFlowB := FGlb.FlowParameters.PubGroundB;
338     fDesignFlowA := FGlb.FlowParameters.D_PubGroundA;
339     fDesignFlowB := FGlb.FlowParameters.D_PubGroundB;
340   end else
341   if (P.Ownership=oPrivate) and (P.SourceWater=swGroundwater) then begin
342     fAvgFlowA := FGlb.FlowParameters.PrivGroundA;
343     fAvgFlowB := FGlb.FlowParameters.PrivGroundB;
344     fDesignFlowA := FGlb.FlowParameters.D_PrivGroundA;
345     fDesignFlowB := FGlb.FlowParameters.D_PrivGroundB;
346   end else
347   if (P.Ownership=oPublic) and (P.SourceWater=swSurfacewater) then begin
348     fAvgFlowA := FGlb.FlowParameters.PubSurfA;
349     fAvgFlowB := FGlb.FlowParameters.PubSurfB;
350     fDesignFlowA := FGlb.FlowParameters.D_PubSurfA;
351     fDesignFlowB := FGlb.FlowParameters.D_PubSurfB;
352   end else
353   if (P.Ownership=oPrivate) and (P.SourceWater=swSurfacewater) then begin
354     fAvgFlowA := FGlb.FlowParameters.PrivSurfA;
355     fAvgFlowB := FGlb.FlowParameters.PrivSurfB;
356     fDesignFlowA := FGlb.FlowParameters.D_PrivSurfA;
357     fDesignFlowB := FGlb.FlowParameters.D_PrivSurfB;
358   end;
359
360   Inf.AverageFlow := fAvgFlowA * Power(P.Population,fAvgFlowB) * 0.001;
361   var dflow := fDesignFlowA * Power(P.Population,fDesignFlowB) * 0.001;
362   Inf.DesignFlow := Max(2 * Inf.AverageFlow,dflow);
363
364   NumHouseholds:=CurPop / 2.9 ;
365   FlowThousandGY :=  (Inf.AverageFlow * 365000) ;
366   PWSAnnualRevenue := FData.Inp.PWSRevenue[P.Ownership,P.SystemSize] * (Inf.AverageFlow * 365000);
367   PWSCapitalCostRate := FData.Inp.PWSCapitalCost[P.Ownership,P.SystemSize] / 100;
368
369   //set to EP specific...
370   Inf.AverageFlow := Inf.AverageFlow / P.EPCount;
371   Inf.DesignFlow := Inf.DesignFlow / P.EPCount;
372
373   //set cost level based on static random generated at input
374   CompLevel := wbsCompMid;
```

```
375   if fData.Cfg.Uncertainty then begin
376     var cl := (P.rnCLS + FGlb.CompLevelShift) - 1;
377     if cl <= 0.20 then
378       CompLevel := wbsCompLow
379     else
380     if cl <= 0.80 then
381       CompLevel := wbsCompMid
382     else
383       CompLevel := wbsCompHigh;
384   end;
385
386   //debugging
387   {$ifdef DEBUG}
388   if FData.Cfg.DebugOut then
389     _d := FDebugList.IndexOf(P.PWSId)>=0
390   else
391    _d := false;
392   {$endif}
393
394   {$ifdef DEBUG}
395   if _d then begin
396     with FData.DebugLog do begin
397       LD('costs','', '*********************************');
398       L('costs',P.OutData);
399       LD('costs','Iteration', FIteration);
400       LD('costs','Pop', P.Population);
401       LD('costs','EP Pop', CurPop);
402       LD('costs','EP Average Flow', Inf.AverageFlow);
403       LD('costs','EP Design Flow', Inf.DesignFlow);
404       LD('costs','Revenue', PWSAnnualRevenue);
405       LD('costs','Capital Cost', PWSCapitalCostRate);
406       LD('costs','Comp Level', CompLevel);
407     end;
408   end;
409   {$endif}
410 end;
411
412 procedure TSafeWaterCostBase.SetupModule;
413 begin
414   //do nothing in the abstract - convenience function for direct access to conc data
415 end;
416
417 procedure TSafeWaterCostBase.SetupYearlyOutputs(Metrics: TCategoryMetrics);
418 begin
419   if not FData.Cfg.YearlyOut then exit;
420   var mc := mcSumOnly;
421   var mt := mtCostYearly;
422   var tg: string;
423   for var Yr := 0 to High(OYearly) do begin
424     var yt := inttostr(yr + FData.Cfg.StartYear);
425     var ayt := 'YYYY'+yt;
426     var dyt := ' Y('+yt+')';
427     var mix := fData.Inp.RateCount-1;
428     if FData.Cfg.ZeroDiscYearly then mix := 0;
429     for var i := 0 to mix do begin
430       tg := floattostr(fData.Inp.Rates[i]);
431
432       Metrics.AddMetric(fOptionName+'.AgencyImpAdmin'+ayt,
433                         'Agency Administration'+dyt, @oYearly[yr,i].AgencyImpAdmin, mt, mc, False, tg);
434       Metrics.AddMetric(fOptionName+'.AgencyImpSampling'+ayt,
435                         'Agency Sampling'+dyt, @oYearly[yr,i].AgencyImpSampling, mt, mc, False, tg);
436       Metrics.AddMetric(fOptionName+'.AgencyTreatmentAdmin'+ayt,
437                         'Agency Treatment Administration'+dyt, @oYearly[yr,i].AgencyTreatmentAdmin, mt, mc, False, tg);
438
439
440       Metrics.AddMetric(fOptionName+'.CapitalCost'+ayt,
441                         'Annual Capital Cost'+dyt, @OYearly[yr,i].CapitalCost, mt, mc, False, tg);
442       Metrics.AddMetric(fOptionName+'.CapitalCost_pws'+ayt,
443                         'Annual Capital Cost_pws'+dyt, @OYearly[yr,i].CapitalCost_pws, mt, mc, False, tg);
444       Metrics.AddMetric(fOptionName+'.OMCost'+ayt,
445                         'Annual O&M Cost'+dyt, @OYearly[yr,i].OMCost, mt, mc, False, tg);
446       Metrics.AddMetric(fOptionName+'.OMCost_pws'+ayt,
447                         'Annual O&M Cost_pws'+dyt, @OYearly[yr,i].OMCost_pws, mt, mc, False, tg);
448
```

```
449       Metrics.AddMetric(fOptionName+'.PWSImpAdmin'+ayt,
450                           'PWS Rule Implementation and Administration Costs'+dyt, @oYearly[yr,i].PWSImpAdmin, mt, mc, False, tg);
451       Metrics.AddMetric(fOptionName+'.PWSImpAdmin_pws'+ayt,
452                           'PWS Rule Implementation and Administration Costs_pws'+dyt, @oYearly[yr,i].PWSImpAdmin_pws, mt, mc, False, tg);
453       Metrics.AddMetric(fOptionName+'.PWSTreatmentAdmin'+ayt,
454                           'PWS Treatment Administration Costs'+dyt, @oYearly[yr,i].PWSTreatmentAdmin, mt, mc, False, tg);
455       Metrics.AddMetric(fOptionName+'.PWSTreatmentAdmin_pws'+ayt,
456                           'PWS Treatment Administration Costs_pws'+dyt, @oYearly[yr,i].PWSTreatmentAdmin_pws, mt, mc, False, tg);
457
458       Metrics.AddMetric(fOptionName+'.PWSSampling'+ayt,
459                           'PWS Sampling Costs'+dyt, @oYearly[yr,i].PWSSampling, mt, mc, False, tg);
460       Metrics.AddMetric(fOptionName+'.PWSSampling_pws'+ayt,
461                           'PWS Sampling Costs_pws'+dyt, @oYearly[yr,i].PWSSampling_pws, mt, mc, False, tg);
462     end;
463   end;
464 end;
465
466 procedure TSafeWaterCostBase.SetupOutputs(Metrics: TCategoryMetrics);
467 begin
468   SetupYearlyOutputs(Metrics);
469   var i,j: integer;
470   var tg: string;
471   //do system mean concs...
472   for i :=0 to FData.Occ.ContamList.Count - 1  do begin
473     Metrics.AddMetric(fOptionName+'.'+FData.Occ.ContamList[i] + '_SysMeanConc',
474        FData.Occ.ContamList[i] + ' System Mean Concentration',
475        @Inf.SystemMeanConc[i], mtInfo, mcMeanOnly, False);
476     Metrics.AddMetric(fOptionName+'.'+FData.Occ.ContamList[i] + '_SysMeanConcFinal',
477        FData.Occ.ContamList[i] + ' System Mean Final Concentration',
478        @Inf.SystemMeanConcFinal[i], mtInfo, mcMeanOnly, False);
479
480   end;
481
482   Metrics.AddMetric(fOptionName+'.EntryPointCount',
483                     'Entry Points Count', @Inf.EntryPointCount, mtInfo, mcMeanSum, False);
484   Metrics.AddMetric(fOptionName+'.PWSCount',
485                     'Total Number of PWSs', @Inf.PWSCount, mtInfo, mcSumOnly, False);
486   Metrics.AddMetric(fOptionName+'.DesignFlowEP',
487                     'Design Flow By Entry Point', @Inf.DesignFlow, mtInfo, mcMeanOnly, False);
488   Metrics.AddMetric(fOptionName+'.AverageFlowEP',
489                     'Average Flow By Entry Point', @Inf.AverageFlow, mtInfo, mcMeanOnly, False);
490
491   for i := 0 to fData.Inp.RateCount-1 do begin
492     tg := floattostr(fData.Inp.Rates[i]);
493     Metrics.AddMetric(fOptionName+'.TotalAnnualCosts',
494                       'Total Annual Rule Costs', @o[i].TotalAnnualCosts, mtCost, mcMeanSum, False, tg);
495
496     Metrics.AddMetric(fOptionName+'.CapitalCost',
497                       'Annualized Capital Cost', @o[i].CapitalCost, mtCost, mcMeanSum, False, tg);
498     Metrics.AddMetric(fOptionName+'.CapitalCost_pws',
499                       'Annualized Capital Cost_pws', @o[i].CapitalCost_pws, mtCost, mcMeanSum, False, tg);
500     Metrics.AddMetric(fOptionName+'.OMCost',
501                       'Annualized O&M Cost', @o[i].OMCost, mtCost, mcMeanSum, False, tg);
502     Metrics.AddMetric(fOptionName+'.OMCost_pws',
503                       'Annualized O&M Cost_pws', @o[i].OMCost_pws, mtCost, mcMeanSum, False, tg);
504
505
506     Metrics.AddMetric(fOptionName+'.AgencyAnnualCosts',
507                       'Agency Annual Rule Costs', @o[i].AgencyAnnualCosts, mtCost, mcMeanSum, False, tg);
508     Metrics.AddMetric(fOptionName+'.PWSAnnualCosts',
509                       'Total Annual PWS Rule Costs', @o[i].PWSAnnualCosts, mtCost, mcAll, False, tg);
510     Metrics.AddMetric(fOptionName+'.PWSAnnualCosts_pws',
511                       'Total Annual PWS Rule Costs_pws', @o[i].PWSAnnualCosts_pws, mtCost, mcAll, False, tg);
512     Metrics.AddMetric(fOptionName+'.PWSAnnualCosts_pws_act',
513                       'Total Annual PWS Rule Costs_pws_act', @o[i].PWSAnnualCosts_pws_act, mtCost, mcAll, True, tg);
514
515     Metrics.AddMetric(fOptionName+'.AgencyImpAdmin',
516                       'Agency Administration', @o[i].AgencyImpAdmin, mtCost, mcMeanSum, False, tg);
517     Metrics.AddMetric(fOptionName+'.AgencyImpSampling',
518                       'Agency Sampling', @o[i].AgencyImpSampling, mtCost, mcMeanSum, False, tg);
519     Metrics.AddMetric(fOptionName+'.AgencyTreatmentAdmin',
520                       'Agency Treatment Administration', @o[i].AgencyTreatmentAdmin, mtCost, mcMeanSum, False, tg);
521
522     Metrics.AddMetric(fOptionName+'.PWSImpAdmin',
523                       'PWS Rule Implementation and Administration Costs', @o[i].PWSImpAdmin, mtCost, mcMeanSum, False, tg);
```

```
524     Metrics.AddMetric(fOptionName+'.PWSImpAdmin_pws',
525                       'PWS Rule Implementation and Administration Costs_pws', @o[i].PWSImpAdmin_pws, mtCost, mcMeanSum, False, tg);
526     Metrics.AddMetric(fOptionName+'.PWSTreatmentAdmin',
527                       'PWS Treatment Administration Costs', @o[i].PWSTreatmentAdmin, mtCost, mcMeanSum, False, tg);
528     Metrics.AddMetric(fOptionName+'.PWSTreatmentAdmin_pws',
529                       'PWS Treatment Administration Costs_pws', @o[i].PWSTreatmentAdmin_pws, mtCost, mcMeanSum, False, tg);
530
531     Metrics.AddMetric(fOptionName+'.PWSSampling',
532                       'PWS Sampling Costs', @o[i].PWSSampling, mtCost, mcMeanSum, False, tg);
533     Metrics.AddMetric(fOptionName+'.PWSSampling_pws',
534                       'PWS Sampling Costs_pws', @o[i].PWSSampling_pws, mtCost, mcMeanSum, False, tg);
535
536     Metrics.AddMetric(fOptionName+'.PWSTreatment',
537                       'PWS Treatment Costs', @o[i].PWSTreatment, mtCost, mcMeanSum, False, tg);
538     Metrics.AddMetric(fOptionName+'.PWSTreatment_pws',
539                       'PWS Treatment Costs_pws', @o[i].PWSTreatment_pws, mtCost, mcMeanSum, False, tg);
540
541     Metrics.AddMetric(fOptionName+'.HHAnnualCosts',
542                       'Total Annual Household Costs', @o[i].HHAnnualCosts, mtCost, mcAll, False, tg);
543     Metrics.AddMetric(fOptionName+'.HHAnnualCosts_pws',
544                       'Total Annual Household Costs_pws', @o[i].HHAnnualCosts_pws, mtCost, mcAll, False, tg);
545     Metrics.AddMetric(fOptionName+'.HHAnnualCosts_pws_act',
546                       'Total Annual Household Costs_pws_act', @o[i].HHAnnualCosts_pws_act, mtCost, mcAll, True, tg);
547
548     Metrics.AddMetric(fOptionName+'.CRG1',
549                       'Cost Revenue Ratio Greater Than 1%', @o[i].CRG1, mtCost, mcMeanSum, False, tg);
550     Metrics.AddMetric(fOptionName+'.CRG3',
551                       'Cost Revenue Ratio Greater Than 3%', @o[i].CRG3, mtCost, mcMeanSum, False, tg);
552
553 Metrics.AddMetric(fOptionName+'.HHCAll',
554 'HHC*Pop', @o[i].HHAll, mtInfo, mcSumOnly, False, tg);
555
556   end;
557
558   //TODO EJ Metrics - should the be optional?
559   setlength(EJHHC,fData.Inp.RateCount,length(FData.EJProps.Names));
560   setlength(EJHHC_act,fData.Inp.RateCount,length(FData.EJProps.Names));
561   setlength(EJAddPop,length(FData.EJProps.Names));
562   setlength(EJAddPop_act,length(FData.EJProps.Names));
563   setlength(EJProps,length(FData.EJProps.Names));
564   if FData.Cfg.PFASEJOutput then begin
565     for i:=1 to length(FData.EJProps.Names)-1 do begin
566       var s := FData.EJProps.Names[i];
567       Metrics.AddMetric(fOptionName+'.TotalEPPop'+s,
568                       'Total '+s+' Pop All Entry Points', @EJAddPop[i], mtEJOutput, mcMeanSum, True, '', @EJInd);
569       Metrics.AddMetric(fOptionName+'.TotalEPPop'+s+'_act',
570                       'Total '+s+' Pop All Entry Points_act', @EJAddPop_act[i], mtEJOutput, mcMeanSum, True, '', @EJInd);
571     end;
572
573     for i := 0 to fData.Inp.RateCount-1 do begin
574       tg := floattostr(fData.Inp.Rates[i]);
575       for j:=1 to length(FData.EJProps.Names)-1 do begin
576         var s := FData.EJProps.Names[j];
577         Metrics.AddMetric(fOptionName+'.HHAnnualCosts'+s,
578                           'Total Annual Household Costs * '+s+' Pop', @EJHHC[i,j], mtEJOutput, mcMeanSum, False, tg, @EJInd);
579         Metrics.AddMetric(fOptionName+'.HHAnnualCosts'+s+'_act',
580                           'Total Annual Household Costs_act * '+s+' Pop', @EJHHC_act[i,j], mtEJOutput, mcMeanSum, False, tg, @EJInd);
581       end;
582     end;
583   end;
584
585
586   setlength(EJO,fData.Inp.RateCount);
587   setlength(EJOAct,fData.Inp.RateCount);
588   if FData.Cfg.PFASEJOutput then begin
589     for i := 0 to fData.Inp.RateCount-1 do begin
590       tg := floattostr(fData.Inp.Rates[i]);
591       Metrics.AddMetric(fOptionName+'.HHAnnualCostsBlack',
592                         'Total Annual Household Costs * Black Pop', @EJO[i].HHCostBlack, mtEJOutput, mcMeanSum, False, tg, @EJInd);
593       Metrics.AddMetric(fOptionName+'.HHAnnualCostsWhite',
594                         'Total Annual Household Costs * White Pop', @EJO[i].HHCostWhite, mtEJOutput, mcMeanSum, False, tg, @EJInd);
595       Metrics.AddMetric(fOptionName+'.HHAnnualCostsOther',
596                         'Total Annual Household Costs * Other Pop', @EJO[i].HHCostOther, mtEJOutput, mcMeanSum, False, tg, @EJInd);
597       Metrics.AddMetric(fOptionName+'.HHAnnualCostsHisp',
```

```
                        'Total Annual Household Costs * Hisp Pop', @EJO[i].HHCostHisp, mtEJOutput, mcMeanSum, False, tg, @EJInd);
      Metrics.AddMetric(fOptionName+'.HHAnnualCostsAll',
                        'Total Annual Household Costs * All Pop', @EJO[i].HHCostAll, mtEJOutput, mcMeanSum, False, tg, @EJInd);

      Metrics.AddMetric(fOptionName+'.HHAnnualCostsBlack_act',
                        'Total Annual Household Costs_act * Black Pop', @EJOAct[i].HHCostBlack, mtEJOutput, mcMeanSum, True, tg, @EJInd);
      Metrics.AddMetric(fOptionName+'.HHAnnualCostsWhite_act',
                        'Total Annual Household Costs_act * White Pop', @EJOAct[i].HHCostWhite, mtEJOutput, mcMeanSum, True, tg, @EJInd);
      Metrics.AddMetric(fOptionName+'.HHAnnualCostsOther_act',
                        'Total Annual Household Costs_act * Other Pop', @EJOAct[i].HHCostOther, mtEJOutput, mcMeanSum, True, tg, @EJInd);
      Metrics.AddMetric(fOptionName+'.HHAnnualCostsHisp_act',
                        'Total Annual Household Costs_act * Hisp Pop', @EJOAct[i].HHCostHisp, mtEJOutput, mcMeanSum, True, tg, @EJInd);
      Metrics.AddMetric(fOptionName+'.HHAnnualCostsAll_act',
                        'Total Annual Household Costs_act * All Pop', @EJOAct[i].HHCostAll, mtEJOutput, mcMeanSum, True, tg, @EJInd);
    end;
  end;

  if FData.Cfg.PFASEJOutput then begin
    Metrics.AddMetric(fOptionName+'.TotalEPPop_act',
                    'Total Pop All Entry Points_act', @EJOPopAct.All, mtEJOutput, mcMeanSum, True, '', @EJInd);
    Metrics.AddMetric(fOptionName+'.TotalEPPopBlack_act',
                      'Total Black Pop All Entry Points_act', @EJOPopAct.Black, mtEJOutput, mcMeanSum, True, '', @EJInd);
    Metrics.AddMetric(fOptionName+'.TotalEPPopWhite_act',
                      'Total White Pop All Entry Points_act', @EJOPOpAct.White, mtEJOutput, mcMeanSum, True, '', @EJInd);
    Metrics.AddMetric(fOptionName+'.TotalEPPopHisp_act',
                      'Total Hispanic Pop All Entry Points_act', @EJOPopAct.Hisp, mtEJOutput, mcMeanSum, True, '', @EJInd);
    Metrics.AddMetric(fOptionName+'.TotalEPPopOther_act',
                      'Total Other Pop All Entry Points_act', @EJOPopAct.Other, mtEJOutput, mcMeanSum, True, '', @EJInd);
    Metrics.AddMetric(fOptionName+'.EJPWSCount_act',
                      'PWS with EJ Data availaible_act', @PWSEJ_act, mtInfo, mcSumOnly, False, '', @EJInd);
  end;


  Metrics.AddMetric(fOptionName+'.SystemsTreating',
                    'Count of systems treating', @Inf.SystemIsTreating, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EPsTreating',
                    'Count of EPs treating', @Inf.EntryPointIsTreating, mtInfo, mcSumOnly, False);

  Metrics.AddMetric(fOptionName+'.PWSPopTreating',
                    'PWS Pop Treating', @Inf.PWSPopTreating, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EPPopTreating',
                    'Entry Point Pop Treating', @Inf.EntryPointPopTreating, mtInfo, mcSumOnly, False);

end;

procedure TSafeWaterCostBase.SetupUncertainty(Unc: TUncertaintyStudy);
begin


end;

end.
```

code\CostModules\SafeWaterCostModules.pas

```
unit SafeWaterCostModules;

interface

uses Classes, Types, SafeWaterCostBase, Generics.Collections, StrUtils, SysUtils,
     SafeWaterUncertBucket, SafewaterMetrics, SafewaterPWSRecords, SafeWaterOccurrence,
     SafewaterModelData,
     // add all available cost modules
     cmPFASOption1, cmPFASOption3, cmPFASOption4, cmPFASOption2s, cmPFASOption2x,
     cmPFASOptionMCL1, cmPFASOccRun;

type
   TSafeWaterCostModulesList = TObjectList<TSafeWaterCostBase>;

```

```
15    TSafeWaterCostModules = class(TSafeWaterCostModulesList)
16   private
17     FActiveCount: integer;
18     FIteration: integer;
19   published
20   //public
21     constructor create(aData: PSWModelData; GlbUnc: TGlobalUncertainty; aIteration: integer);
22
23     procedure SetActive(ActiveList: TStringDynArray);
24     procedure SetupUncertainty(Unc: TUncertaintyStudy);
25     procedure SetupOutputs(Metrics: TCategoryMetrics);
26     function InitOk : boolean;
27     function GetMaxMCL: double;
28
29     procedure SetPWSWideEPValues(const P: TPWSRecord);
30     procedure CollectPWSWideEPValues(const P: TPWSRecord);
31     procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray);
32     procedure AddEntryPointCostsPostProcess(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray);
33     procedure ResetCosts;
34     property ActiveCount: integer read FActiveCount write FActiveCount;
35   end;
36 implementation
37
38 { TSafeWaterCostModules }
39
40 procedure TSafeWaterCostModules.AddEntryPointCosts(const P: TPWSRecord;
41   const Conc: TContamArray; var FinalConc: TContamArray);
42 begin
43   for var T in self do begin
44     if T.Active then
45       T.AddEntryPointCosts(P,Conc,FinalConc);
46   end;
47 end;
48
49 procedure TSafeWaterCostModules.AddEntryPointCostsPostProcess(
50   const P: TPWSRecord; const Conc: TContamArray; var FinalConc: TContamArray);
51 begin
52   for var T in self do begin
53     if T.Active then
54       T.AddEntryPointCostsPostProcess(P, Conc, FinalConc);
55   end;
56 end;
57
58 procedure TSafeWaterCostModules.CollectPWSWideEPValues(const P: TPWSRecord);
59 begin
60   for var T in self do begin
61     if T.Active then
62       T.CollectPWSWideEPValues(P);
63   end;
64 end;
65
66 constructor TSafeWaterCostModules.create(aData: PSWModelData; GlbUnc: TGlobalUncertainty; aIteration: integer);
67 begin
68   inherited create(true);
69   FIteration := aIteration;
70   Add(TcmPFASOption1.Create(aData, GlbUnc, 'PFOS,PFOA','PFASOption1a', fIteration));
71   Add(TcmPFASOption1.Create(aData, GlbUnc, 'PFOS,PFOA','PFASOption1b', fIteration));
72   Add(TcmPFASOption1.Create(aData, GlbUnc, 'PFOS,PFOA','PFASOption1c', fIteration));
73   Add(TcmPFASOption1.Create(aData, GlbUnc, 'PFOS,PFOA','PFASOption1d', fIteration));
74   Add(TcmPFASOption1.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS','PFASOption2', fIteration));
75   Add(TcmPFASOption3.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS','PFASOption3', fIteration));
76   Add(TcmPFASOption4.Create(aData, GlbUnc, 'PFOS,PFHPA,PFOA,PFHXS','PFASOption4', fIteration));
77   Add(TcmPFASOption2s.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASOption2s', fIteration));
78   Add(TcmPFASOption2x.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASOption2x', fIteration));
79   Add(TcmPFASOption2x.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASOption2x-PFNA', fIteration));
80   Add(TcmPFASOption2x.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASOption2x-HFPODA', fIteration));
81   Add(TcmPFASOption2x.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASOption2x-PFHXS', fIteration));
82   Add(TcmPFASOptionMCL1.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASMCL-PFNA', fIteration));
83   Add(TcmPFASOptionMCL1.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASMCL-HFPODA', fIteration));
84   Add(TcmPFASOptionMCL1.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASMCL-PFHXS', fIteration));
85   Add(TcmPFASOccRun.Create(aData, GlbUnc, 'PFOS,PFOA,PFHXS,PFBS,PFNA,HFPODA','PFASOccRun', fIteration));
86
87 end;
88
89 function TSafeWaterCostModules.GetMaxMCL: double;
```

```
90  begin
91    Result := 0;
92    var tmp: double;
93    for var T in self do begin
94      if T.Active then begin
95        tmp := T.GetMaxMCL;
96        if tmp>Result then Result:=tmp;
97      end;
98    end;
99  end;
100
101 function TSafeWaterCostModules.InitOK : boolean;
102 begin
103   Result:=True;
104   for var T in self do begin
105     if not T.Active then continue;
106     if not T.Init then begin
107       Result:=False;
108       T.Active := False;
109       dec(fActiveCount);
110     end;
111   end;
112 end;
113
114 procedure TSafeWaterCostModules.ResetCosts;
115 begin
116   for var T in self do begin
117     if T.Active then
118       T.ResetCosts;
119   end;
120 end;
121
122 procedure TSafeWaterCostModules.SetActive(ActiveList: TStringDynArray);
123 begin
124   ActiveCount:=0;
125   var L := TSTringList.Create;
126   for var s in ActiveList  do L.Add(s);
127   for var T in self do begin
128     if L.IndexOf(T.OptionName)>=0 then begin
129       T.Active := True;
130       inc(fActiveCount);
131     end;
132   end;
133   L.Free;
134 end;
135
136 procedure TSafeWaterCostModules.SetPWSWideEPValues(const P: TPWSRecord);
137 begin
138   for var T in self do begin
139     if T.Active then
140       T.SetPWSWideEPValues(P);
141   end;
142 end;
143
144 procedure TSafeWaterCostModules.SetupOutputs(Metrics: TCategoryMetrics);
145 begin
146   for var T in self do begin
147     if T.Active then
148       T.SetupOutputs(Metrics);
149   end;
150 end;
151
152 procedure TSafeWaterCostModules.SetupUncertainty;
153 begin
154   for var T in self do begin
155     if T.Active then
156       T.SetupUncertainty(Unc);
157   end;
158 end;
159
160 end.
161
```

code\CostModules\SafeWaterTFObject.pas

```
unit SafeWaterTFObject;

interface

uses SysUtils, Classes, StrUtils, SafeWaterUtility, SafeWaterGlobals;

const
  ttNone=-1;
  ttGAC=0;    ttIX=1;    ttCRO=2;
  ttPOU=3;    ttNTI=4;   ttNTNW=5;

  stNone = 0; stPressure=1; stGravity=2;
  tocLow=0;   tocHigh=1;

  dfLow=0;    dfMed=1;   dfHigh=2;

type
  TTechnologies = array[ttGAC..ttNTNW] of double;
  TTechnologiesViable = array[ttGAC..ttNTNW] of boolean;

  TAllTechnologies = array[dfLow..dfHigh,tocLow..tocHigh] of TTechnologies;

  TAllTechnologiesObj = class(Tobject)
    Active: boolean;
    tF : TAllTechnologies;
    function TextToTech(s: string) : integer;
    function TechToText(s: integer) : string;
    procedure LoadTech(aFilename: string);
    procedure GetTechForecast(const vdf,vtoc: double; var res: TTechnologies);
  end;

implementation

{ TcmAllTechnologiesObj }

procedure TAllTechnologiesObj.GetTechForecast(const vdf,vtoc: double; var res: TTechnologies);
begin
  var itoc, idf: integer;
  if vtoc <= 1.5 then
    itoc := tocLow
  else
    itoc := tocHigh;

  if vdf < 1 then
    idf := dfLow
  else
  if vdf < 10 then
    idf := dfMed
  else
    idf := dfHigh;

  move(tf[idf,itoc,0],Res[0], length(Res) * sizeof(Res[0]));
end;

procedure TAllTechnologiesObj.LoadTech(aFilename: string);
begin
  var T := TStreamReader.Create(aFilename);
  var s := T.ReadLine;
  Active := True;
  while not T.EndOfStream do begin
    try
      s := T.ReadLine;
      var f := SplitString(s,',');
      var tt := TextToTech(f[0]);
      tF[strtoint(f[2]),strtoint(f[1]),tt] := strtofloat(f[3]);
    except
      //bad file format
      Active := False;
      break;
    end;
  end;
  T.Free;
end;
```

```
function TAllTechnologiesObj.TechToText(s: integer): string;
begin
  if s=ttGAC then Result := 'GAC' else
  if s=ttIX then Result := 'IX' else
  if s=ttCRO then Result := 'CRO' else
  if s=ttPOU then Result := 'POU' else
  if s=ttNTI then Result := 'NTI' else
  if s=ttNTNW then Result := 'NNTW' else
    result := '?';
end;

function TAllTechnologiesObj.TextToTech(s: string): integer;
begin
  var t := uppercase(s);
  if t = 'GAC' then result := ttGAC else
  if t = 'IX' then result := ttIX else
  if t = 'CRO' then result := ttCRO else
  if t = 'POU' then result := ttPOU else
  if t = 'NTI' then result := ttNTI else
  if t = 'NTNW' then result := ttNTNW else
    result := -1;
end;

end.

```

code\CostModules\SafeWaterWBSCostCurves.pas

```
unit SafeWaterWBSCostCurves;

interface

uses SysUtils, Classes, Types, StrUtils, SafeWaterUtility, SafeWaterGlobals, SafeWaterTFObject,
     Generics.Collections, Math ;

const
  wbsCompLow = 0;     wbsCompMid = 1;      wbsCompHigh = 2;

type
  TCostCurve = record
    cc, oc: array[1..10] of double; //total capital, annual o&m
    UsefulLife: double;
    PartCC,PartOC: integer;
    procedure values(const Q,AF: double; var Capital: double; var OM: double);
  end;

  TCostCurveDict = TDictionary<integer,TCostCurve>;

  TWBSCostCurves = class(Tobject)
  protected
    Curves: TCostCurveDict;
    FileErrors : integer;
    FInfo: TStringLIst;
    FErrLog: TSimpleLog;
    function MakeID(Source,Tech,SubTech,SizeCat,CompLevel,BedVolume: integer): integer;
    procedure FromID(const ID: integer; var Source,Tech,SubTech,SizeCat,CompLevel,BedVolume: integer);
    procedure ReadFile(aFIlename: string);
  public

    constructor Create(aLog : TSimplelog);
    destructor Destroy; override;

    procedure ReadFiles(aFiles : TStringDynArray);
    procedure GetCosts(const C0Total: double; const Source,Tech,SubTech,CompLevel: integer;
                       const BedVolumes, aDesignFlow, aAverageFlow : double;
                       var TotalCapital, OM, UsefulLife : double);

    property Info: TStringList read FInfo;
  end;

implementation

```

```
45 { TCostCurve }
46
47 procedure TCostCurve.values(const Q, AF: double; var Capital, OM: double);
48 begin
49   //Cost = C1 * Q ^ C2 + C3 * Ln(Q) + C4 + C5 * Exp (C6 * Q) + C7 * Q^3 + C8 * Q^2 + C9 * Q + C10
50   //using just 2 now until WBS example curves supply other functional forms
51   if PartCC=1 then begin
52     Capital := cc[1] * power(Q,cc[2]);
53   end else
54   if PartCC=4 then begin
55     Capital := cc[7] * intpower(Q,3) + cc[8] * Q * Q + cc[9] * Q + cc[10];
56   end else begin
57     Capital := -9e99;
58   end;
59
60   if PartOC=1 then begin
61     OM := oc[1] * power(Q,oc[2]);
62   end else
63   if PartOC=4 then begin
64     OM := oc[7] * intpower(AF,3) + oc[8] * AF * AF + oc[9] * AF + oc[10];
65   end else begin
66     OM := -9e99;
67   end;
68
69 end;
70
71 { TWBSCostCurves }
72
73 constructor TWBSCostCurves.Create(aLog : TSimplelog);
74 begin
75   Curves := TCostCurveDict.Create;
76   FileErrors:=0;
77   FInfo:=TStringLIst.Create;
78   FErrLog := aLog;
79 end;
80
81 destructor TWBSCostCurves.Destroy;
82 begin
83   Curves.Free;
84   FInfo.Free;
85   inherited;
86 end;
87
88 procedure TWBSCostCurves.FromID(const ID: integer; var Source,Tech,SubTech,SizeCat,CompLevel,BedVolume: integer);
89 begin
90
91 end;
92
93 procedure TWBSCostCurves.GetCosts(const C0Total: double; const Source, Tech, SubTech,
94   CompLevel: integer; const BedVolumes, aDesignFlow, aAverageFlow: double; var TotalCapital,
95   OM, UsefulLife: double);
96 begin
97   var DesignFlow := aDesignFlow;
98   if DesignFlow>162 then begin
99     DesignFlow := 162;
100    FErrLog.L(format('Design flow exceeded %g',[aDesignFlow]));
101  end;
102
103  //TODO How should this be handled?
104  if Tech = ttPOU then begin
105    TotalCapital := DesignFlow * 5;
106    UsefulLife := 10;
107    OM := DesignFlow * 0.1;
108    exit;
109  end;
110
111  var SizeCat := dfHigh;
112  if DesignFlow<1 then
113    SizeCat := dfLow
114  else
115  if DesignFlow<10 then
116    SizeCat := dfMed;
117  //TODO WARNING - hardcoding here for PFAS cost curves..
118  var cBV := 0;
```

```
  if Tech = ttGAC then cBV := max(trunc(BedVolumes/5000),1)  else
  if Tech = ttIX then cBV := max(trunc(BedVolumes/20000),1) ;
  var id := MakeID(Source, Tech, SubTech, SizeCat, CompLevel, cBV);
  OM := -9e9;
  TotalCapital := -9e9;
  UsefulLife := -9e9;
  var CC: TCostCurve;
  if Curves.TryGetValue(id, CC) then begin
    CC.values(DesignFlow, aAverageFlow, TotalCapital, OM);
    UsefulLife := CC.UsefulLife;
  end;
end;

function TWBSCostCurves.MakeID(Source, Tech, SubTech, SizeCat, CompLevel,
    BedVolume: integer): integer;
begin
  Result := Source;
  Result := Result*100 + tech;
  Result := Result*100 + subtech;
  Result := Result*10 + SizeCat;
  Result := Result*10 + CompLevel;
  Result := Result*100 + BedVolume;
end;


procedure TWBSCostCurves.ReadFile(aFIlename: string);
var ic: array[1..10] of byte;
begin
  const T = TStreamReader.Create(aFilename);
  const Hdr:TArray<string> = SplitString(T.ReadLine,',');
  const iSrc = IndexStr('GW/SW', Hdr);
  const iTech1 = IndexStr('Technology', Hdr);
  const iTech2 = IndexStr('Design Type', Hdr);
  const iSz = IndexStr('Size Category', Hdr);
  const iCmp = IndexStr('Comp Level', Hdr);
  const iBV = IndexStr('Bed Volumes', Hdr);
  const iCostType = IndexStr('Cost Type', Hdr);
  const iLife = IndexStr('Useful Life', Hdr);
  for var i := 1 to 10 do begin
    ic[i] := IndexStr('C'+i.ToString, Hdr);
  end;

  //TODO make sure all indexes are >0 (except bed volume)

  var ILine: TArray<string>;
  var sSrc,sTEch1,sTech2,sSz,sCmp,sBV,sCostType,sLife: string;
  var cSrc,cTech,cSubTech,cSz,cCmp,cBV,cCostType: integer;
  var CC: TCostCurve;
  while not T.EndOfStream do begin
    ILine := SplitString(T.ReadLine,',');
    sSrc := ILine[iSrc];   sTech1 := ILine[iTech1];
    sSz  := ILine[iSz];
    sCmp := ILine[iCmp];   sCostType := ILine[iCostType];
    sLife := ILine[iLife];
    if iBV>-1 then sBV := ILine[iBV] else sBV:='0';
    if iTech2>-1 then sTech2 := ILine[iTech2] else sTech2:='';

    //here comes the ugliness to convert to our internal ids
    if CompareText(sSrc,'gw')=0 then cSrc := swGroundWater else cSrc := swSurfaceWater;
    if CompareText(sSz,'small')=0 then cSz := dfLow else
    if CompareText(sSz,'medium')=0 then cSz := dfMed else
      cSz := dfHigh;
    if CompareText(sCmp,'low')=0 then cCmp := wbsCompLow else
    if CompareText(sCmp,'mid')=0 then cCmp := wbsCompMid else
      cCmp := wbsCompHigh;
    if CompareText(sCostType,'Total Capital')=0 then cCostType := 0 else
    if CompareText(sCostType,'Annual O&M')=0 then cCostType := 1 else
      cCostType := -1;
    var Tech := sTech1 + sTech2;
    if CompareText(Tech,'Non-TreatmentInterconnection')=0 then cTech := ttNTI else
    if CompareText(Tech,'Non-TreatmentNew Well Construction')=0 then cTech := ttNTNW else
    if CompareText(sTech1,'Reverse Osmosis / Nanofiltration')=0 then cTech := ttCRO else
    if CompareText(sTech1,'Granular Activated Carbon')=0 then cTech := ttGAC else
    if CompareText(sTech1,'Anion Exchange for PFAS')=0 then cTech := ttIX;
    cSubTech := stNone;
```

```
    if CompareText(sTech2,'Gravity')=0 then cSubTech := stGravity else
    if CompareText(sTech2,'Pressure')=0 then cSubTech := stPressure;
    cBV := 0;
    //TODO WARNING - hardcoding here for PFAS cost curves..  Find pattern from WBS experts
    if cTech = ttGAC then cBV := trunc(strtofloat(sBV)/5000) ;
    if cTech = ttIX then cBV := trunc(strtofloat(sBV)/20000) ;

    var id := MakeID(cSrc, cTech, cSubTech, cSz, cCmp, cBV);
    if not Curves.TryGetValue(id, CC) then begin
      CC := Default(TCostCurve);
    end;
    if cCostType = 0 then begin //capital
      for var i := 1 to 10 do
        CC.cc[i] := strtofloat(ILine[ic[i]]);
      CC.UsefulLife := strtofloat(sLife);
      if CC.cc[1]<>0 then CC.PartCC := 1 else CC.PartCC:=4;
    end else
    if cCostType = 1 then begin //o&m
      for var i := 1 to 10 do
        CC.oc[i] := strtofloat(ILine[ic[i]]);
      if CC.oc[1]<>0 then CC.PartOC := 1 else CC.PartOC:=4;
    end;
    Curves.AddOrSetValue(id, CC);
  end;
  T.Free;
end;

procedure TWBSCostCurves.ReadFiles(aFiles: TStringDynArray);
begin
  for var s in aFiles do begin
    ReadFile(s);
    FInfo.Add(ExtractFilename(s)+' read.')
  end;
  FInfo.Add('Curves read: '+ inttostr(Curves.Count * 2));
end;

end.

```

code\CostModules\cmPFASBase.pas

```
unit cmPFASBase;
interface

uses SysUtils, Types, Classes, StrUtils, Math, SafeWaterOccurrence, SafeWaterPWSRecords,
     SafeWaterModelData, SafeWaterTFObject, SafewaterMetrics,
     SafeWaterCostBase, SafeWaterUtility, SafeWaterGlobals, Spliner, CompiledCost,
     Generics.Collections;

type
  TTOCDistArray = array [0..12,0..1] of double;

const
  //TODO hard coding here instead of reading.  When format is formalized - we can read in...
  //Distributions of influent TOC concentration in mg/L by GW,SW
  TOCDistArraySW : TTOCDistArray =
    ((0,0.002), (0.05,0.65),(0.15,1.1),(0.25,1.38),(0.35,1.6),(0.45,1.85),
    (0.5,1.97),(0.55,2.14),(0.65,2.54),(0.75,3.04),(0.85,3.63),(0.95,4.81), (1,146.2));
  TOCDistArrayGW : TTOCDistArray =
    ((0,0.005), (0.05,0.35),(0.15,0.48),(0.25,0.5),(0.35,0.5),(0.45,0.58),
    (0.5,0.69),(0.55,0.75),(0.65,1),(0.75,1.39),(0.85,2.01),(0.95,3.8),(1,55.5));

  //b_ix values
  tabB_IX =  'HFPODA=212867,PFHxA=212867,PFBS=439515,PFHpA=319511,PFHxS=439515,'+
       'PFOA=390787,PFOS=439515,PFNA=390787';
  tabB_GAC = 'HFPODA=113034,PFHxA=113967,PFBS=129357,PFHpA=129357,PFHxS=129357,'+
       'PFOA=139862,PFOS=143731,PFNA=143731';

type
  //hacking in a method for VL systems with set "after" conc...
  TOCRec = record
    ID: string;
    conc: array[0..5] of double;
```

```
33      forcepop: double;
34    end;
35    TOCRecs = TDictionary<string,TOCRec>;
36
37    TcmPFASBase = class(TSafeWaterCostBase)
38      //custom output for GAC
39      epGACTOCBins,
40      popGACTOCBins,
41      rfGACTOCBINS,
42      sysGACTOCBins  : array[1..5] of double;
43
44
45      //choice outputs
46      tTechChoice: TTechnologies;
47      //technology forecast - viability/ pct set based on option process
48      tF : TTechnologies;
49      tFV: TTechnologiesViable;
50      CMax: integer; //contam highest index
51
52      //conc TOC
53      cTOC : double;
54      //sum of all concentrations
55      c0Total: double;
56      //system means
57      MN: TDoubleDynArray;
58      //MCLs
59      MCL: TDoubleDynArray;
60      //Multipliers (used in HI calcs)
61      CMULT: TDoubleDynArray;
62      //Detection Limits
63      DL: TDoubleDynArray;
64
65      //used to set after concentrations for adhoc large system benefit run
66      VLSC: TOCRecs;
67
68      //parameters for removal tech
69      B_IX, B_GAC : TDoubleDynArray;
70      Total_PFAS_limit,A_PFAS_IX,A_R_IX : double;
71      TOC_limit,A_TOC,A_R : double;
72      Max_BV_GAC,Max_BV_IX : double;
73
74      //spliners for TOC levels
75      TOC : array[swGroundWater..swSurfaceWater] of TSpliner;
76
77      //Bed volumes to keep for final costs
78      BV_GAC, BV_IX: double;
79      //for output
80      oBV_GAC, oBV_IX: double;
81      oBVRF_GAC, oBVRF_IX : double;
82
83      //percent removals
84      PR : TDoubleDynArray;
85      PRM: double; //maximum removal required
86
87      //variables needed for cost spreadsheet
88      AgencyWeight,b_exceed, b_nondetect, statepfas,
89      b_quarterly,b_annual,b_triennial,
90      b_nondetect_all_ep,b_treat_nondetect,epa533,epa537, b_source, b_treat : double;
91
92      //Cost Spreadsheets - Note: No baseline costs
93      csOption: TCCCompiledCostOption;
94
95      NoPOU,            //options can setthis to eliminate POU from consideration
96      FreshPWS: boolean;
97      RValueSelectTech : double;
98      OnEP: integer;
99      //TODO get this from config
100     MCLSafetyFactor: double;
101
102     //StateMCLS
103     StateMCLS : array of array[0..78] of double;
104     CountOccCap: double;
105     ixPFOA, ixPFOS: integer;
106
```

```
107     //PFAS specific ?  maybe more generally applicable
108     procedure RedistributeTechForecast;
109     procedure SetTechForecast;
110     procedure PerformanceGACIX; virtual;
111     function SelectTech(const P: TPWSRecord): integer;
112     function GetMaxMCL: double; override;
113     function GetWBSCosts(const P: TPWSRecord; const OnYrChange, TotYears, Tech: integer): boolean;
114
115     procedure DetermineDetect(const P: TPWSRecord); virtual;
116     procedure CapInputConc(const P: TPWSRecord; var Conc: TContamArray; const EPNo: integer; var UpdatePop: double); override;
117     procedure ForceOutputConc(const P: TPWSRecord; var FinalConc: TContamArray; const EPNo: integer); override;
118
119     procedure SetupTechParameters;
120     procedure SetupModule; override;
121     procedure BreakdownModule; override;
122     procedure SetupOutputs(Metrics: TCategoryMetrics); override;
123
124     procedure AddEntryPointCostsPostProcess(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
125     procedure SetPWSWideEPValues(const P: TPWSRecord); override;
126     procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
127   end;
128
129 implementation
130
131 { TcmPFASBase }
132
133 procedure TcmPFASBase.AddEntryPointCosts(const P: TPWSRecord;
134   const Conc: TContamArray; var FinalConc: TContamArray);
135 begin
136   inherited;
137   for var i := 0 to fData.Inp.RateCount - 1 do
138     _T[i] := Default(TCostOutputs);
139   c0Total := 0;
140   inc(OnEP);
141
142   for var c := 0 to CMax do
143     c0Total := c0Total + CurConc[c];
144   {$ifdef DEBUG}
145   if _D then
146     FData.DebugLog.L('costs','New Entry Point');
147   {$endif}
148   SetTechForecast;
149 end;
150
151 procedure TcmPFASBase.AddEntryPointCostsPostProcess(const P: TPWSRecord;
152   const Conc: TContamArray; var FinalConc: TContamArray);
153 begin
154   inherited;
155   for var bin := 1 to 5 do begin
156     if epGACTOCBins[bin]>0 then
157       rfGACTOCBins[bin] := rfGACTOCBins[bin]/epGACTOCBins[bin];
158   end;
159 end;
160
161 procedure TcmPFASBase.BreakdownModule;
162 begin
163   inherited;
164   for var T in TOC do
165     T.Free;
166   csOption.Free;
167   VLSC.Free;
168 end;
169
170 procedure TcmPFASBase.CapInputConc(const P: TPWSRecord; var Conc: TContamArray; const EPNo: integer; var UpdatePop: double);
171 begin
172   UpdatePop := 0;
173   for var c := 0 to CMax do begin
174     if Conc[c] > StateMCLS[c,P.StateFIPS] then begin
175        Conc[c] :=StateMCLS[c,P.StateFIPS];
176        CountOccCap := CountOccCap + 1;
177     end;
178   end;
179   if fData.Cfg.ForcePFASConc then begin
180     if P.SystemSize = ss1Mp then begin
181       //do not allow large systemms to be over MCL except when we have EP specific data
```

```
182       for var c := 0 to CMax do begin
183         if Conc[c] > MCL[c] then
184            Conc[c] := MCL[c] * 0.99;
185       end;
186
187       //shim for VLS concentrations - valid for all iterations
188       var id := P.PWSId + '_' + EpNo.ToString;
189       var OO: TOCRec;
190       if VLSC.TryGetValue(id,oo) then begin
191         fillchar(Conc[0],length(Conc), 0);
192         UpdatePop := OO.forcepop;
193         //crazy - please fix
194         if (id='FL4130871_1') or (id='FL4130871_2') then begin
195           Conc[ixPFOS] := 13; Conc[ixPFOA]:=4.7;
196         end else
197         if (id='FL4130871_3')  then begin
198           Conc[ixPFOS] := 22; Conc[ixPFOA]:=5.6;
199         end else
200         if (id='FL4130871_4')  then begin
201           Conc[ixPFOS] := 34; Conc[ixPFOA]:=11;
202         end else
203         if (id='NY5110526_1')  then begin
204           Conc[ixPFOS] := 8; Conc[ixPFOA]:=5.28;
205         end;
206       end;
207     end;
208   end;
209 end;
210
211 procedure TcmPFASBase.ForceOutputConc(const P: TPWSRecord; var FinalConc: TContamArray; const EPNo: integer);
212 begin
213   var OO: TOCRec;
214   if fData.Cfg.ForcePFASConc then begin
215     if P.SystemSize = ss1Mp then begin
216       var id := P.PWSId + '_' + EpNo.ToString;
217       if VLSC.TryGetValue(id,oo) then begin
218         for var c := 0 to CMax do
219           FinalConc[c] := OO.conc[c];
220       end;
221     end;
222   end;
223 end;
224
225
226 procedure TcmPFASBase.DetermineDetect(const P: TPWSRecord);
227 begin
228 (*
229   //this is called before the by entry point (and thus year) loop.
230   //NOTE:  mean isn't currently used - can be removed
231   b_nondetect_all_ep := 1;
232   for var c := 0 to CMax do Mn[c]:=0;
233    for var i := 1 to P.EPCount do begin
234     fData.Occ.GetAll(P.iPWSID, i, FIteration,CurConc);
235     for var c := 0 to CMax do begin
236       MN[c] := MN[c] + CurConc[c];
237       if (CurConc[c] > DL[c]) and (MCL[c]<1e30) then
238         b_nondetect_all_ep := 0;
239     end;
240   end;
241
242   for var c := 0 to CMax do
243     MN[c] := MN[c] / P.EPCount;
244 *)
245 end;
246
247 function TcmPFASBase.GetMaxMCL: double;
248 begin
249   Result := 0;
250   for var c := 0 to CMax do
251     if (MCL[c]>Result) and (MCL[c]<1e30) then
252       Result := MCL[c];
253 end;
254
255 function TcmPFASBase.GetWBSCosts(const P: TPWSRecord; const OnYrChange, TotYears, Tech: integer): boolean;
```

```
begin
  Result := True;
  var _CC, _OM, _UL, BV, tmpOM, tmpCC : double;
  var _CC1, _OM1, _CC2, _OM2, _UL1, _UL2,AC1,AC2 : double;
  const OnYr = FData.Cfg.StartYear + OnYrChange;
  _CC:=0; _OM:=0;
  var st := stNone;
  var i, y, yy, oyear, _ULR: integer;
  oBV_GAC := 0;
  oBV_IX := 0;
  oBVRF_GAC := 0;
  oBVRF_IX := 0;

  if Tech = ttNone  then begin
    Result := False;
    exit;
  end;
  inc(EPCostNo);

  if (Tech = ttGAC) and (P.SourceWater = swSurfaceWater) then begin
    BV := Min(BV_GAC, Max_BV_GAC);
    oBV_GAC := BV_GAC;
    oBVRF_GAC := (oBV_GAC * 10 * (Inf.DesignFlow / Inf.AverageFlow)) / 43200;
    FData.WBS.GetCosts(c0Total, P.SourceWater, Tech, stGravity, CompLevel, BV, Inf.DesignFlow, Inf.AverageFlow, _CC1, _OM1, _UL1);
    FData.WBS.GetCosts(c0Total, P.SourceWater, Tech, stPressure, CompLevel, BV, Inf.DesignFlow, Inf.AverageFlow, _CC2, _OM2, _UL2);
    //test cost comparision
    AC1 := _CC1/_UL1 + _OM1;
    AC2 := _CC2/_UL2 + _OM2;
    if AC1<AC2 then begin
      _CC := _CC1;
      _OM := _OM1;
      _UL := _UL1;
      st := stGravity;
    end else begin
      _CC := _CC2;
      _OM := _OM2;
      _UL := _UL2;
      st := stPressure;
    end;
  end else begin
    if Tech = ttIX then begin
      BV := Min(BV_IX, Max_BV_IX);
      oBV_IX := BV_IX;
      oBVRF_IX := (oBV_IX * 10 * (Inf.DesignFlow / Inf.AverageFlow)) / 43200;
    end else
    if Tech = ttGAC then begin
      BV := Min(BV_GAC, Max_BV_GAC);
      oBV_GAC := BV_GAC;
      oBVRF_GAC := (oBV_GAC * 10 * (Inf.DesignFlow / Inf.AverageFlow)) / 43200;
    end else
      BV := 0;
    if Tech = ttGAC then st := stPressure;
    if Tech = ttCRO then begin
      var df := min(PRM/0.95,1) * Inf.DesignFlow;
      var af := min(PRM/0.95,1) * Inf.AverageFlow;
      FData.WBS.GetCosts(c0Total, P.SourceWater, Tech, st, CompLevel, BV, df, af, _CC, _OM, _UL);
    end else
      FData.WBS.GetCosts(c0Total, P.SourceWater, Tech, st, CompLevel, BV, Inf.DesignFlow, Inf.AverageFlow, _CC, _OM, _UL);
  end;

  {$ifdef DEBUG}
    if _d then begin
      FData.DebugLog.L('costs','COSTS FROM WBS');
      FData.DebugLog.L('costs',Format('pws:%s, sw:%d, c0T:%g, df:%g, tech:%d, subtech:%d, bv:%g, complevel:%d, toc:%g, cc:%g, om:%g,  ulife:%g ',
                    [P.PWSId, P.SourceWater, SigFigsD(c0Total,5), SigFIgsD(Inf.DesignFlow,5), tech, st, SigFigsD(bv,5), complevel, SigFigsD(cTOC,5), SigFigsD(_cc,5), SigFigsD(_om,5), _UL]));
    end;

//FData.ErrLog.L(Format('pws, sw, c0T, df, tech, subtech, bv, complevel, toc, cc, om,  ulife',
//FData.ErrLog.L(Format('%s, %d, %g, %g, %d, %d, %g, %d, %g, %g, %g, %g ',
//              [P.PWSId, P.SourceWater, SigFigsD(c0Total,5), SigFIgsD(Inf.DesignFlow,5), tech, st, SigFigsD(bv,5), complevel, SigFigsD(cTOC,5), SigFigsD(_cc,5), SigFigsD(_om,5), _UL]));


  {$endif}
```

```
329
330   if (_cc<0) or (_om<0) then begin
331     var lo := Format('Negative costs calculated: pws:%s, sw:%d, c0T:%g, df:%g, tech:%d, subtech:%d, bv:%g, complevel:%d, toc:%g, cc:%g, om:%g ',
332                      [P.PWSId, P.SourceWater, SigFigsD(c0Total,5), SigFIgsD(Inf.DesignFlow,5), tech, st, SigFigsD(bv,5), complevel, SigFigsD(cTOC,5),
SigFigsD(_cc,5), SigFigsD(_om,5)]);
333     FData.ErrLog.L(lo);
334     Result := False;
335     exit;
336   end;
337
338   for i := 0 to fData.Inp.RateCount-1 do begin
339     if FData.Cfg.SingleCostYear=0 then begin
340       tmpCC := _cc / FData.Inp.PreCalcDiscRate[i,OnYrChange+FData.Cfg.PeriodAdjust];
341       _i(_T[i].PWSTreatment, tmpCC);
342       _i(_T[i].CapitalCost, tmpCC);
343       _i(OYearly[OnYrChange,i].PWSTreatment,tmpCC);
344       _i(OYearly[OnYrChange,i].CapitalCost,tmpCC);
345 //if i=0 then
346 //FData.ErrLog.L(Format('%s, %d, %d, %d, %g',[P.PWSId, FIteration, EPCostNo, OnYrChange,sigFigsD(tmpcc,5)]));
347
348       tmpCC := _cc / FData.Inp.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,i,OnYrChange+FData.Cfg.PeriodAdjust];
349       _i(_T[i].PWSTreatment_pws, tmpCC);
350       _i(_T[i].CapitalCost_pws, tmpCC);
351       _i(OYearly[OnYrChange,i].PWSTreatment_pws,tmpCC);
352       _i(OYearly[OnYrChange,i].CapitalCost_pws,tmpCC);
353
354       //add O&M
355       tmpOM := (_om * FData.Inp.PreCalcDiscRateSum[i,TotYears]) / FData.Inp.PreCalcDiscRate[i,OnYrChange+FData.Cfg.PeriodAdjust];
356       _i(_T[i].PWSTreatment, tmpOM);
357       _i(_T[i].OMCost, tmpOM);
358       for y := 0 to TotYears do begin
359         tmpOM := _om / FData.Inp.PreCalcDiscRate[i,OnYrChange+y+FData.Cfg.PeriodAdjust];
360         _i(OYearly[y + OnYrChange,i].PWSTreatment,tmpOM);
361         _i(OYearly[y + OnYrChange,i].OMCost,tmpOM);
362       end;
363
364       tmpOM := _om * FData.Inp.PWSPreCalcDiscRateSum[P.SourceWater,P.SystemSize,i,TotYears] / FData.Inp.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,
i,OnYrChange+FData.Cfg.PeriodAdjust] ;
365       _i(_T[i].PWSTreatment_pws, tmpOM);
366       _i(_T[i].OMCost_pws, tmpOM);
367       for y := 0 to TotYears do begin
368         tmpOM := _om / FData.Inp.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,i,OnYrChange+y+FData.Cfg.PeriodAdjust];
369         _i(OYearly[y + OnYrChange,i].PWSTreatment_pws,tmpOM);
370         _i(OYearly[y + OnYrChange,i].OMCost_pws,tmpOM);
371       end;
372       //check if useful life is less than remainder of analysis period
373       _ULR := ceil(_UL);
374       if FData.Cfg.PFASChangeYear + _ULR < FData.Cfg.EndYear then begin
375         yy := _ULR;
376         while yy < TotYears do begin
377            oyear := round(yy) + OnYrChange;
378            tmpCC := _cc / FData.Inp.PreCalcDiscRate[i,oyear+FData.Cfg.PeriodAdjust];
379            _i(_T[i].PWSTreatment, tmpCC);
380            _i(_T[i].CapitalCost, tmpCC);
381            _i(OYearly[oyear,i].PWSTreatment,tmpCC);
382            _i(OYearly[oyear,i].CapitalCost,tmpCC);
383 //if i=0 then
384 //FData.ErrLog.L(Format('%s, %d, %d, %d, %g',[P.PWSId, FIteration, EPCostNo, Oyear,sigFigsD(tmpcc,5)]));
385
386            tmpCC := _cc / FData.Inp.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,i,oyear+FData.Cfg.PeriodAdjust];
387            _i(_T[i].PWSTreatment_pws, tmpCC);
388            _i(_T[i].CapitalCost_pws, tmpCC);
389            _i(OYearly[oyear,i].PWSTreatment_pws,tmpCC);
390            _i(OYearly[oyear,i].CapitalCost_pws,tmpCC);
391            yy := yy + _ULR;
392         end;
393       end;
394     end else begin
395       //TODO Deprecate now that yearly results are available...
396       raise Exception.Create('Single Cost year deprecated');
397       if (FData.Cfg.StartYear + OnYrChange = FData.Cfg.SingleCostYear) then begin
398         tmpCC := _cc / FData.Inp.PreCalcDiscRate[i,OnYrChange];
399         _i(_T[i].PWSTreatment, tmpCC);
400         _i(_T[i].CapitalCost, tmpCC);
401         _i(OYearly[OnYrChange,i].PWSTreatment,tmpCC);
```

```
402         _i(OYearly[OnYrChange,i].CapitalCost,tmpCC);
403         tmpCC := _cc / FData.Inp.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,i,OnYrChange];
404         _i(_T[i].PWSTreatment_pws, tmpCC);
405         _i(_T[i].CapitalCost_pws, tmpCC);
406         _i(OYearly[OnYrChange,i].PWSTreatment_pws,tmpCC);
407         _i(OYearly[OnYrChange,i].CapitalCost_pws,tmpCC);
408
409       end;
410       //just add 1 year of O&M
411       if (FData.Cfg.StartYear + OnYrChange >= FData.Cfg.SingleCostYear) then begin
412          tmpOM := _om / FData.Inp.PreCalcDiscRate[i,FData.Cfg.SingleCostYear - FData.Cfg.StartYear];
413         _i(_T[i].PWSTreatment, tmpOM);
414         _i(_T[i].OMCost, tmpOM);
415         _i(OYearly[FData.Cfg.SingleCostYear - FData.Cfg.StartYear,i].PWSTreatment,tmpOM);
416         _i(OYearly[FData.Cfg.SingleCostYear - FData.Cfg.StartYear,i].OMCost,tmpOM);
417          tmpOM := _om / FData.Inp.PWSPreCalcDiscRate[P.SourceWater,P.SystemSize,i,FData.Cfg.SingleCostYear - FData.Cfg.StartYear];
418         _i(_T[i].PWSTreatment_pws, tmpOM);
419         _i(_T[i].OMCost_pws, tmpOM);
420         _i(OYearly[FData.Cfg.SingleCostYear - FData.Cfg.StartYear,i].PWSTreatment_pws,tmpOM);
421         _i(OYearly[FData.Cfg.SingleCostYear - FData.Cfg.StartYear,i].OMCost_pws,tmpOM);
422       end;
423     end;
424   end;
425 end;
426
427 function _gpr(const c0,mcl,mclmult,sf: double; var PRM: double) : double;
428 begin
429   if (c0>0) and (mcl<1e30) and (c0>mcl*mclmult) then
430     Result := (c0 - mcl * mclmult * sf) / c0
431   else
432     Result := 0;
433   if Result > PRM then PRM := Result;
434 end;
435
436 procedure TcmPFASBase.PerformanceGACIX;
437 begin
438   var i: integer;
439   PRM := 0;
440   //TODO should these be set to large numbers in the absence of MCLs
441   for i := 0 to CMax do
442      PR[i] := _gpr(CurConc[i], MCL[i], CMULT[i], MCLSafetyFactor,PRM);
443
444   if (PRM=0) then begin
445     exit;
446   end;
447
448   if PRM > 0.99 then begin
449     tFV[ttGAC] := false;
450     //per EPA email 10/12/23 - Make IX viable at a higher removal pct
451     if PRM > 0.996 then
452        tFV[ttIX] := false;
453   end;
454
455   BV_GAC := 1e30;
456   if tFV[ttGAC] then
457     for i := 0 to CMax do
458       if PR[i]>0 then
459         BV_GAC := min(BV_GAC,A_TOC * cTOC + A_R * PR[i] + B_GAC[i]);
460   if BV_GAC < 5000 then tFV[ttGAC] := false;
461
462   BV_IX := 1e30;
463   if tFV[ttIX] then
464     for i := 0 to CMax do
465       if PR[i]>0 then
466         BV_IX := min(BV_IX, A_PFAS_IX * C0Total + A_R_IX * PR[i] + B_IX[i]);
467   if BV_IX < 20000 then tFV[ttIX] := false;
468
469   RedistributeTechForecast;
470 end;
471
472 procedure TcmPFASBase.RedistributeTechForecast;
473 begin
474   {$ifdef DEBUG}
475   if _d then begin
476     FData.DebugLog.L('costs','Redistrbute Tech Forecast Start, Option: '+fOptionName);
```

```
477     for var i := ttGAC to ttNTNW do begin
478       FData.DebugLog.L('costs',FData.TechForecast.TechToText(i)+':'+ floattostr(TF[i]) +'  viable:'+booltostr(tFV[i]));
479     end;
480   end;
481   {$endif}
482
483   var tnv: double := 0;
484   for var i := ttGAC to ttNTNW do begin
485     if not tFV[i] then
486       TF[i] := 0
487     else
488       tnv := tnv + TF[i];
489   end;
490   if tnv = 0 then exit;
491   var s:double := 0;
492   for var i := ttGAC to ttNTNW do begin
493      TF[i] := TF[i] / tnv;
494      s:=s+TF[i];
495   end;
496
497   if (s<0.999) and (s>1.001) then
498     FData.ErrLog.LD('Tech forecast probabilty error: totalprob',s);
499
500   {$ifdef DEBUG}
501   if _d then begin
502     FData.DebugLog.L('costs','Redistrbute Tech Forecast End, Option: '+fOptionName);
503     for var i := ttGAC to ttNTNW do begin
504       FData.DebugLog.L('costs',FData.TechForecast.TechToText(i)+':'+ floattostr(TF[i]) +'  viable:'+booltostr(tFV[i]));
505     end;
506   end;
507   {$endif}
508 end;
509
510 function TcmPFASBase.SelectTech(const P: TPWSRecord): integer;
511 begin
512   Result := -1;
513   var r: double;
514   var s: integer;
515   //TODO Tech is chosen randomly by entry point - OK?
516   s := (FIteration + OnEP) Mod high(P.rn);
517   r := P.rn[s];
518
519   var a: double := 0;
520   for var i := ttGAC to ttNTNW do begin
521     a := a + tF[i];
522     if r <= a then begin
523       Result := i;
524       //7.20.2022 changing to PWS count instead of entry point
525       //tTechChoice[i] := tTechChoice[i] + 1;
526       tTechChoice[i] := max(tTechChoice[i] + 1, 1);
527       break;
528     end;
529   end;
530
531
532   if Result=-1 then begin
533     FData.ErrLog.L(P.Outdata);
534     FData.ErrLog.LD('Failed random tech choice',r);
535     FData.ErrLog.LD('Failed random tech choice TotalProb',a);
536     FData.ErrLog.L('Current Tech Options: '+fOptionName);
537     FData.ErrLog.LD('TOC:', cTOC);
538     FData.ErrLog.LD('C0TOTAL:', c0Total);
539     FData.ErrLog.LD('BV_GAC:', BV_GAC);
540     FData.ErrLog.LD('BV_IX:', BV_IX);
541     FData.ErrLog.LD('Design Flow:', Inf.DesignFlow);
542     for var i := ttGAC to ttNTNW do begin
543       FData.ErrLog.LD(FData.TechForecast.TechToText(i),TF[i]);
544       FData.ErrLog.LD('viable', booltostr(tFV[i]));
545     end;
546     FData.ErrLog.L('**********');
547     exit;
548   end;
549
550   //If tech = GAC update custom metric for GACTOC
```

```
551   var bin: integer := 0;
552   if Result = ttGAC then begin
553      if cTOC<1 then bin := 1 else
554      if cTOC<2 then bin := 2 else
555      if cTOC<3.5 then bin := 3 else
556      if cTOC<5 then bin := 4 else
557        bin := 5;
558      sysGACTOCBins[bin] := 1;
559      epGACTOCBins[bin] := epGACTOCBins[bin] + 1;
560      popGACTOCBins[bin] := popGACTOCBins[bin] + CurPop;
561      //denom = 60 * 24 * 30
562      rfGACTOCBins[bin] := rfGACTOCBins[bin] + (BV_GAC * 10 * (Inf.DesignFlow / Inf.AverageFlow)) / 43200;
563   end;
564   FCostInfo.GACTOCBin := bin;
565
566   Inf.SystemIsTreating := 1;
567   Inf.PWSPopTreating := P.Population;
568   Inf.EntryPointPopTreating := Inf.EntryPointPopTreating + CurPop;
569   Inf.EntryPointIsTreating := Inf.EntryPointIsTreating + 1;
570 end;
571
572 procedure TcmPFASBase.SetPWSWideEPValues(const P: TPWSRecord);
573 begin
574   inherited;
575   fillchar(popGACTOCBins,sizeof(popGACTOCBins),0);
576   fillchar(epGACTOCBins,sizeof(epGACTOCBins),0);
577   fillchar(sysGACTOCBins,sizeof(sysGACTOCBins),0);
578   fillchar(rfGACTOCBins,sizeof(rfGACTOCBins),0);
579
580   //use a static PWS random value to assign TOC
581   cTOC := TOC[P.SourceWater].SplineValue(P.rn[1]);
582   DetermineDetect(P);
583   fillchar(tTechChoice, sizeof(tTechChoice), 0);
584   FreshPWS := True;
585   AgencyWeight := P.AgencyWeight;
586   StatePFAS := P.StatePFAS;
587   OnEp:=0;
588   CountOccCap:=0;
589
590   {$ifdef DEBUG}
591   if _d then begin
592     FData.DebugLog.LD('costs','TOC',cTOC);
593     FData.DebugLog.LD('costs','b_nondetect_all_ep', b_nondetect_all_ep);
594     FData.DebugLog.LD('costs','state_pfas', statepfas);
595     FData.DebugLog.L('costs','----start entry points----');
596   end;
597   {$endif}
598 end;
599
600 procedure TcmPFASBase.SetTechForecast;
601 begin
602   fillchar(tFV, sizeof(tFV), true);
603   FData.TechForecast.GetTechForecast(Inf.DesignFlow, cTOC, tf);
604   for var i := ttGAC to ttNTNW do
605     if TF[i]=0 then tFV[i] := false;
606
607   //set limits common to options
608   if cTOC > 3.2 then tFV[ttGAC] := false;
609   if c0Total > Total_PFAS_limit then tFV[ttIX] := false; //7044
610   if c0Total > 1500 then tFV[ttPOU] := false;
611   if Inf.DesignFlow > 3.536 then begin
612     tFV[ttNTI] := false;
613     tFV[ttNTNW] := false;
614   end;
615   if NoPOU then tFV[ttPOU] := false;
616
617   {$ifdef DEBUG}
618   if _d then begin
619     FData.DebugLog.L('costs','Init Tech Forecast, Option:'+FOptionName);
620     for var i := ttGAC to ttNTNW do begin
621       FData.DebugLog.L('costs',FData.TechForecast.TechToText(i)+':'+ floattostr(TF[i]) +'  viable:'+booltostr(tFV[i]));
622     end;
623     FData.DebugLog.LD('costs','TOC', cTOC);
624     FData.DebugLog.LD('costs','c0Total', c0Total);
625   end;
```

```
626   {$endif}
627
628   RedistributeTechForecast;
629 end;
630
631 procedure TcmPFASBase.SetupModule;
632 begin
633   inherited;
634   MCLSafetyFactor := 0.8;
635   NoPOU := False;
636   CMax := FData.Occ.ContamList.Count - 1;
637
638   //very ungeneral here - specific to PFAS so leave for now..
639   setlength(StateMCLS, CMax+1);
640   for var i := 0 to CMax do begin
641     for var f := 0 to high(StateMCLs[i]) do begin
642       StateMCLs[i,f] := 1e99;
643     end;
644   end;
645   //Set State MCLs from table
646   var ix := FData.Occ.ContamList.IndexOf('PFOA');
647   ixPFOA := ix;
648   if ix>=0 then begin
649     StateMCLs[ix,34] := 14;  //NJ
650     StateMCLs[ix,33] := 12;  //NH
651     StateMCLs[ix,26] := 8;  //MI
652     StateMCLs[ix,36] := 10;  //NY
653     StateMCLs[ix,42] := 14;  //PA
654     StateMCLs[ix,55] := 70;  //WI
655   end;
656   ix := FData.Occ.ContamList.IndexOf('PFOS');
657   ixPFOS := ix;
658   if ix>=0 then begin
659     StateMCLs[ix,34] := 13;  //NJ
660     StateMCLs[ix,33] := 15;  //NH
661     StateMCLs[ix,26] := 16;  //MI
662     StateMCLs[ix,36] := 10;  //NY
663     StateMCLs[ix,42] := 18;  //PA
664     StateMCLs[ix,55] := 70;  //WI
665   end;
666   ix := FData.Occ.ContamList.IndexOf('PFHXS');
667   if ix>=0 then begin
668     StateMCLs[ix,33] := 18;  //NH
669     StateMCLs[ix,26] := 51;  //MI
670   end;
671   ix := FData.Occ.ContamList.IndexOf('PFBS');
672   if ix>=0 then begin
673     StateMCLs[ix,26] := 420;  //MI
674   end;
675   ix := FData.Occ.ContamList.IndexOf('PFNA');
676   if ix>=0 then begin
677     StateMCLs[ix,34] := 13;  //NJ
678     StateMCLs[ix,33] := 11;  //NH
679     StateMCLs[ix,26] := 6;  //MI
680   end;
681   ix := FData.Occ.ContamList.IndexOf('HFPODA');
682   if ix>=0 then begin
683     StateMCLs[ix,26] := 370;  //MI
684   end;
685
686
687   setlength(B_IX,CMax + 1);
688   setlength(B_GAC,CMax + 1);
689   setlength(PR,CMax + 1);
690   setlength(MCL,CMax + 1);
691   setlength(CMULT,CMax + 1);
692   setlength(DL,CMax + 1);
693   setlength(MN,CMax + 1);
694   VLSC := TOCRecs.Create;
695   SetupTechParameters;
696   for var i := swGroundWater to swSurfacewater do begin
697     TOC[I] := TSpliner.create;
698     TOC[I].Method := smLinear;
699     TOC[I].SetDataLength(High(TOCDistArrayGW) + 1);
```

```
700     if i = swGroundWater then begin
701       for var j := Low(TOCDistArrayGW) to High(TOCDistArrayGW) do begin
702         TOC[i].AddData(TOCDistArrayGW[j,0],TOCDistArrayGW[j,1]);
703       end;
704     end else begin
705       for var j := Low(TOCDistArraySW) to High(TOCDistArraySW) do begin
706         TOC[i].AddData(TOCDistArraySW[j,0],TOCDistArraySW[j,1]);
707       end;
708     end;
709   end;
710   csOption:=TCCCompiledCostOption.create;
711
712   //To run costs for only a single year (Usually implemntation year).
713   if FData.Cfg.SingleCostYear>0 then
714    csOption._SetSingleYear(FData.Cfg.SingleCostYear - FData.Cfg.StartYear);
715
716   //TODO set epa537 epa533 somewhere in Config
717   csOption._SetVarPointer('b_quarterly', @b_quarterly);
718   csOption._SetVarPointer('b_annual', @b_annual);
719   csOption._SetVarPointer('b_triennial', @b_triennial);
720   csOption._SetVarPointer('b_exceed', @b_exceed);
721   csOption._SetVarPointer('b_nondetect', @b_nondetect);
722
723   //TODO these 2 are not in workbook now - are they deprecated?
724   csOption._SetVarPointer('b_nondetect_all_ep', @b_nondetect_all_ep);
725   csOption._SetVarPointer('b_treat_nondetect', @b_treat_nondetect);
726
727   csOption._SetVarPointer('b_treat', @b_treat);
728   csOption._SetVarPointer('b_source', @b_source);
729   csOption._SetVarPointer('epa533', @epa533);
730   csOption._SetVarPointer('epa537', @epa537);
731   csOption._SetVarPointer('aw', @AgencyWeight);
732   csOption._SetVarPointer('statepfas', @StatePFAS);
733
734   //TODO set this in Config????
735   epa533 := 1;
736
737   csOption._SetOutPointer(_T, OYearly);
738
739   //TODO - set these from config - probably in descendant SetupModule
740   for var i := 0 to CMax do begin
741     MCL[i] := 1e30;
742     DL[i]:=0;
743     CMULT[i] := 1;
744   end;
745
746   ix := FData.Occ.ContamList.IndexOf('PFOS');
747   if ix>-1 then MCL[ix] := 4.05;  //0.05 added - rounded for rule 4.049 = 4
748   if ix>-1 then DL[ix] := 1.1;
749
750
751   ix := FData.Occ.ContamList.IndexOf('PFOA');
752   if ix>-1 then MCL[ix] := 4.05; //0.05 added - rounded for rule 4.049 = 4
753   if ix>-1 then DL[ix] := 0.53;
754
755   //Note - this is a fragile method to include large system benefits for a couple systems
756   if FData.Cfg.ForcePFASConc then begin
757      var OCR: TOCRec;
758      var TR := TStreamReader.Create(FData.Cfg.ForceConcFileName);
759      var iLine := TStringLIst.Create;
760      var iHdr := TStringLIst.Create;
761      iHdr.CommaText := uppercase(TR.ReadLine);
762      while not TR.EndOfStream do begin
763        iLine.CommaText := TR.ReadLine;
764        if CompareText(fOptionName,iLine[0])<>0 then continue;
765        OCR.ID := iLine[1];
766        for ix := 0 to FData.Occ.ContamList.Count-1 do begin
767          if ix>5 then raise exception.Create('Contam Count exceeded for VLS');
768          var iix := iHdr.IndexOf(FData.Occ.ContamList[ix]);
769          OCR.conc[ix] := strtofloat(ILine[iix]);
770        end;
771        OCR.forcepop:=strtofloat(ILine[8]);
772        VLSC.AddOrSetValue(OCR.ID,OCR);
773      end;
774      iLine.Free;
```

```
775      iHdr.Free;
776      TR.Free;
777    end;
778
779  end;
780
781  procedure TcmPFASBase.SetupOutputs(Metrics: TCategoryMetrics);
782  begin
783    inherited;
784    Metrics.AddMetric(fOptionName+'.TechChoiceGAC',
785                      'Tech Choice GAC PWS Count', @tTechChoice[ttGAC], mtInfo, mcSumOnly, False);
786    Metrics.AddMetric(fOptionName+'.TechChoiceIX',
787                      'Tech Choice IX PWS Count', @tTechChoice[ttIX], mtInfo, mcSumOnly, False);
788    Metrics.AddMetric(fOptionName+'.TechChoiceCRO',
789                      'Tech Choice CRO PWS Count', @tTechChoice[ttCRO], mtInfo, mcSumOnly, False);
790    Metrics.AddMetric(fOptionName+'.TechChoicePOU',
791                      'Tech Choice POU PWS Count', @tTechChoice[ttPOU], mtInfo, mcSumOnly, False);
792    Metrics.AddMetric(fOptionName+'.TechChoiceNTI',
793                      'Tech Choice NTI PWS Count', @tTechChoice[ttNTI], mtInfo, mcSumOnly, False);
794    Metrics.AddMetric(fOptionName+'.TechChoiceNTNW',
795                      'Tech Choice NTNW PWS Count', @tTechChoice[ttNTNW], mtInfo, mcSumOnly, False);
796    Metrics.AddMetric(fOptionName+'.AgencyWeight',
797                      'Sum Of Agency Weights', @AgencyWeight, mtInfo, mcSumOnly, False);
798
799    Metrics.AddMetric(fOptionName+'.AgencyWeight',
800                      'Sum Of Agency Weights', @AgencyWeight, mtInfo, mcSumOnly, False);
801
802    Metrics.AddMetric(fOptionName+'.BV_IX',
803                      'Bed Volume for IX', @oBV_IX, mtInfo, mcAll, true);
804    Metrics.AddMetric(fOptionName+'.BV_GAC',
805                      'Bed Volume for GAC', @oBV_GAC, mtInfo, mcAll, true);
806    Metrics.AddMetric(fOptionName+'.BV_IX',
807                      'Bed Volume RF for IX', @oBVRF_IX, mtInfo, mcAll, true);
808    Metrics.AddMetric(fOptionName+'.BV_GAC',
809                      'Bed Volume RF for GAC', @oBVRF_GAC, mtInfo, mcAll, true);
810
811    Metrics.AddMetric(fOptionName+'.CountOccCap',
812                      'Count of Contams at EPs Set To MCL', @CountOccCap, mtInfo, mcSumOnly, False);
813
814    for var Opt in FData.Cfg.OutputOptions do begin
815      if CompareText(Opt,'GACTOC')= 0 then begin
816        for var i := 1 to 5 do begin
817          Metrics.AddMetric(fOptionName+'.epGACTOCBin'+i.ToString,
818                            'Number of Eps Installing GAC, TOC BIN '+i.ToString, @epGACTOCBins[i], mtCostExtended, mcSumOnly, False);
819          Metrics.AddMetric(fOptionName+'.sysGACTOCBin'+i.ToString,
820                            'Number of Systems with at least One EP Installing GAC, TOC BIN '+i.ToString, @sysGACTOCBins[i], mtCostExtended, mcSumOnly,
False);
821          Metrics.AddMetric(fOptionName+'.rfGACTOCBin'+i.ToString,
822                            'Replacement frequency for EPs installing GAC, TOC BIN '+i.ToString, @rfGACTOCBins[i], mtCostExtended, mcAll, true);
823        end;
824      end else
825      if CompareText(Opt,'GACTOCPOP')= 0 then begin
826        for var i := 1 to 5 do begin
827          Metrics.AddMetric(fOptionName+'.popGACTOCBin'+i.ToString,
828                            'Total Population of EPs Installing GAC, TOC BIN '+i.ToString, @popGACTOCBins[i], mtCostExtended, mcSumOnly, False);
829        end;
830      end;
831
832    end;
833  end;
834
835  procedure TcmPFASBase.SetupTechParameters;
836  begin
837    var T := TstringList.Create;
838    T.CommaText := uppercase(tabB_IX);
839    for var i := 0 to CMax do begin
840      var ix := T.IndexOfName(FData.Occ.ContamList[i]);
841      if ix > -1 then
842        B_IX[i] := strtofloat(T.Values[FData.Occ.ContamList[i]])
843      else
844        B_IX[i] := 0;
845    end;
846
847    T.CommaText := uppercase(tabB_GAC);
848    for var i := 0 to CMax do begin
```

```
    var ix := T.IndexOfName(FData.Occ.ContamList[i]);
    if ix > -1 then
      B_GAC[i] := strtofloat(T.Values[FData.Occ.ContamList[i]])
    else
      B_GAC[i] := 0;
  end;
  T.Free;

  TOC_limit := 3.2;
  A_TOC := -37932;
  A_R := -36309;
  Total_PFAS_limit := 7044;
  A_PFAS_IX := -6.04;
  A_R_IX := -198242;
  Max_BV_GAC := 75000;
  Max_BV_IX := 260000;

end;


end.
```

code\CostModules\cmPFASOccRun.pas

```
unit cmPFASOccRun;

interface

uses Sysutils, Classes,SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterCostBase,
     SafeWaterUtility, SafewaterGlobals, SafeWaterTFObject, cmPFASBase, Math,
     SafewaterMetrics, Generics.Collections;

type
  TOCount=record
    PWS,EP, PWSPop,EPPop, EPPopConc : double;
  end;
  TOConc = record
    All, GMCL : double;
  end;
  TOCounts = array of TOCount;
  TOConcs = array of TOConc;

  TcmPFASOccRun = class(TcmPFASBase)
    aFlag: boolean;
    ECounts,CCounts, OCounts : TOCounts;
    MCount,ECountT: TOCount;
    xConc : TOConcs;
    ixPFHXS,ixPFNA,ixHFPODA   : integer;
    cgPFNA, cgPFHXS, cgHFPODA: double;

    constructor Create(aData : PSWModelData; GlbUnc: TGlobalUncertainty;  aNeededContams: string; aOptionName: string; aIteration: integer); override;
    procedure SetPWSWideEPValues(const P: TPWSRecord); override;
    procedure SetupModule; override;
    procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
  end;

implementation


{ TcmPFASOccRun }

procedure TcmPFASOccRun.AddEntryPointCosts(const P: TPWSRecord;
  const Conc: TContamArray; var FinalConc: TContamArray);
begin
  inherited;
  var TotYears := FData.Cfg.EndYear - FData.Cfg.PFASChangeYear;
  var OnYrChange := FData.Cfg.PFASChangeYear - FData.Cfg.StartYear;

  const ImUnder = (CurConc[ixPFOA] < MCL[ixPFOA]) and (CurConc[ixPFOS] < MCL[ixPFOS]);
```

```
47    b_exceed := 0;
48    var c : integer;
49    for c := 0 to CMax do begin
50      xConc[c].All := xConc[c].All + CurConc[c]/P.EPCount;
51      if CurConc[c] >= MCL[c] then begin
52        if CurConc[c]>xConc[c].GMCL then
53          xConc[c].GMCL := CurConc[c];
54        if (ImUnder) and (c<>ixPFOA) and (c<>ixPFOS) then begin
55          CCounts[c].PWS := 1;
56          CCounts[c].PWSPop := P.Population;
57          CCounts[c].EP := CCounts[c].EP + 1;
58          CCounts[c].EPPop := CCounts[c].EPPop + CurPop;
59        end else
60        if (not ImUnder) and (c<>ixPFOA) and (c<>ixPFOS) then begin
61          OCounts[c].PWS := 1;
62          OCounts[c].PWSPop := P.Population;
63          OCounts[c].EP := OCounts[c].EP + 1;
64          OCounts[c].EPPop := OCounts[c].EPPop + CurPop;
65        end;
66        ECounts[c].PWS := 1;
67        ECounts[c].PWSPop := P.Population;
68        ECounts[c].EP := ECounts[c].EP + 1;
69        ECounts[c].EPPop := ECounts[c].EPPop + CurPop;
70        ECounts[c].EPPopConc := ECounts[c].EPPopConc + CurPop * CurConc[c];
71        if b_exceed=0 then begin
72          ECountT.PWS := 1;
73          ECountT.PWSPop := P.Population;
74          ECountT.EP := ECountT.EP + 1;
75          ECountT.EPPop := ECountT.EPPop + CurPop;
76        end;
77        b_exceed:=1;
78      end;
79    end;
80
81    if (ImUnder) and (Conc[ixPFNA]>MCL[ixPFNA]) then  cgPFNA := Conc[ixPFNA];
82    if (ImUnder) and (Conc[ixPFHXS]>MCL[ixPFHXS]) then  cgPFHXS := Conc[ixPFHXS];
83    if (ImUnder) and (Conc[ixHFPODA]>MCL[ixHFPODA]) then  cgHFPODA := Conc[ixHFPODA];
84
85    if not ImUnder then begin
86      MCount.PWS := 1;
87      MCount.PWSPop := P.Population;
88      MCount.EP := MCount.EP + 1;
89      MCount.EPPop := MCount.EPPop + CurPop;
90     end;
91  end;
92
93
94  constructor TcmPFASOccRun.Create(aData: PSWModelData;
95    GlbUnc: TGlobalUncertainty; aNeededContams, aOptionName: string;
96    aIteration: integer);
97  begin
98    inherited;
99    setlength(ECounts,FData.Occ.ContamList.Count);
100   setlength(CCounts,FData.Occ.ContamList.Count);
101   setlength(OCounts,FData.Occ.ContamList.Count);
102   setlength(xConc,FData.Occ.ContamList.Count);
103   ixPFOS := FData.Occ.ContamList.IndexOf('PFOS');
104   ixPFOA := FData.Occ.ContamList.IndexOf('PFOA');
105   ixPFHXS := FData.Occ.ContamList.IndexOf('PFHXS');
106   ixHFPODA := FData.Occ.ContamList.IndexOf('HFPODA');
107   ixPFNA := FData.Occ.ContamList.IndexOf('PFNA');
108 end;
109
110 procedure TcmPFASOccRun.SetPWSWideEPValues(const P: TPWSRecord);
111 begin
112   inherited;
113   ECountT := Default(TOCOunt);
114   MCount := Default(TOCOunt);
115
116   for var ix := 0 to FData.Occ.ContamList.Count-1 do begin
117     ECounts[ix] := Default(TOCOunt);
118     CCounts[ix] := Default(TOCOunt);
119     xConc[ix] := Default(TOConc);
120     OCounts[ix] := Default(TOCOunt);
121   end;
```

```
122   cgPFNA:=0; cgPFHXS:=0; cgHFPODA:=0;
123 end;
124
125 procedure TcmPFASOccRun.SetupModule;
126 begin
127   inherited;
128   var ix : integer;
129   MCLSafetyFactor := 0.8;
130
131   //reset everything
132   for var i := 0 to CMax do begin
133     MCL[i] := 1e30;
134     DL[i]:=0;
135     CMULT[i] := 1;
136   end;
137
138   ix := FData.Occ.ContamList.IndexOf('PFOS');
139   if ix>-1 then MCL[ix] := 4.05;  //0.05 added - rounded for rule 4.049 = 4
140   if ix>-1 then DL[ix] := 1.1;
141   ix := FData.Occ.ContamList.IndexOf('PFOA');
142   if ix>-1 then MCL[ix] := 4.05; //0.05 added - rounded for rule 4.049 = 4
143   if ix>-1 then DL[ix] := 0.53;
144   ix := FData.Occ.ContamList.IndexOf('PFNA');
145   if ix>-1 then MCL[ix] := 15;
146   if ix>-1 then DL[ix] := 0.7;
147   ix := FData.Occ.ContamList.IndexOf('HFPODA');
148   if ix>-1 then MCL[ix] := 15;
149   if ix>-1 then DL[ix] := 1.9;
150   ix := FData.Occ.ContamList.IndexOf('PFHXS');
151   if ix>-1 then MCL[ix] := 15;
152   if ix>-1 then DL[ix] := 1.4;
153
154 end;
155
156 procedure TcmPFASOccRun.SetupOutputs(Metrics: TCategoryMetrics);
157 begin
158   inherited;
159
160   for var ix := 0 to FData.Occ.ContamList.Count-1 do begin
161     var s := FData.Occ.ContamList[ix];
162
163     Metrics.AddMetric(fOptionName+'.'+s+'PWS',
164                       'PWSs with '+s+' Exceedance', @ECounts[ix].PWS, mtAdhoc, mcSumOnly, False);
165     Metrics.AddMetric(fOptionName+'.'+s+'PWSPOP',
166                     'PWS Population Impacted by '+s+' Exceedance', @ECounts[ix].PWSPop, mtAdhoc, mcSumOnly, False);
167     Metrics.AddMetric(fOptionName+'.'+s+'EP',
168                       'EPs with '+s+' Exceedance', @ECounts[ix].EP, mtAdhoc, mcSumOnly, False);
169     Metrics.AddMetric(fOptionName+'.'+s+'EPPOP',
170                     'EPs Population Impacted by '+s+' Exceedance', @ECounts[ix].EPPop, mtAdhoc, mcSumOnly, False);
171     Metrics.AddMetric(fOptionName+'.'+s+'EPPOPConc',
172                     'EPs Pop Weighted Conc '+s+' Exceedance', @ECounts[ix].EPPopConc, mtAdhoc, mcSumOnly, False);
173
174     Metrics.AddMetric(fOptionName+'.'+s+'PWS_N',
175                       'PWSs with '+s+' Exceedance PFOA-PFOS Under MCL', @CCounts[ix].PWS, mtAdhoc, mcSumOnly, False);
176     Metrics.AddMetric(fOptionName+'.'+s+'PWSPOP_N',
177                     'PWS Population Impacted by '+s+' Exceedance PFOA-PFOS Under MCL', @CCounts[ix].PWSPop, mtAdhoc, mcSumOnly, False);
178     Metrics.AddMetric(fOptionName+'.'+s+'EP_N',
179                       'EPs with '+s+' Exceedance PFOA-PFOS Under MCL', @CCounts[ix].EP, mtAdhoc, mcSumOnly, False);
180     Metrics.AddMetric(fOptionName+'.'+s+'EPPOP_N',
181                     'EPs Population Impacted by '+s+' Exceedance PFOA-PFOS Under MCL', @CCounts[ix].EPPop, mtAdhoc, mcSumOnly, False);
182
183     Metrics.AddMetric(fOptionName+'.'+s+'PWS_N2',
184                       'PWSs with '+s+' Exceedance PFOA-PFOS Over MCL', @OCounts[ix].PWS, mtAdhoc, mcSumOnly, False);
185     Metrics.AddMetric(fOptionName+'.'+s+'PWSPOP_N2',
186                     'PWS Population Impacted by '+s+' Exceedance PFOA-PFOS Over MCL', @OCounts[ix].PWSPop, mtAdhoc, mcSumOnly, False);
187     Metrics.AddMetric(fOptionName+'.'+s+'EP_N2',
188                       'EPs with '+s+' Exceedance PFOA-PFOS Over MCL', @OCounts[ix].EP, mtAdhoc, mcSumOnly, False);
189     Metrics.AddMetric(fOptionName+'.'+s+'EPPOP_N2',
190                     'EPs Population Impacted by '+s+' Exceedance PFOA-PFOS Over MCL', @OCounts[ix].EPPop, mtAdhoc, mcSumOnly, False);
191
192     Metrics.AddMetric(fOptionName+'.'+s+'AllConc',
193                     s+' Conc All EPs', @xConc[ix].All, mtAdhoc, mcMeanOnly, False);
194     Metrics.AddMetric(fOptionName+'.'+s+'AMclConc',
195                     s+' Conc EPs above MCL', @xConc[ix].GMCL, mtAdhoc, mcMeanOnly, True);
```

```
  end;

  Metrics.AddMetric(fOptionName+'.PFOAPFOSPWS',
                    'PWSs with Exceedance of PFOA or PFOS', @MCount.PWS, mtAdhoc, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFOAPFOSPWSPOP',
                 'PWS Population Impacted by Exceedance of PFOA or PFOS', @MCount.PWSPop, mtAdhoc, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFOAPFOSEP',
                    'EPs with Exceedance of PFOA or PFOS', @MCount.EP, mtAdhoc, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFOAPFOSEPPOP',
                    'EPs Population Impacted by Exceedance of PFOA or PFOS', @MCount.EPPop, mtAdhoc, mcSumOnly, False);


  Metrics.AddMetric(fOptionName+'.PFNAOGConc',
                 'PFNA Conc EPs above MCL with PFOA-PFOS Below MCL', @cgPFNA, mtAdhoc, mcMeanOnly, True);
  Metrics.AddMetric(fOptionName+'.PFHXSOGConc',
                 'PFHXS Conc EPs above MCL with PFOA-PFOS Below MCL', @cgPFHXS, mtAdhoc, mcMeanOnly, True);
  Metrics.AddMetric(fOptionName+'.HFPODAOGConc',
                 'HFPODA Conc EPs above MCL with PFOA-PFOS Below MCL', @cgHFPODA, mtAdhoc, mcMeanOnly, True);
end;

end.
```

code\CostModules\cmPFASOption1.pas

```
unit cmPFASOption1;

interface

uses Sysutils, Classes,SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterCostBase,
     SafeWaterUtility, SafewaterGlobals, SafeWaterTFObject, cmPFASBase, Math,
     SafewaterMetrics, Generics.Collections;

type
  TOCount=record
    PWS,EP, PWSPop,EPPop : double;
  end;
  TOCounts = array of TOCount;

  TcmPFASOption1 = class(TcmPFASBase)
    aFlag: boolean;
    b_CntAnnual: integer;
    ECounts, ACounts,NDCounts: TOCounts;
    ECountT, ACountT,NDCountT: TOCount;
    //late addition to metrics - just shimming in here....
    EPY1_PFE0, EPY1_ND, EPY1_E: double;

//TODO temp debug outputs remove
DidSmpT, DidSmpO : array of double;

    ixPFHXS : integer;
    constructor Create(aData : PSWModelData; GlbUnc: TGlobalUncertainty;  aNeededContams: string; aOptionName: string; aIteration: integer); override;
    procedure SetPWSWideEPValues(const P: TPWSRecord); override;
    procedure SetupModule; override;
    procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
  end;

implementation


{ TcmPFASOption1 }

procedure TcmPFASOption1.AddEntryPointCosts(const P: TPWSRecord;
  const Conc: TContamArray; var FinalConc: TContamArray);
begin
  inherited;
  var TotYears := FData.Cfg.EndYear - FData.Cfg.PFASChangeYear;
  var OnYrChange := FData.Cfg.PFASChangeYear - FData.Cfg.StartYear;

  b_source := 0;
  b_treat := 0;
```

```
48    b_exceed := 0;
49    b_nondetect := 1;
50    b_CntAnnual := 0;  //just  temp variable to handle annual monitoring counts - not otherwise used
51    var c : integer;
52    for c := 0 to CMax do begin
53      if (CurConc[c]>DL[c]) and (MCL[c]<1e30) then begin
54        NDCounts[c].PWS := 1;
55        NDCounts[c].PWSPop := P.Population;
56        NDCounts[c].EP := NDCounts[c].EP + 1;
57        NDCounts[c].EPPop := NDCounts[c].EPPop + CurPop;
58        if b_nondetect=1 then begin
59          NDCountT.PWS := 1;
60          NDCountT.PWSPop := P.Population;
61          NDCountT.EP := NDCountT.EP + 1;
62          NDCountT.EPPop := NDCountT.EPPop + CurPop;
63        end;
64        b_nondetect:=0;
65      end;
66      if CurConc[c] >= (MCL[c] * CMULT[c]) then begin
67        ECounts[c].PWS := 1;
68        ECounts[c].PWSPop := P.Population;
69        ECounts[c].EP := ECounts[c].EP + 1;
70        ECounts[c].EPPop := ECounts[c].EPPop + CurPop;
71        if b_exceed=0 then begin
72          ECountT.PWS := 1;
73          ECountT.PWSPop := P.Population;
74          ECountT.EP := ECountT.EP + 1;
75          ECountT.EPPop := ECountT.EPPop + CurPop;
76        end;
77        b_exceed:=1;
78      end;
79      if (CurConc[c] > MCL[c]/2) and (CurConc[c] < MCL[c]) and (MCL[c]<1e30) then begin
80        ACounts[c].PWS := 1;
81        ACounts[c].PWSPop := P.Population;
82        ACounts[c].EP := ACounts[c].EP + 1;
83        ACounts[c].EPPop := ACounts[c].EPPop + CurPop;
84        if b_CntAnnual=0 then begin
85          ACountT.PWS := 1;
86          ACountT.PWSPop := P.Population;
87          ACountT.EP := ACountT.EP + 1;
88          ACountT.EPPop := ACountT.EPPop + CurPop;
89        end;
90        b_CntAnnual:=1;
91      end;
92
93    end;
94    //adhoc counts for icr -  pop < 3300
95    if (P.StatePFAS=0) and (P.SystemSize < 4) then begin
96      EPY1_PFE0 := EPY1_PFE0 + 1;
97      if b_nondetect = 0 then EPY1_ND := EPY1_ND + 1;
98      if b_exceed = 1 then EPY1_E := EPY1_E + 1;
99    end;
100
101   {$ifdef DEBUG}
102   if _d then begin
103     FData.DebugLog.LD('costs','b_exceed', b_exceed);
104     FData.DebugLog.LD('costs','b_nondetect', b_nondetect);
105     FData.DebugLog.LD('costs','b_annual', b_annual);
106     FData.DebugLog.LD('costs','b_quarterly', b_quarterly);
107   end;
108   {$endif}
109
110   if (b_exceed <> 0) then begin
111     PerformanceGACIX;
112     var Tech := SelectTech(P);
113     if Tech < 0 then begin
114       FData.ErrLog.L(Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
115       exit;
116       //TODO what to do?
117     end;
118     TookAction := True;
119     GetWBSCosts(P, OnYrChange, TotYears, Tech);
120     //setting final concentrations based on Tech for this option
121     var tmp: double;
122     var ConcSet := false;
```

```
123     case Tech of
124       ttGAC  : begin
125                  b_treat := 1;
126                  for c := 0 to CMax do
127                    PR[c] := min((BV_GAC - A_TOC * cTOC - B_GAC[c]) / A_R, 0.99);
128                end;
129       ttIX   : begin
130                  b_treat := 1;
131                  for c := 0 to CMax do
132                    PR[c] := min((BV_IX - A_PFAS_IX * c0Total - B_IX[c]) / A_R_IX, 0.996);
133                end;
134       ttCRO  : begin
135                  b_treat := 1;
136                  tmp := min(PRM / 0.95,1); //B
137                  for c := 0 to CMax do
138                    if tmp = 1 then
139                      FinalConc[c] := (CurConc[c] * (1-PRM))
140                    else
141                      FinalConc[c] := (1-tmp) * CurConc[c] + tmp * (CurConc[c] * (1-0.95));
142                   concSet := true;
143                end;
144       ttPOU  : begin
145                  b_treat := 1;
146                  tmp := (c0Total - 70) / c0Total;
147                  if tmp<0 then tmp:=1;
148                  for c := 0 to CMax do
149                    PR[c] := tmp;
150                end;
151       else     begin
152                  b_source := 1;
153                  //remove everything for other technologies
154                  for c := 0 to CMax do
155                    PR[c] := 1;
156                end;
157     end;
158     if not ConcSet then begin
159       for c := 0 to CMax do
160         FinalConc[c] := CurConc[c] * (1 - max(PR[c],0));
161     end;
162
163     //make sure everything is under MCL
164     var OkMCL := true;
165     for c := 0 to CMax do begin
166        if FinalConc[c] > MCL[c] * CMult[c] then begin
167          OkMCL:=False;
168          break;
169        end;
170     end;
171
172     if not OkMCL then begin
173       FData.ErrLog.L('MCL Error:'+fOptionName);
174       FData.ErrLog.LD('tech',FData.TechForecast.TechToText(Tech));
175       FData.ErrLog.LD('df', Inf.DesignFlow);
176       FData.ErrLog.LD('af', Inf.AverageFlow);
177       FData.ErrLog.LD('toc', cTOC);
178       FData.ErrLog.LD('bv_gac', BV_GAC);
179       FData.ErrLog.LD('bv_ix', BV_IX);
180       for c := 0 to CMax do begin
181          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
182          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
183          FData.ErrLog.LD(' MCL',MCL[c]);
184          FData.ErrLog.LD(' PR',PR[c]);
185       end;
186       FData.ErrLog.L('*****');
187     end;
188
189     {$ifdef DEBUG}
190     if _d then begin
191       FData.DebugLog.L('costs','TECH');
192       FData.DebugLog.LD('costs','tech',FData.TechForecast.TechToText(Tech));
193       FData.DebugLog.LD('costs','df', Inf.DesignFlow);
194       FData.DebugLog.LD('costs','af', Inf.AverageFlow);
195       FData.DebugLog.LD('costs','toc', cTOC);
196       FData.DebugLog.LD('costs','bv_gac', BV_GAC);
```

```
197       FData.DebugLog.LD('costs','bv_ix', BV_IX);
198       for c := 0 to CMax do begin
199          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
200          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
201          FData.DebugLog.LD('costs','MCL',MCL[c]);
202          FData.DebugLog.LD('costs','PR',PR[c]);
203       end;
204     end;
205     {$endif}
206
207
208   end;  //b_exceed<>0
209
210   var OnYr: integer;
211   var aflag2 := false;
212   var DidSample: double := 0;
213   for var Yr := FData.Cfg.StartYear to FData.Cfg.EndYear do begin
214     OnYr := Yr  - FData.Cfg.StartYear;
215
216     if OnYr = 0 then begin   ///years 1,2,3
217       b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
218     end else
219     if OnYr = 3 then begin //3,4,5 years 4,5,6
220       b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
221       for c := 0 to CMax do begin
222         if (CurConc[c] > MCL[c]/2) then
223           b_quarterly := 1;
224       end;
225       if b_quarterly = 0 then begin  // all others - triennial
226          b_triennial := 1;
227       end;
228     end else
229     if OnYr = 6 then begin //year 7 - one year after treatment
230        //switch quarterly to annual in year 7 (all treating systems with get to between 1/2*mcl and mcl
231        if b_quarterly=1 then begin
232          b_quarterly := 0;
233          b_annual := 1;
234        end;
235     end;
236
237
238     if (FData.Cfg.SingleCostYear <> 0) and (Yr <> FData.Cfg.SingleCostYear) then continue;
239
240     if FreshPWS then
241       csOption._EvaluateSystem(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
242
243     csOption._Evaluate(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
244     {$ifdef DEBUG}
245     (*
246     if (aFlag = false) and (b_nondetect_all_ep=1) and (P.SystemSize=2) and
247        (P.SourceWater = swGroundwater) and (FIteration=1)  then begin
248
249       var TT := csOption._DumpCalcs;
250       FData.DebugLog.L('costmicro','***WORKBOOK BY YEAR: '+OnYr.ToString+'  PWS:'+P.PWSId+'  b_nondetect:'+b_nondetect.ToString   );
251       for var tti := 0 to TT.Count-1 do
252         FData.DebugLog.L('costmicro',TT[tti]);
253       TT.Free;
254       aFlag2:=True;
255
256     end;
257     *)
258     {$endif}
259
260     if _T[0].PWSSampling-DidSample >0 then begin
261       if b_triennial = 1 then
262         DidSmpT[OnYr] := DidSmpT[OnYr] + 1
263       else
264         DidSmpO[OnYr] := DidSmpO[OnYr] + 1
265     end;
266     DidSample := _T[0].PWSSampling;
267
268   end;
269   aFlag:=aFlag2;
270
271   FreshPWS := False;
```

```
272   //accumulate to the system for this entry point...
273   for var i := 0 to fData.Inp.RateCount-1 do begin
274     _i(O[i].AgencyImpAdmin,_T[i].AgencyImpAdmin);
275     _i(O[i].AgencyTreatmentAdmin, _T[i].AgencyTreatmentAdmin);
276     _i(O[i].PWSImpAdmin,_T[i].PWSImpAdmin);
277     _i(O[i].AgencyImpSampling,_T[i].AgencyImpSampling);
278     _i(O[i].PWSSampling,_T[i].PWSSampling);
279     _i(O[i].PWSImpAdmin_pws,_T[i].PWSImpAdmin_pws);
280     _i(O[i].PWSSampling_pws,_T[i].PWSSampling_pws);
281     _i(O[i].PWSTreatmentAdmin, _T[i].PWSTreatmentAdmin);
282     _i(O[i].PWSTreatmentAdmin_pws, _T[i].PWSTreatmentAdmin_pws);
283
284     _i(O[i].PWSTreatment,_T[i].PWSTreatment);
285     _i(O[i].PWSTreatment_pws,_T[i].PWSTreatment_pws);
286     _i(O[i].CapitalCost,_T[i].CapitalCost);
287     _i(O[i].CapitalCost_pws,_T[i].CapitalCost_pws);
288     _i(O[i].OMCost,_T[i].OMCost);
289     _i(O[i].OMCost_pws,_T[i].OMCost_pws);
290   end;
291 end;
292
293
294 constructor TcmPFASOption1.Create(aData: PSWModelData;
295   GlbUnc: TGlobalUncertainty; aNeededContams, aOptionName: string;
296   aIteration: integer);
297 begin
298   inherited;
299   ixPFOS := FData.Occ.ContamList.IndexOf('PFOS');
300   ixPFOA := FData.Occ.ContamList.IndexOf('PFOA');
301   ixPFHXS := FData.Occ.ContamList.IndexOf('PFHXS');
302   setlength(ECounts,FData.Occ.ContamList.Count);
303   setlength(ACounts,FData.Occ.ContamList.Count);
304   setlength(NDCounts,FData.Occ.ContamList.Count);
305   setlength(DidSmpT, FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
306   setlength(DidSmpO, FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
307 end;
308
309 procedure TcmPFASOption1.SetPWSWideEPValues(const P: TPWSRecord);
310 begin
311   inherited;
312   ECountT := Default(TOCOunt);
313   ACountT := Default(TOCOunt);
314   NDCountT := Default(TOCOunt);
315   EPY1_PFE0:=0;
316   EPY1_ND:=0;
317   EPY1_E:=0;
318
319   for var i := 0 to High(DidSmpT) do begin
320     DidSmpT[i] := 0;
321     DidSmpO[i] := 0;
322   end;
323
324   if ixPFOS>=0 then begin
325     ECounts[ixPFOS] := Default(TOCOunt);
326     ACounts[ixPFOS] := Default(TOCOunt);
327     NDCounts[ixPFOS] := Default(TOCOunt);
328   end;
329   if ixPFOA>=0 then begin
330     ECounts[ixPFOA] := Default(TOCOunt);
331     ACounts[ixPFOA] := Default(TOCOunt);
332     NDCounts[ixPFOA] := Default(TOCOunt);
333   end;
334   if ixPFHXS>=0 then begin
335     ECounts[ixPFHXS] := Default(TOCOunt);
336     ACounts[ixPFHXS] := Default(TOCOunt);
337     NDCounts[ixPFHXS] := Default(TOCOunt);
338   end;
339 end;
340
341 procedure TcmPFASOption1.SetupModule;
342 begin
343   inherited;
344   var ix := FData.Occ.ContamList.IndexOf('PFOS');
345   var ix2 := FData.Occ.ContamList.IndexOf('PFOA');
```

```
346   var ix3 := FData.Occ.ContamList.IndexOf('PFHXS');
347   MCLSafetyFactor := 0.8;
348
349   //TODO: get these from config
350   if CompareText(fOptionName,'PFASOption1a')=0 then begin
351     //pfos
352     if ix>-1 then MCL[ix] := 4.05;
353     //pfoa
354     if ix2>-1 then MCL[ix2] := 4.05;
355   end else
356   if CompareText(fOptionName,'PFASOption1b')=0 then begin
357     //pfos
358     if ix>-1 then MCL[ix] := 5.05;
359     //pfoa
360     if ix2>-1 then MCL[ix2] := 5.05;
361   end else
362   if CompareText(fOptionName,'PFASOption1c')=0 then begin
363     //pfos
364     if ix>-1 then MCL[ix] := 10.05;
365     //pfoa
366     if ix2>-1 then MCL[ix2] := 10.05;
367   end;
368   if CompareText(fOptionName,'PFASOption1d')=0 then begin
369     //pfos
370     if ix>-1 then MCL[ix] := 20.05;
371     //pfoa
372     if ix2>-1 then MCL[ix2] := 20.05;
373   end else
374   if CompareText(fOptionName,'PFASOption2')=0 then begin
375     //pfos
376     if ix>-1 then MCL[ix] := 4.05;
377     //pfoa
378     if ix2>-1 then MCL[ix2] := 4.05;
379     //pfhxs
380     if ix3>-1 then begin
381       MCL[ix3] :=10;
382       CMULT[ix3] := 1.499;
383       DL[ix3] := 1.4;
384     end;
385   end;
386
387
388   //for option 1, the MCLs are under 70ppt so POU is eliminated
389   NoPOU := true;
390 end;
391
392 procedure TcmPFASOption1.SetupOutputs(Metrics: TCategoryMetrics);
393 begin
394   inherited;
395
396   Metrics.AddMetric(fOptionName+'.PFASPWS',
397                     'PWSs with PFAS Exceedance (Quarterly Samples)', @ECountT.PWS, mtInfo, mcSumOnly, False);
398   Metrics.AddMetric(fOptionName+'.PFASPWSPOP',
399                     'PWS Population Impacted by PFAS Exceedance (Quarterly Samples)', @ECountT.PWSPop, mtInfo, mcSumOnly, False);
400   Metrics.AddMetric(fOptionName+'.PFASEP',
401                     'EPs with PFAS Exceedance (Quarterly Samples)', @ECountT.EP, mtInfo, mcSumOnly, False);
402   Metrics.AddMetric(fOptionName+'.PFASEPPOP',
403                     'EPs Population Impacted by PFAS Exceedance (Quarterly Samples)', @ECountT.EPPop, mtInfo, mcSumOnly, False);
404
405
406   Metrics.AddMetric(fOptionName+'.PFASPWS_A',
407                     'PWSs with PFAS Annual Sampling', @ACountT.PWS, mtInfo, mcSumOnly, False);
408   Metrics.AddMetric(fOptionName+'.PFASPWSPOP_A',
409                     'PWS Population Impacted by PFAS Annual Sampling', @ACountT.PWSPop, mtInfo, mcSumOnly, False);
410   Metrics.AddMetric(fOptionName+'.PFASEP_A',
411                     'EPs with PFAS Annual Sampling', @ACountT.EP, mtInfo, mcSumOnly, False);
412   Metrics.AddMetric(fOptionName+'.PFASEPPOP_A',
413                     'EPs Population Impacted by PFAS Annualy Sampling', @ACountT.EPPop, mtInfo, mcSumOnly, False);
414
415
416   Metrics.AddMetric(fOptionName+'.PFASPWS_ND',
417                     'PWSs with PFAS Detects', @NDCountT.PWS, mtInfo, mcSumOnly, False);
418   Metrics.AddMetric(fOptionName+'.PFASPWSPOP_ND',
419                     'PWS Population Impacted by PFAS Detects', @NDCountT.PWSPop, mtInfo, mcSumOnly, False);
420   Metrics.AddMetric(fOptionName+'.PFASEP_ND',
```

```
                    'EPs with PFAS Detects', @NDCountT.EP, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASEPPOP_ND',
                    'EPs Population Impacted by PFAS Detects', @NDCountT.EPPop, mtInfo, mcSumOnly, False);

  if ixPFOS>=0 then begin
    Metrics.AddMetric(fOptionName+'.PFOSPWS',
                      'PWSs with PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSPWSPOP',
                      'PWS Population Impacted by PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSEP',
                      'EPs with PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSEPPOP',
                      'EPs Population Impacted by PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFOSPWS_A',
                      'PWSs with PFOS Annual Sampling', @ACounts[ixPFOS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSPWSPOP_A',
                      'PWS Population Impacted by PFOS Annual Sampling', @ACounts[ixPFOS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSEP_A',
                      'EPs with PFOS Annual Sampling', @ACounts[ixPFOS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSEPPOP_A',
                      'EPs Population Impacted by PFOS Annual Sampling', @ACounts[ixPFOS].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFOSPWS_ND',
                      'PWSs with PFOS Detects', @NDCounts[ixPFOS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSPWSPOP_ND',
                      'PWS Population Impacted by PFOS Detects', @NDCounts[ixPFOS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSEP_ND',
                      'EPs with PFOS Detects', @NDCounts[ixPFOS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSEPPOP_ND',
                      'EPs Population Impacted by PFOS Detects', @NDCounts[ixPFOS].EPPop, mtInfo, mcSumOnly, False);

  end;
  if ixPFOA>=0 then begin
    Metrics.AddMetric(fOptionName+'.PFOAPWS',
                      'PWSs with PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAPWSPOP',
                      'PWS Population Impacted by PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEP',
                      'EPs with PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEPPOP',
                      'EPs Population Impacted by PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFOAPWS_A',
                      'PWSs with PFOA Annual Sampling', @ACounts[ixPFOA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAPWSPOP_A',
                      'PWS Population Impacted by PFOA Annual Sampling', @ACounts[ixPFOA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEP_A',
                      'EPs with PFOA Annual Sampling', @ACounts[ixPFOA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEPPOP_A',
                      'EPs Population Impacted by PFOA Annual Sampling', @ACounts[ixPFOA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFOAPWS_ND',
                      'PWSs with PFOA Detects', @NDCounts[ixPFOA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAPWSPOP_ND',
                      'PWS Population Impacted by PFOA Detects', @NDCounts[ixPFOA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEP_ND',
                      'EPs with PFOA Detects', @NDCounts[ixPFOA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEPPOP_ND',
                      'EPs Population Impacted by Detects', @NDCounts[ixPFOA].EPPop, mtInfo, mcSumOnly, False);

  end;
  if ixPFHXS>=0 then begin
    Metrics.AddMetric(fOptionName+'.PFHXSPWS',
                      'PWSs with PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSPWSPOP',
                      'PWS Population Impacted by PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEP',
                      'EPs with PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEPPOP',
                      'EPs Population Impacted by PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFHXSPWS_A',
                      'PWSs with PFHXS Annual Sampling', @ACounts[ixPFHXS].PWS, mtInfo, mcSumOnly, False);
```

```
    Metrics.AddMetric(fOptionName+'.PFHXSPWSPOP_A',
                      'PWS Population Impacted by PFHXS Annual Sampling', @ACounts[ixPFHXS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEP_A',
                      'EPs with PFHXS Exceedance Annual Sampling', @ACounts[ixPFHXS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEPPOP_A',
                      'EPs Population Impacted by PFHXS Annual Sampling', @ACounts[ixPFHXS].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFHXSPWS_ND',
                      'PWSs with PFHXS Detects', @NDCounts[ixPFHXS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSPWSPOP_ND',
                      'PWS Population Impacted by PFHXS Detects', @NDCounts[ixPFHXS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEP_ND',
                      'EPs with PFHXS Detects', @NDCounts[ixPFHXS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEPPOP_ND',
                      'EPs Population Impacted by PFHXS Detects', @NDCounts[ixPFHXS].EPPop, mtInfo, mcSumOnly, False);
  end;
  Metrics.AddMetric(fOptionName+'.EP_quarterly_yr1',
                      'All EPs in PWS pop < 3300 and with STATEPFAS = 0', @EPY1_PFE0, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EP_exceed_yr1',
                      'Number of EP in PWS pop < 3300 and with STATEPFAS = 0 above MCL for any contaminant in year 1', @EPY1_E, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EP_ nondetect_yr1',
                      'Number of EP in PWS pop < 3300 and with STATEPFAS = 0 above any contaminant detection level in year 1', @EPY1_ND, mtInfo, mcSumOnly, False);

(*
  var s: string;
  for var i := 0 to high(DidSmpT) do begin
    if i<9 then s := '0'+inttostr(i+1) else s:= inttostr(i+1);
    Metrics.AddMetric(fOptionName+'.NumEPWithSamplingCostY'+s+'_Tri',
                        'EPs Tri Sampling in year '+inttostr(i+1), @DidSmpT[i], mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.NumEPWithSamplingCostY'+s+'_Oth',
                        'EPs Other Sampling in year '+inttostr(i+1), @DidSmpO[i], mtInfo, mcSumOnly, False);

  end;
*)
end;

end.
```

code\CostModules\cmPFASOption2s.pas

```
unit cmPFASOption2s;

interface
uses Sysutils, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterCostBase,
     SafeWaterUtility, SafewaterGlobals, SafeWaterTFObject, cmPFASBase, Math,
      SafewaterMetrics;

type
  TcmPFASOption2s = class(TcmPFASBase)
    ixPFOS, ixPFOA, ixPFHXS, ixPFHPA,
    ixPFBS, ixPFNA, ixHFPODA : integer;

    HIC,HB: array of double;
    isHI, isMCL: array of boolean;

    HI0, RHI: double;
    PRg, PRi: array[0..20] of double;
    b_exceedHI: boolean;


    procedure PerformanceGACIX4;
    procedure PerformanceGACIX4_1;
    procedure FinalPctRemovals;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
    procedure SetPWSWideEPValues(const P: TPWSRecord); override;

    procedure SetupModule; override;
    procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
  end;

```

```
31 implementation
32
33
34 { TcmPFASOption2s }
35
36 procedure TcmPFASOption2s.AddEntryPointCosts(const P: TPWSRecord;
37   const Conc: TContamArray; var FinalConc: TContamArray);
38 begin
39   inherited;
40   var TotYears := FData.Cfg.EndYear - FData.Cfg.PFASChangeYear;
41   var OnYrChange := FData.Cfg.PFASChangeYear - FData.Cfg.StartYear;
42   var c : integer;
43
44   b_source := 0;
45   b_treat := 0;
46   b_exceed := 0;
47   b_nondetect := 1;
48   for c := 0 to CMax do begin
49     if (CurConc[c]>DL[c]) and (DL[c]>0) then b_nondetect:=0;   //change 11/2
50   end;
51
52   //pfhxs<3 - make 0 for HI calculation - EPA call 9/5/23
53   for var i := 0 to CMax do
54     if (i<>ixPFHXS) or (CurConc[i] >= 3) then
55       HIC[i] := CurConc[i] / HB[i]
56     else
57       HIC[i] := 0;
58
59   HI0:=0;
60   for var i := 0 to CMax do
61     if isHI[i] then
62       HI0 := HI0 + HIC[i];
63
64   RHI:=0;
65   if HI0>0 then
66     RHI := (HI0 - (1 * MCLSafetyFactor)) / HI0;
67   b_exceedHI := HI0>1.499;
68
69   var Tech: integer;
70   if HI0>100 then begin  //?? guard against too much removal?
71     b_exceed := 1;
72     tFV[ttGAC] := false;
73     tFV[ttIX] := false;
74     RedistributeTechForecast;
75     Tech := SelectTech(P);
76   end else begin
77     PerformanceGACIX4;
78     if b_exceed<>0 then
79       Tech := SelectTech(P);
80   end;
81
82   {$ifdef DEBUG}
83   if _d then begin
84     FData.DebugLog.LD('costs','b_exceed', b_exceed);
85     FData.DebugLog.LD('costs','b_nondetect', b_nondetect);
86   end;
87   {$endif}
88
89   if (b_exceed <> 0) then begin
90
91     if Tech < 0 then begin
92       FData.ErrLog.L(Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
93       exit;
94       //TODO what to do?
95     end;
96
97     TookAction := True;
98     //set PRM for CRO cost...
99     var tPA := 0.0;
100    var tPS := 0.0;
101    if CurConc[ixPFOA]>0 then tPA := (CurConc[ixPFOA] - MCL[ixPFOA] * MCLSafetyFactor) / CurConc[ixPFOA];
102    if CurConc[ixPFOS]>0 then tPS := (CurConc[ixPFOS] - MCL[ixPFOS] * MCLSafetyFactor) / CurConc[ixPFOS];
103    PRM := max(RHI,max(tPA,tPS));
104
105    GetWBSCosts(P, OnYrChange, TotYears, Tech);
```

```
106
107     //setting final concentrations based on Tech for this option
108     var tmp: double;
109     var ConcSet := false;
110     FinalPctRemovals();
111     case Tech of
112       ttGAC  : begin
113                  b_treat := 1;
114                  for c := 0 to CMax do
115                    PR[c] := min(PRg[c], 0.99);
116                end;
117       ttIX   : begin
118                  b_treat := 1;
119                  for c := 0 to CMax do
120                    PR[c] := min(PRi[c], 0.996);
121                end;
122       ttCRO  : begin
123                  b_treat := 1;
124                  for c := 0 to CMax do begin
125                    PR[c] := PRM;
126                  end;
127                end;
128       ttPOU  : begin
129                  b_treat := 1;
130                  tmp := (c0Total - 70) / c0Total;
131                  if tmp<0 then tmp:=1;
132                  for c := 0 to CMax do
133                    PR[c] := tmp;
134                end;
135       else     begin
136                  b_source := 1;
137                  //remove everything for other technologies
138                  for c := 0 to CMax do
139                    PR[c] := 1;
140                end;
141     end;
142     if not ConcSet then begin
143       for c := 0 to CMax do
144         FinalConc[c] := CurConc[c] * (1 - max(PR[c],0));
145     end;
146
147     //make sure everything is under MCL
148     var OkMCL := true;
149     var tHIC: double := 0;
150     for c := 0 to CMax do begin
151       if FinalConc[c] > MCL[c] * CMult[c] then begin
152         OkMCL:=False;
153         break;
154       end;
155       (* sufficient to know it exceeds.  BV takes care of cost - no ben for 4 pollutants in HI
156       if isHI[c] then
157        if (c<>ixPFHXS) or (CurConc[c] >= 3) then
158          tHIC := tHIC + FinalConc[c] / HB[c];
159
160       if tHIC > 1.5 then begin
161         OkMCL:=False;
162         break;
163       end;
164       *)
165     end;
166
167     if not OkMCL then begin
168       FData.ErrLog.L('MCL Error:'+fOptionName);
169       FData.ErrLog.LD('tech',FData.TechForecast.TechToText(Tech));
170       FData.ErrLog.LD('df', Inf.DesignFlow);
171       FData.ErrLog.LD('af', Inf.AverageFlow);
172       FData.ErrLog.LD('toc', cTOC);
173       FData.ErrLog.LD('bv_gac', BV_GAC);
174       FData.ErrLog.LD('bv_ix', BV_IX);
175       FData.ErrLog.LD('HI', HI0);
176       for c := 0 to CMax do begin
177          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
178          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
179          FData.ErrLog.LD(' MCL',MCL[c]);
```

```
180          FData.ErrLog.LD(' PR',PR[c]);
181          FData.ErrLog.LD(' HB',HB[c]);
182        end;
183      end;
184
185      {$ifdef DEBUG}
186      if _d then begin
187        FData.DebugLog.L('costs','TECH');
188        FData.DebugLog.LD('costs','tech',FData.TechForecast.TechToText(Tech));
189        FData.DebugLog.LD('costs','df', Inf.DesignFlow);
190        FData.DebugLog.LD('costs','af', Inf.AverageFlow);
191        FData.DebugLog.LD('costs','toc', cTOC);
192        FData.DebugLog.LD('costs','bv_gac', BV_GAC);
193        FData.DebugLog.LD('costs','bv_ix', BV_IX);
194        FData.DebugLog.LD('costs','PRM', PRM);
195        for c := 0 to CMax do begin
196           FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
197           FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
198           FData.DebugLog.LD('costs','MCL',MCL[c]);
199           FData.DebugLog.LD('costs','PR',PR[c]);
200        end;
201      end;
202      {$endif}
203
204    end;  //b_exceed<>0
205
206
207    b_quarterly := 1;  //quarterly until imp year..
208    b_annual := 0;
209    b_triennial := 0;
210    var OnYr: integer;
211    for var Yr := FData.Cfg.StartYear to FData.Cfg.EndYear do begin
212      OnYr := Yr  - FData.Cfg.StartYear;
213
214      if OnYr = 0 then begin
215        b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
216      end else
217      if OnYr = 3 then begin
218        b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
219
220        if (CurConc[ixPFOA] > MCL[ixPFOA]/2) or (CurConc[ixPFOS] > MCL[ixPFOS]/2) or
221           (HI0>1/2) then
222            b_quarterly := 1
223        else
224          b_triennial := 1;
225      end else
226      if OnYr = 6 then begin // year 7 - one year after treatment
227         //switch quaterly to annual in year 6 (all treating systems with get to between 1/2*mcl and mcl
228         if b_quarterly=1 then begin
229           b_quarterly := 0;
230           b_annual := 1;
231         end;
232      end;
233
234      if FreshPWS then
235        csOption._EvaluateSystem(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
236
237      csOption._Evaluate(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
238      {$ifdef DEBUG}
239      if _d then begin
240         (*
241        var TT := csOption._DumpCalcs;
242        FData.DebugLog.L('***WORKBOOK BY YEAR: '+OnYr.ToString);
243        for var tti := 0 to TT.Count-1 do
244          FData.DebugLog.L(TT[tti]);
245        TT.Free;
246        *)
247      end;
248      {$endif}
249      inc(OnYr);
250    end;
251    FreshPWS := False;
252    //accumulate to the system for this entry point...
253    for var i := 0 to fData.Inp.RateCount-1 do begin
254      _i(O[i].AgencyImpAdmin,_T[i].AgencyImpAdmin);
```

```
255     _i(O[i].AgencyTreatmentAdmin, _T[i].AgencyTreatmentAdmin);
256     _i(O[i].PWSImpAdmin,_T[i].PWSImpAdmin);
257     _i(O[i].AgencyImpSampling,_T[i].AgencyImpSampling);
258     _i(O[i].PWSSampling,_T[i].PWSSampling);
259     _i(O[i].PWSImpAdmin_pws,_T[i].PWSImpAdmin_pws);
260     _i(O[i].PWSSampling_pws,_T[i].PWSSampling_pws);
261     _i(O[i].PWSTreatmentAdmin, _T[i].PWSTreatmentAdmin);
262     _i(O[i].PWSTreatmentAdmin_pws, _T[i].PWSTreatmentAdmin_pws);
263
264     _i(O[i].PWSTreatment,_T[i].PWSTreatment);
265     _i(O[i].PWSTreatment_pws,_T[i].PWSTreatment_pws);
266     _i(O[i].CapitalCost,_T[i].CapitalCost);
267     _i(O[i].CapitalCost_pws,_T[i].CapitalCost_pws);
268     _i(O[i].OMCost,_T[i].OMCost);
269     _i(O[i].OMCost_pws,_T[i].OMCost_pws);
270   end;
271 end;
272
273 procedure TcmPFASOption2s.FinalPctRemovals;
274 begin
275   //A_PFAS_IX * C0Total + A_R_IX * PR[i] + B_IX[i]
276   //A_TOC * cTOC + A_R * PR[i] + B_GAC[i]
277   for var c := 0  to CMax do begin
278     PRg[c] := max(0, (BV_GAC - A_TOC * cTOC - B_GAC[c]) / A_R);
279     PRi[c] := max(0, (BV_IX - A_PFAS_IX * C0Total - B_IX[c]) / A_R_IX);
280   end;
281 end;
282
283 procedure TcmPFASOption2s.PerformanceGACIX4;
284 var MRT : array[0..20,0..20] of double;
285 begin
286   var
287   bvgPFNA, bvgPFBS, bvgHFPODA, bvgPFHXS,
288   bviPFNA, bviPFBS, bviHFPODA, bviPFHXS,
289   C0T
290   : double;
291 //csl('IN');
292   var c,t: integer;
293   var HISF:double := 1 * MCLSafetyFactor;
294
295   if b_exceedHI then begin
296     b_exceed := 1;
297     //set terms for GAC
298     for c := 0  to CMax do begin
299       PRg[c] := 0;
300       if not isHI[c] then continue;
301       for t := 0  to CMax do begin
302         if not isHI[t] then continue;
303         if c=t then continue;
304         MRT[c,t] := (B_GAC[t] - B_GAC[c]) / A_R;
305       end;
306     end;
307
308     var subcnt:integer;
309     var sub := false;
310     C0t := HI0;
311     subcnt:=0;
312
313     for subcnt:=0 to CMax do begin
314       for c := 0  to CMax do begin
315         if PRg[c] = 0.99 then continue;
316         if not isHI[c] then continue;
317         PRg[c] := HIC[c];
318         for t := 0  to CMax do begin
319           if c=t then continue;
320           if not isHI[t] then continue;
321   //csl('c:'+c.ToString+'  t:'+t.ToString+ ' pr1:'+PRg[c].ToString+'  mrt:'+MRT[c,t].ToString);
322           PRg[c] := PRg[c] + HIC[t] * (1 + MRT[c,t]);
323   //csl('c:'+c.ToString+'  t:'+t.ToString+' pr2:'+PRg[c].ToString);
324         end;
325         PRg[c] := (PRg[c] - HISF) / HI0;
326       end;
327
328   (*
```

```
329   csl(subcnt.ToString);
330   csl('c0t:'+c0t.ToString);
331   for c := 0  to CMax do begin
332     for t := 0  to CMax do begin
333       csl('c:'+c.ToString+'   t:'+t.ToString+'   m:'+MRT[c,t].ToString);
334     end;
335   end;
336   for c := 0  to CMax do begin
337   csl('c:'+c.ToString+'  p:'+PRg[c].ToString);
338   end;
339   *)
340       sub := false;
341       for c := 0  to CMax do begin
342         if not isHI[c] then continue;
343         if PRg[c] > 0.99 then begin
344           C0T := C0T - HIC[c];
345           PRg[c] := 0.99;
346           sub:=true;
347           for t := 0  to CMax do begin
348             if not isHI[t] then continue;
349             MRT[t,c] := -0.99;
350           end;
351         end;
352       end;
353       if not sub then break;
354     end;
355
356     //set terms for IX
357     for c := 0  to CMax do begin
358       PRi[c] := 0;
359       if not isHI[c] then continue;
360       for t := 0  to CMax do begin
361         if c=t then continue;
362         if not isHI[t] then continue;
363         MRT[c,t] := (B_IX[t] - B_IX[c]) / A_R_IX;
364       end;
365     end;
366     C0t := HI0;
367     subcnt:=0;
368     for subcnt:=0 to CMax do begin
369       for c := 0  to CMax do begin
370         if not isHI[c] then continue;
371         if PRi[c] = 0.99 then continue;
372         PRi[c] := HIC[c];
373         for t := 0  to CMax do begin
374           if c=t then continue;
375           if not isHI[t] then continue;
376           PRi[c] := PRi[c] + HIC[t] * (1 + MRT[c,t]);
377         end;
378         PRi[c] := (PRi[c] - HISF) / HI0;
379       end;
380
381       sub := false;
382       for c := 0  to CMax do begin
383         if not isHI[c] then continue;
384         if PRi[c] > 0.99 then begin
385           C0T := C0T - HIC[c];
386           PRi[c] := 0.99;
387           sub:=true;
388           for t := 0  to CMax do begin
389             if not isHI[t] then continue;
390             MRT[t,c] := -0.99;
391           end;
392         end;
393       end;
394       if not sub then break;
395     end;
396   end; //b_exceedHI
397
398   //Check Option 1...
399   PerformanceGACIX4_1;
400
401   if b_exceed=0 then exit;
402
403   //Bed volumes
```

```
404   if b_exceedHI then begin
405     bvgPFNA := A_TOC * cTOC + A_R * PRg[ixPFNA] + B_GAC[ixPFNA];
406     bvgHFPODA := A_TOC * cTOC + A_R * PRg[ixHFPODA] + B_GAC[ixHFPODA];
407     bvgPFBS := A_TOC * cTOC + A_R * PRg[ixPFBS] + B_GAC[ixPFBS];
408     bvgPFHXS := A_TOC * cTOC + A_R * PRg[ixPFHXS] + B_GAC[ixPFHXS];
409     BV_GAC := min(BV_GAC, min(min(bvgPFNA, bvgPFBS), min(bvgHFPODA, bvgPFHXS)));
410   end;
411   if BV_GAC < 5000 then tFV[ttGAC] := false;
412
413   if b_exceedHI then begin
414     bviPFNA := A_PFAS_IX + A_R_IX * PRi[ixPFNA] + B_IX[ixPFNA];
415     bviHFPODA := A_PFAS_IX + A_R_IX * PRi[ixHFPODA] + B_IX[ixHFPODA];
416     bviPFBS := A_PFAS_IX + A_R_IX * PRi[ixPFBS] + B_IX[ixPFBS];
417     bviPFHXS := A_PFAS_IX + A_R_IX * PRi[ixPFHXS] + B_IX[ixPFHXS];
418     BV_IX := min(BV_IX, min(min(bviPFNA, bviPFBS), min(bviHFPODA, bviPFHXS)));
419   end;
420   if BV_IX < 20000 then tFV[ttIX] := false;
421
422   RedistributeTechForecast;
423 end;
424
425 function _gpr(const c0,mcl,sf: double; var PRM: double) : double;
426 begin
427   if (c0>0) and (mcl<1e30) and (c0>mcl) then
428     Result := (c0 - mcl * sf) / c0
429   else
430     Result := 0;
431   if Result > PRM then PRM := Result;
432 end;
433
434 procedure TcmPFASOption2s.PerformanceGACIX4_1;
435 begin
436   var i: integer;
437   PRM := 0;
438   //TODO should these be set to large numbers in the absence of MCLs
439   for i := 0 to CMax do
440      PR[i] := _gpr(CurConc[i], MCL[i], MCLSafetyFactor,PRM);
441
442   BV_GAC := 1e30;
443   BV_IX := 1e30;
444   if (PRM=0) then begin
445     exit;
446   end;
447   b_exceed := 1;
448
449   if PRM > 0.99 then begin
450     tFV[ttGAC] := false;
451     if PRM > 0.996 then
452       tFV[ttIX] := false;
453   end;
454
455   if tFV[ttGAC] then
456     for i := 0 to CMax do
457       if PR[i]>0 then
458         BV_GAC := min(BV_GAC,A_TOC * cTOC + A_R * PR[i] + B_GAC[i]);
459
460   if tFV[ttIX] then
461     for i := 0 to CMax do
462       if PR[i]>0 then
463         BV_IX := min(BV_IX, A_PFAS_IX * C0Total + A_R_IX * PR[i] + B_IX[i]);
464 end;
465
466 procedure TcmPFASOption2s.SetPWSWideEPValues(const P: TPWSRecord);
467 begin
468   inherited;
469 end;
470
471 procedure TcmPFASOption2s.SetupModule;
472 begin
473   inherited;
474   setlength(HIC,CMax+1);
475   setlength(isHI,CMax+1);
476   setlength(isMCL,CMax+1);
477   setlength(HB,CMax+1);
```

```
  for var i := 0 to CMax do begin
    HB[i] := 1e99;
    isMCL[i] := false;
    isHI[i] := false;
  end;

  ixPFOS := FData.Occ.ContamList.IndexOf('PFOS');
  ixPFOA := FData.Occ.ContamList.IndexOf('PFOA');
  ixPFHXS := FData.Occ.ContamList.IndexOf('PFHXS');
  ixPFHPA := FData.Occ.ContamList.IndexOf('PFHPA');
  ixPFBS := FData.Occ.ContamList.IndexOf('PFBS');
  ixPFNA := FData.Occ.ContamList.IndexOf('PFNA');
  ixHFPODA := FData.Occ.ContamList.IndexOf('HFPODA');

  isHI[ixPFBS]:=true;
  isHI[ixPFNA]:=true;
  isHI[ixHFPODA]:=true;
  isHI[ixPFHXS]:=true;

  isMCL[ixPFOA]:=true;
  isMCL[ixPFOS]:=true;

  HB[ixPFBS]:=2000;
  HB[ixPFNA]:=10;
  HB[ixHFPODA]:=10;
  HB[ixPFHXS]:=10;

  MCLSafetyFactor := 0.8;

  MCL[ixPFOS] := 4.05;
  MCL[ixPFOA] := 4.05;

  //for option 2, the MCLs are under 70ppt so POU is eliminated
  NoPOU := true;
end;

procedure TcmPFASOption2s.SetupOutputs(Metrics: TCategoryMetrics);
begin
  inherited;
end;

end.


end.

```

code\CostModules\cmPFASOption2x.pas

```
unit cmPFASOption2x;

interface

uses Sysutils, Classes,SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterCostBase,
     SafeWaterUtility, SafewaterGlobals, SafeWaterTFObject, cmPFASBase, Math,
     SafewaterMetrics, Generics.Collections;

//Note: unit copied from option1 and modified to do analysis of single MCLs for HI contaminants
type

  TOCount=record
    PWS,EP, PWSPop,EPPop : double;
  end;
  TOCounts = array of TOCount;

  TcmPFASOption2x = class(TcmPFASBase)
    aFlag: boolean;
    b_CntAnnual: integer;
    ECounts, ACounts,NDCounts: TOCounts;
    ECountT, ACountT,NDCountT: TOCount;
    //late addition to metrics - just shimming in here....
    EPY1_PFE0, EPY1_ND, EPY1_E: double;

```

```
25  //TODO temp debug outputs remove
26  DidSmpT, DidSmpO : array of double;
27
28      ixPFHXS,ixPFNA,ixHFPODA   : integer;
29      constructor Create(aData : PSWModelData; GlbUnc: TGlobalUncertainty;  aNeededContams: string; aOptionName: string; aIteration: integer); override;
30      procedure SetPWSWideEPValues(const P: TPWSRecord); override;
31      procedure SetupModule; override;
32      procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
33      procedure SetupOutputs(Metrics: TCategoryMetrics); override;
34    end;
35
36  implementation
37
38
39  { TcmPFASOption2x }
40
41  procedure TcmPFASOption2x.AddEntryPointCosts(const P: TPWSRecord;
42    const Conc: TContamArray; var FinalConc: TContamArray);
43  begin
44    inherited;
45    var TotYears := FData.Cfg.EndYear - FData.Cfg.PFASChangeYear;
46    var OnYrChange := FData.Cfg.PFASChangeYear - FData.Cfg.StartYear;
47
48    b_source := 0;
49    b_treat := 0;
50    b_exceed := 0;
51    b_nondetect := 1;
52    b_CntAnnual := 0;  //just  temp variable to handle annual monitoring counts - not otherwise used
53    var c : integer;
54    for c := 0 to CMax do begin
55      if (CurConc[c]>DL[c]) and (MCL[c]<1e30) then begin
56        NDCounts[c].PWS := 1;
57        NDCounts[c].PWSPop := P.Population;
58        NDCounts[c].EP := NDCounts[c].EP + 1;
59        NDCounts[c].EPPop := NDCounts[c].EPPop + CurPop;
60        if b_nondetect=1 then begin
61          NDCountT.PWS := 1;
62          NDCountT.PWSPop := P.Population;
63          NDCountT.EP := NDCountT.EP + 1;
64          NDCountT.EPPop := NDCountT.EPPop + CurPop;
65        end;
66        b_nondetect:=0;
67      end;
68      if CurConc[c] >= (MCL[c] * CMULT[c]) then begin
69        ECounts[c].PWS := 1;
70        ECounts[c].PWSPop := P.Population;
71        ECounts[c].EP := ECounts[c].EP + 1;
72        ECounts[c].EPPop := ECounts[c].EPPop + CurPop;
73        if b_exceed=0 then begin
74          ECountT.PWS := 1;
75          ECountT.PWSPop := P.Population;
76          ECountT.EP := ECountT.EP + 1;
77          ECountT.EPPop := ECountT.EPPop + CurPop;
78        end;
79        b_exceed:=1;
80      end;
81      if (CurConc[c] > MCL[c]/2) and (CurConc[c] < MCL[c]) and (MCL[c]<1e30) then begin
82        ACounts[c].PWS := 1;
83        ACounts[c].PWSPop := P.Population;
84        ACounts[c].EP := ACounts[c].EP + 1;
85        ACounts[c].EPPop := ACounts[c].EPPop + CurPop;
86        if b_CntAnnual=0 then begin
87          ACountT.PWS := 1;
88          ACountT.PWSPop := P.Population;
89          ACountT.EP := ACountT.EP + 1;
90          ACountT.EPPop := ACountT.EPPop + CurPop;
91        end;
92        b_CntAnnual:=1;
93      end;
94
95    end;
96    //adhoc counts for icr -  pop < 3300
97    if (P.StatePFAS=0) and (P.SystemSize < 4) then begin
98      EPY1_PFE0 := EPY1_PFE0 + 1;
99      if b_nondetect = 0 then EPY1_ND := EPY1_ND + 1;
```

```
100     if b_exceed = 1 then EPY1_E := EPY1_E + 1;
101   end;
102
103   {$ifdef DEBUG}
104   if _d then begin
105     FData.DebugLog.LD('costs','b_exceed', b_exceed);
106     FData.DebugLog.LD('costs','b_nondetect', b_nondetect);
107     FData.DebugLog.LD('costs','b_annual', b_annual);
108     FData.DebugLog.LD('costs','b_quarterly', b_quarterly);
109   end;
110   {$endif}
111
112 Exit;
113
114   if (b_exceed <> 0) then begin
115     PerformanceGACIX;
116     var Tech := SelectTech(P);
117     if Tech < 0 then begin
118       FData.ErrLog.L(Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
119       exit;
120       //TODO what to do?
121     end;
122     TookAction := True;
123     GetWBSCosts(P, OnYrChange, TotYears, Tech);
124     //setting final concentrations based on Tech for this option
125     var tmp: double;
126     var ConcSet := false;
127     case Tech of
128       ttGAC  : begin
129                  b_treat := 1;
130                  for c := 0 to CMax do
131                    PR[c] := min((BV_GAC - A_TOC * cTOC - B_GAC[c]) / A_R, 0.99);
132                end;
133       ttIX   : begin
134                  b_treat := 1;
135                  for c := 0 to CMax do
136                    PR[c] := min((BV_IX - A_PFAS_IX * c0Total - B_IX[c]) / A_R_IX, 0.996);
137                end;
138       ttCRO  : begin
139                  b_treat := 1;
140                  tmp := min(PRM / 0.95,1); //B
141                  for c := 0 to CMax do
142                    if tmp = 1 then
143                      FinalConc[c] := (CurConc[c] * (1-PRM))
144                    else
145                      FinalConc[c] := (1-tmp) * CurConc[c] + tmp * (CurConc[c] * (1-0.95));
146                  ConcSet := true;
147                end;
148       ttPOU  : begin
149                  b_treat := 1;
150                  tmp := (c0Total - 70) / c0Total;
151                  if tmp<0 then tmp:=1;
152                  for c := 0 to CMax do
153                    PR[c] := tmp;
154                end;
155       else     begin
156                  b_source := 1;
157                  //remove everything for other technologies
158                  for c := 0 to CMax do
159                    PR[c] := 1;
160                end;
161     end;
162     if not ConcSet then begin
163       for c := 0 to CMax do
164         FinalConc[c] := CurConc[c] * (1 - max(PR[c],0));
165     end;
166
167     //make sure everything is under MCL
168     var OkMCL := true;
169     for c := 0 to CMax do begin
170        if FinalConc[c] > MCL[c] * CMult[c] then begin
171          OkMCL:=False;
172          break;
173        end;
```

```
174     end;
175
176     if not OkMCL then begin
177       FData.ErrLog.L('MCL Error:'+fOptionName);
178       FData.ErrLog.LD('tech',FData.TechForecast.TechToText(Tech));
179       FData.ErrLog.LD('df', Inf.DesignFlow);
180       FData.ErrLog.LD('af', Inf.AverageFlow);
181       FData.ErrLog.LD('toc', cTOC);
182       FData.ErrLog.LD('bv_gac', BV_GAC);
183       FData.ErrLog.LD('bv_ix', BV_IX);
184       for c := 0 to CMax do begin
185          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
186          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
187          FData.ErrLog.LD(' MCL',MCL[c]);
188          FData.ErrLog.LD(' PR',PR[c]);
189       end;
190       FData.ErrLog.L('*****');
191     end;
192
193     {$ifdef DEBUG}
194     if _d then begin
195       FData.DebugLog.L('costs','TECH');
196       FData.DebugLog.LD('costs','tech',FData.TechForecast.TechToText(Tech));
197       FData.DebugLog.LD('costs','df', Inf.DesignFlow);
198       FData.DebugLog.LD('costs','af', Inf.AverageFlow);
199       FData.DebugLog.LD('costs','toc', cTOC);
200       FData.DebugLog.LD('costs','bv_gac', BV_GAC);
201       FData.DebugLog.LD('costs','bv_ix', BV_IX);
202       for c := 0 to CMax do begin
203          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
204          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
205          FData.DebugLog.LD('costs','MCL',MCL[c]);
206          FData.DebugLog.LD('costs','PR',PR[c]);
207       end;
208     end;
209     {$endif}
210
211
212   end;  //b_exceed<>0
213
214   var OnYr: integer;
215   var aflag2 := false;
216   var DidSample: double := 0;
217   for var Yr := FData.Cfg.StartYear to FData.Cfg.EndYear do begin
218     OnYr := Yr  - FData.Cfg.StartYear;
219
220     if OnYr = 0 then begin   ///years 1,2,3
221       b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
222     end else
223     if OnYr = 3 then begin //3,4,5 years 4,5,6
224       b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
225       for c := 0 to CMax do begin
226         if (CurConc[c] > MCL[c]/2) then
227           b_quarterly := 1;
228       end;
229       if b_quarterly = 0 then begin  // all others - triennial
230          b_triennial := 1;
231       end;
232     end else
233     if OnYr = 6 then begin //year 7 - one year after treatment
234        //switch quarterly to annual in year 7 (all treating systems with get to between 1/2*mcl and mcl
235        if b_quarterly=1 then begin
236          b_quarterly := 0;
237          b_annual := 1;
238        end;
239     end;
240
241
242     if (FData.Cfg.SingleCostYear <> 0) and (Yr <> FData.Cfg.SingleCostYear) then continue;
243
244     if FreshPWS then
245       csOption._EvaluateSystem(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
246
247     csOption._Evaluate(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
248     {$ifdef DEBUG}
```

```
249     (*
250     if (aFlag = false) and (b_nondetect_all_ep=1) and (P.SystemSize=2) and
251        (P.SourceWater = swGroundwater) and (FIteration=1)  then begin
252
253       var TT := csOption._DumpCalcs;
254       FData.DebugLog.L('costmicro','***WORKBOOK BY YEAR: '+OnYr.ToString+'  PWS:'+P.PWSId+'  b_nondetect:'+b_nondetect.ToString   );
255       for var tti := 0 to TT.Count-1 do
256         FData.DebugLog.L('costmicro',TT[tti]);
257       TT.Free;
258       aFlag2:=True;
259
260     end;
261     *)
262     {$endif}
263
264     if _T[0].PWSSampling-DidSample >0 then begin
265       if b_triennial = 1 then
266         DidSmpT[OnYr] := DidSmpT[OnYr] + 1
267       else
268         DidSmpO[OnYr] := DidSmpO[OnYr] + 1
269     end;
270     DidSample := _T[0].PWSSampling;
271
272   end;
273   aFlag:=aFlag2;
274
275   FreshPWS := False;
276   //accumulate to the system for this entry point...
277   for var i := 0 to fData.Inp.RateCount-1 do begin
278     _i(O[i].AgencyImpAdmin,_T[i].AgencyImpAdmin);
279     _i(O[i].AgencyTreatmentAdmin, _T[i].AgencyTreatmentAdmin);
280     _i(O[i].PWSImpAdmin,_T[i].PWSImpAdmin);
281     _i(O[i].AgencyImpSampling,_T[i].AgencyImpSampling);
282     _i(O[i].PWSSampling,_T[i].PWSSampling);
283     _i(O[i].PWSImpAdmin_pws,_T[i].PWSImpAdmin_pws);
284     _i(O[i].PWSSampling_pws,_T[i].PWSSampling_pws);
285     _i(O[i].PWSTreatmentAdmin, _T[i].PWSTreatmentAdmin);
286     _i(O[i].PWSTreatmentAdmin_pws, _T[i].PWSTreatmentAdmin_pws);
287
288     _i(O[i].PWSTreatment,_T[i].PWSTreatment);
289     _i(O[i].PWSTreatment_pws,_T[i].PWSTreatment_pws);
290     _i(O[i].CapitalCost,_T[i].CapitalCost);
291     _i(O[i].CapitalCost_pws,_T[i].CapitalCost_pws);
292     _i(O[i].OMCost,_T[i].OMCost);
293     _i(O[i].OMCost_pws,_T[i].OMCost_pws);
294   end;
295 end;
296
297
298 constructor TcmPFASOption2x.Create(aData: PSWModelData;
299   GlbUnc: TGlobalUncertainty; aNeededContams, aOptionName: string;
300   aIteration: integer);
301 begin
302   inherited;
303   ixPFOS := FData.Occ.ContamList.IndexOf('PFOS');
304   ixPFOA := FData.Occ.ContamList.IndexOf('PFOA');
305   ixPFHXS := FData.Occ.ContamList.IndexOf('PFHXS');
306   ixHFPODA := FData.Occ.ContamList.IndexOf('HFPODA');
307   ixPFNA := FData.Occ.ContamList.IndexOf('PFNA');
308   setlength(ECounts,FData.Occ.ContamList.Count);
309   setlength(ACounts,FData.Occ.ContamList.Count);
310   setlength(NDCounts,FData.Occ.ContamList.Count);
311   setlength(DidSmpT, FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
312   setlength(DidSmpO, FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
313 end;
314
315 procedure TcmPFASOption2x.SetPWSWideEPValues(const P: TPWSRecord);
316 begin
317   inherited;
318   ECountT := Default(TOCOunt);
319   ACountT := Default(TOCOunt);
320   NDCountT := Default(TOCOunt);
321   EPY1_PFE0:=0;
322   EPY1_ND:=0;
```

```
323   EPY1_E:=0;
324
325   for var i := 0 to High(DidSmpT) do begin
326     DidSmpT[i] := 0;
327     DidSmpO[i] := 0;
328   end;
329
330   if ixPFOS>=0 then begin
331     ECounts[ixPFOS] := Default(TOCOunt);
332     ACounts[ixPFOS] := Default(TOCOunt);
333     NDCounts[ixPFOS] := Default(TOCOunt);
334   end;
335   if ixPFOA>=0 then begin
336     ECounts[ixPFOA] := Default(TOCOunt);
337     ACounts[ixPFOA] := Default(TOCOunt);
338     NDCounts[ixPFOA] := Default(TOCOunt);
339   end;
340   if ixPFHXS>=0 then begin
341     ECounts[ixPFHXS] := Default(TOCOunt);
342     ACounts[ixPFHXS] := Default(TOCOunt);
343     NDCounts[ixPFHXS] := Default(TOCOunt);
344   end;
345   if ixHFPODA>=0 then begin
346     ECounts[ixHFPODA] := Default(TOCOunt);
347     ACounts[ixHFPODA] := Default(TOCOunt);
348     NDCounts[ixHFPODA] := Default(TOCOunt);
349   end;
350   if ixPFNA>=0 then begin
351     ECounts[ixPFNA] := Default(TOCOunt);
352     ACounts[ixPFNA] := Default(TOCOunt);
353     NDCounts[ixPFNA] := Default(TOCOunt);
354   end;
355
356 end;
357
358 procedure TcmPFASOption2x.SetupModule;
359 begin
360   inherited;
361   MCLSafetyFactor := 0.8;
362
363   //pfos
364   var ix := FData.Occ.ContamList.IndexOf('PFOS');
365   if ix>-1 then MCL[ix] := 4.05;
366   //pfoa
367   ix := FData.Occ.ContamList.IndexOf('PFOA');
368   if ix>-1 then MCL[ix] := 4.05;
369
370   ix := FData.Occ.ContamList.IndexOf('PFHXS');
371   if ix>-1 then DL[ix] := 1.4;
372   ix := FData.Occ.ContamList.IndexOf('PFNA');
373   if ix>-1 then DL[ix] := 0.7;
374   ix := FData.Occ.ContamList.IndexOf('HFPODA');
375   if ix>-1 then DL[ix] := 1.9;
376
377   if CompareText(fOptionName,'PFASOption2x')=0 then begin
378     //all HI chems
379     ix := FData.Occ.ContamList.IndexOf('PFHXS');
380     if ix>-1 then MCL[ix] := 15;
381
382     ix := FData.Occ.ContamList.IndexOf('PFNA');
383     if ix>-1 then MCL[ix] := 15;
384
385     ix := FData.Occ.ContamList.IndexOf('HFPODA');
386     if ix>-1 then MCL[ix] := 15;
387   end else
388   if CompareText(fOptionName,'PFASOption2x-PFNA')=0 then begin
389     ix := FData.Occ.ContamList.IndexOf('PFHXS');
390     if ix>-1 then MCL[ix] := 15;
391
392     ix := FData.Occ.ContamList.IndexOf('HFPODA');
393     if ix>-1 then MCL[ix] := 15;
394   end else
395   if CompareText(fOptionName,'PFASOption2x-PFHXS')=0 then begin
396     ix := FData.Occ.ContamList.IndexOf('PFNA');
397     if ix>-1 then MCL[ix] := 15;
```

```
398
399     ix := FData.Occ.ContamList.IndexOf('HFPODA');
400     if ix>-1 then MCL[ix] := 15;
401   end else
402   if CompareText(fOptionName,'PFASOption2x-HFPODA')=0 then begin
403     ix := FData.Occ.ContamList.IndexOf('PFHXS');
404     if ix>-1 then MCL[ix] := 15;
405
406     ix := FData.Occ.ContamList.IndexOf('PFNA');
407     if ix>-1 then MCL[ix] := 15;
408   end else
409     raise exception.Create('Bad Option name in PFASOption2x');
410
411   //the MCLs are under 70ppt so POU is eliminated
412   NoPOU := true;
413 end;
414
415 procedure TcmPFASOption2x.SetupOutputs(Metrics: TCategoryMetrics);
416 begin
417   inherited;
418
419   Metrics.AddMetric(fOptionName+'.PFASPWS',
420                     'PWSs with PFAS Exceedance (Quarterly Samples)', @ECountT.PWS, mtInfo, mcSumOnly, False);
421   Metrics.AddMetric(fOptionName+'.PFASPWSPOP',
422                     'PWS Population Impacted by PFAS Exceedance (Quarterly Samples)', @ECountT.PWSPop, mtInfo, mcSumOnly, False);
423   Metrics.AddMetric(fOptionName+'.PFASEP',
424                     'EPs with PFAS Exceedance (Quarterly Samples)', @ECountT.EP, mtInfo, mcSumOnly, False);
425   Metrics.AddMetric(fOptionName+'.PFASEPPOP',
426                     'EPs Population Impacted by PFAS Exceedance (Quarterly Samples)', @ECountT.EPPop, mtInfo, mcSumOnly, False);
427
428
429   Metrics.AddMetric(fOptionName+'.PFASPWS_A',
430                     'PWSs with PFAS Annual Sampling', @ACountT.PWS, mtInfo, mcSumOnly, False);
431   Metrics.AddMetric(fOptionName+'.PFASPWSPOP_A',
432                     'PWS Population Impacted by PFAS Annual Sampling', @ACountT.PWSPop, mtInfo, mcSumOnly, False);
433   Metrics.AddMetric(fOptionName+'.PFASEP_A',
434                     'EPs with PFAS Annual Sampling', @ACountT.EP, mtInfo, mcSumOnly, False);
435   Metrics.AddMetric(fOptionName+'.PFASEPPOP_A',
436                     'EPs Population Impacted by PFAS Annualy Sampling', @ACountT.EPPop, mtInfo, mcSumOnly, False);
437
438
439   Metrics.AddMetric(fOptionName+'.PFASPWS_ND',
440                     'PWSs with PFAS Detects', @NDCountT.PWS, mtInfo, mcSumOnly, False);
441   Metrics.AddMetric(fOptionName+'.PFASPWSPOP_ND',
442                     'PWS Population Impacted by PFAS Detects', @NDCountT.PWSPop, mtInfo, mcSumOnly, False);
443   Metrics.AddMetric(fOptionName+'.PFASEP_ND',
444                     'EPs with PFAS Detects', @NDCountT.EP, mtInfo, mcSumOnly, False);
445   Metrics.AddMetric(fOptionName+'.PFASEPPOP_ND',
446                     'EPs Population Impacted by PFAS Detects', @NDCountT.EPPop, mtInfo, mcSumOnly, False);
447
448   if ixPFOS>=0 then begin
449     Metrics.AddMetric(fOptionName+'.PFOSPWS',
450                       'PWSs with PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].PWS, mtInfo, mcSumOnly, False);
451     Metrics.AddMetric(fOptionName+'.PFOSPWSPOP',
452                       'PWS Population Impacted by PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].PWSPop, mtInfo, mcSumOnly, False);
453     Metrics.AddMetric(fOptionName+'.PFOSEP',
454                       'EPs with PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].EP, mtInfo, mcSumOnly, False);
455     Metrics.AddMetric(fOptionName+'.PFOSEPPOP',
456                       'EPs Population Impacted by PFOS Exceedance (Quarterly Samples)', @ECounts[ixPFOS].EPPop, mtInfo, mcSumOnly, False);
457
458     Metrics.AddMetric(fOptionName+'.PFOSPWS_A',
459                       'PWSs with PFOS Annual Sampling', @ACounts[ixPFOS].PWS, mtInfo, mcSumOnly, False);
460     Metrics.AddMetric(fOptionName+'.PFOSPWSPOP_A',
461                       'PWS Population Impacted by PFOS Annual Sampling', @ACounts[ixPFOS].PWSPop, mtInfo, mcSumOnly, False);
462     Metrics.AddMetric(fOptionName+'.PFOSEP_A',
463                       'EPs with PFOS Annual Sampling', @ACounts[ixPFOS].EP, mtInfo, mcSumOnly, False);
464     Metrics.AddMetric(fOptionName+'.PFOSEPPOP_A',
465                       'EPs Population Impacted by PFOS Annual Sampling', @ACounts[ixPFOS].EPPop, mtInfo, mcSumOnly, False);
466
467     Metrics.AddMetric(fOptionName+'.PFOSPWS_ND',
468                       'PWSs with PFOS Detects', @NDCounts[ixPFOS].PWS, mtInfo, mcSumOnly, False);
469     Metrics.AddMetric(fOptionName+'.PFOSPWSPOP_ND',
470                       'PWS Population Impacted by PFOS Detects', @NDCounts[ixPFOS].PWSPop, mtInfo, mcSumOnly, False);
471     Metrics.AddMetric(fOptionName+'.PFOSEP_ND',
```

```
                    'EPs with PFOS Detects', @NDCounts[ixPFOS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOSEPPOP_ND',
                    'EPs Population Impacted by PFOS Detects', @NDCounts[ixPFOS].EPPop, mtInfo, mcSumOnly, False);

  end;
  if ixPFOA>=0 then begin
    Metrics.AddMetric(fOptionName+'.PFOAPWS',
                    'PWSs with PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAPWSPOP',
                    'PWS Population Impacted by PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEP',
                    'EPs with PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEPPOP',
                    'EPs Population Impacted by PFOA Exceedance (Quarterly Samples)', @ECounts[ixPFOA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFOAPWS_A',
                    'PWSs with PFOA Annual Sampling', @ACounts[ixPFOA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAPWSPOP_A',
                    'PWS Population Impacted by PFOA Annual Sampling', @ACounts[ixPFOA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEP_A',
                    'EPs with PFOA Annual Sampling', @ACounts[ixPFOA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEPPOP_A',
                    'EPs Population Impacted by PFOA Annual Sampling', @ACounts[ixPFOA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFOAPWS_ND',
                    'PWSs with PFOA Detects', @NDCounts[ixPFOA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAPWSPOP_ND',
                    'PWS Population Impacted by PFOA Detects', @NDCounts[ixPFOA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEP_ND',
                    'EPs with PFOA Detects', @NDCounts[ixPFOA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFOAEPPOP_ND',
                    'EPs Population Impacted by Detects', @NDCounts[ixPFOA].EPPop, mtInfo, mcSumOnly, False);

  end;
  if ixPFHXS>=0 then begin
    Metrics.AddMetric(fOptionName+'.PFHXSPWS',
                    'PWSs with PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSPWSPOP',
                    'PWS Population Impacted by PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEP',
                    'EPs with PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEPPOP',
                    'EPs Population Impacted by PFHXS Exceedance (Quarterly Samples)', @ECounts[ixPFHXS].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFHXSPWS_A',
                    'PWSs with PFHXS Annual Sampling', @ACounts[ixPFHXS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSPWSPOP_A',
                    'PWS Population Impacted by PFHXS Annual Sampling', @ACounts[ixPFHXS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEP_A',
                    'EPs with PFHXS Exceedance Annual Sampling', @ACounts[ixPFHXS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEPPOP_A',
                    'EPs Population Impacted by PFHXS Annual Sampling', @ACounts[ixPFHXS].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFHXSPWS_ND',
                    'PWSs with PFHXS Detects', @NDCounts[ixPFHXS].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSPWSPOP_ND',
                    'PWS Population Impacted by PFHXS Detects', @NDCounts[ixPFHXS].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEP_ND',
                    'EPs with PFHXS Detects', @NDCounts[ixPFHXS].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFHXSEPPOP_ND',
                    'EPs Population Impacted by PFHXS Detects', @NDCounts[ixPFHXS].EPPop, mtInfo, mcSumOnly, False);
  end;
  if ixPFNA>=0 then begin
    Metrics.AddMetric(fOptionName+'.PFNAPWS',
                    'PWSs with PFNA Exceedance (Quarterly Samples)', @ECounts[ixPFNA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAPWSPOP',
                    'PWS Population Impacted by PFNA Exceedance (Quarterly Samples)', @ECounts[ixPFNA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAEP',
                    'EPs with PFNA Exceedance (Quarterly Samples)', @ECounts[ixPFNA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAEPPOP',
                    'EPs Population Impacted by PFNA Exceedance (Quarterly Samples)', @ECounts[ixPFNA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFNAPWS_A',
                    'PWSs with PFNA Annual Sampling', @ACounts[ixPFNA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAPWSPOP_A',
```

```
                      'PWS Population Impacted by PFNA Annual Sampling', @ACounts[ixPFNA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAEP_A',
                      'EPs with PFNA Exceedance Annual Sampling', @ACounts[ixPFNA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAEPPOP_A',
                      'EPs Population Impacted by PFNA Annual Sampling', @ACounts[ixPFNA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.PFNAPWS_ND',
                      'PWSs with PFNA Detects', @NDCounts[ixPFNA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAPWSPOP_ND',
                      'PWS Population Impacted by PFNA Detects', @NDCounts[ixPFNA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAEP_ND',
                      'EPs with PFNA Detects', @NDCounts[ixPFNA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.PFNAEPPOP_ND',
                      'EPs Population Impacted by PFNA Detects', @NDCounts[ixPFNA].EPPop, mtInfo, mcSumOnly, False);
  end;
  if ixHFPODA>=0 then begin
    Metrics.AddMetric(fOptionName+'.HFPODAPWS',
                      'PWSs with HFPODA Exceedance (Quarterly Samples)', @ECounts[ixHFPODA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAPWSPOP',
                      'PWS Population Impacted by HFPODA Exceedance (Quarterly Samples)', @ECounts[ixHFPODA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAEP',
                      'EPs with HFPODA Exceedance (Quarterly Samples)', @ECounts[ixHFPODA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAEPPOP',
                      'EPs Population Impacted by HFPODA Exceedance (Quarterly Samples)', @ECounts[ixHFPODA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.HFPODAPWS_A',
                      'PWSs with HFPODA Annual Sampling', @ACounts[ixHFPODA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAPWSPOP_A',
                      'PWS Population Impacted by HFPODA Annual Sampling', @ACounts[ixHFPODA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAEP_A',
                      'EPs with HFPODA Exceedance Annual Sampling', @ACounts[ixHFPODA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAEPPOP_A',
                      'EPs Population Impacted by HFPODA Annual Sampling', @ACounts[ixHFPODA].EPPop, mtInfo, mcSumOnly, False);

    Metrics.AddMetric(fOptionName+'.HFPODAPWS_ND',
                      'PWSs with HFPODA Detects', @NDCounts[ixHFPODA].PWS, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAPWSPOP_ND',
                      'PWS Population Impacted by HFPODA Detects', @NDCounts[ixHFPODA].PWSPop, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAEP_ND',
                      'EPs with HFPODA Detects', @NDCounts[ixHFPODA].EP, mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.HFPODAEPPOP_ND',
                      'EPs Population Impacted by HFPODA Detects', @NDCounts[ixHFPODA].EPPop, mtInfo, mcSumOnly, False);
  end;



  Metrics.AddMetric(fOptionName+'.EP_quarterly_yr1',
                      'All EPs in PWS pop < 3300 and with STATEPFAS = 0', @EPY1_PFE0, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EP_exceed_yr1',
                      'Number of EP in PWS pop < 3300 and with STATEPFAS = 0 above MCL for any contaminant in year 1', @EPY1_E, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EP_ nondetect_yr1',
                      'Number of EP in PWS pop < 3300 and with STATEPFAS = 0 above any contaminant detection level in year 1', @EPY1_ND, mtInfo, mcSumOnly, False);

(*
  var s: string;
  for var i := 0 to high(DidSmpT) do begin
    if i<9 then s := '0'+inttostr(i+1) else s:= inttostr(i+1);
    Metrics.AddMetric(fOptionName+'.NumEPWithSamplingCostY'+s+'_Tri',
                        'EPs Tri Sampling in year '+inttostr(i+1), @DidSmpT[i], mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.NumEPWithSamplingCostY'+s+'_Oth',
                        'EPs Other Sampling in year '+inttostr(i+1), @DidSmpO[i], mtInfo, mcSumOnly, False);

  end;
*)
end;

end.

```

code\CostModules\cmPFASOption3.pas

```
unit cmPFASOption3;

interface
uses Sysutils, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterCostBase,
     SafeWaterUtility, SafewaterGlobals, SafeWaterTFObject, cmPFASBase, Math,
      SafewaterMetrics;

type
  TcmPFASOption3 = class(TcmPFASBase)
    ixPFOS, ixPFOA, ixPFHXS, ixPFHPA: integer;
    MCLA, MCLS: double;  //group mcls
    cGrpA, cGrpS : double;  //sum of conc by group;
    GrpAT, GrpST: double;

    procedure PerformanceGACIX; override;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
    procedure SetPWSWideEPValues(const P: TPWSRecord); override;

    procedure SetupModule; override;
    procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
  end;

implementation


{ TcmPFASOption3 }

procedure TcmPFASOption3.AddEntryPointCosts(const P: TPWSRecord;
  const Conc: TContamArray; var FinalConc: TContamArray);
begin
  inherited;
  var TotYears := FData.Cfg.EndYear - FData.Cfg.PFASChangeYear;
  var OnYrChange := FData.Cfg.PFASChangeYear - FData.Cfg.StartYear;
  cGrpA := Conc[ixPFOA] + Conc[ixPFHPA];
  cGrpS := Conc[ixPFOS] + Conc[ixPFHXS];

  if (cGrpA>MCLA) and (Conc[ixPFOA]>MCLA) and (Conc[ixPFHPA]>MCLA) then
    GrpAT :=GrpAT + 1;
  if (cGrpS>MCLS) and (Conc[ixPFOS]>MCLS) and (Conc[ixPFHXS]>MCLS) then
    GrpST :=GrpST + 1;

  b_source := 0;
  b_treat := 0;
  b_exceed := 0;
  b_nondetect := 1;
  for var c := 0 to CMax do begin
    if CurConc[c]>DL[c] then b_nondetect:=0;
  end;
  if (cGrpA>MCLA) or (cGrpS>MCLS) then
   b_exceed := 1;

  {$ifdef DEBUG}
  if _d then begin
    FData.DebugLog.LD('costs','b_exceed', b_exceed);
    FData.DebugLog.LD('costs','b_nondetect', b_nondetect);
  end;
  {$endif}

  if (b_exceed <> 0) then begin
    TookAction := True;
    PerformanceGACIX;
    var Tech := SelectTech(P);
    if Tech < 0 then begin
      FData.ErrLog.L(Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
      {$ifdef DEBUG}
      if _d then
        FData.DebugLog.L('costs',Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
      {$endif}
      exit;
      //TODO what to do?
    end;
    GetWBSCosts(P, OnYrChange, TotYears, Tech);
    //setting final concentrations based on Tech for this option
    var tmp: double;
    var ConcSet := false;
```

```
76      var c : integer;
77      case Tech of
78        ttGAC  : begin
79                   b_treat := 1;
80                   for c := 0 to CMax do
81                     PR[c] := min((BV_GAC - A_TOC * cTOC - B_GAC[c]) / A_R, 0.99);
82                 end;
83        ttIX   : begin
84                   b_treat := 1;
85                   for c := 0 to CMax do
86                     PR[c] := min((BV_IX - A_PFAS_IX * c0Total - B_IX[c]) / A_R_IX, 0.99);
87                 end;
88        ttCRO  : begin
89                   b_treat := 1;
90                   tmp := min(PRM / 0.95,1); //B
91                   for c := 0 to CMax do begin
92                     if tmp = 1 then begin
93                       FinalConc[c] := (CurConc[c] * (1-PRM));
94                     end else begin
95                       FinalConc[c] := (1-tmp) * CurConc[c] + tmp * (CurConc[c] * (1-0.95));
96                     end;
97                     PR[c] := 1 - ((CurConc[c] - FinalConc[c]) / CurConc[c]);
98                   end;
99                   ConcSet := true;
100                end;
101       ttPOU  : begin
102                  b_treat := 1;
103                  tmp := (c0Total - 70) / c0Total;
104                  if tmp<0 then tmp:=1;
105                  for c := 0 to CMax do
106                    PR[c] := tmp;
107                end;
108       else     begin
109                  b_source := 1;
110                  //remove everything for other technologies
111                  for c := 0 to CMax do
112                    PR[c] := 1;
113                end;
114     end;
115     if not ConcSet then begin
116       for c := 0 to CMax do
117         FinalConc[c] := CurConc[c] * (1 - PR[c]);
118     end;
119
120     {$ifdef DEBUG}
121     if _d then begin
122       FData.DebugLog.L('costs','TECH');
123       FData.DebugLog.LD('costs','tech',FData.TechForecast.TechToText(Tech));
124       FData.DebugLog.LD('costs','df', Inf.DesignFlow);
125       FData.DebugLog.LD('costs','af', Inf.AverageFlow);
126       FData.DebugLog.LD('costs','toc', cTOC);
127       FData.DebugLog.LD('costs','bv_gac', BV_GAC);
128       FData.DebugLog.LD('costs','bv_ix', BV_IX);
129       for c := 0 to CMax do begin
130          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
131          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
132          FData.DebugLog.LD('costs','MCL',MCL[c]);
133          FData.DebugLog.LD('costs','PR',PR[c]);
134       end;
135     end;
136     {$endif}
137
138
139   end;  //b_exceed<>0
140
141   var OnYr: integer;
142   for var Yr := FData.Cfg.StartYear to FData.Cfg.EndYear do begin
143     OnYr := Yr  - FData.Cfg.StartYear;
144     if FreshPWS then
145       csOption._EvaluateSystem(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
146
147     csOption._Evaluate(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
148     {$ifdef DEBUG}
149     if _d then begin
```

```
150        (*
151       var TT := csOption._DumpCalcs;
152       FData.DebugLog.L('***WORKBOOK BY YEAR: '+OnYr.ToString);
153       for var tti := 0 to TT.Count-1 do
154         FData.DebugLog.L(TT[tti]);
155       TT.Free;
156       *)
157     end;
158     {$endif}
159     inc(OnYr);
160   end;
161   FreshPWS := False;
162   //accumulate to the system for this entry point...
163   for var i := 0 to fData.Inp.RateCount-1 do begin
164     _i(O[i].AgencyImpAdmin,_T[i].AgencyImpAdmin);
165     _i(O[i].AgencyTreatmentAdmin, _T[i].AgencyTreatmentAdmin);
166     _i(O[i].PWSImpAdmin,_T[i].PWSImpAdmin);
167     _i(O[i].AgencyImpSampling,_T[i].AgencyImpSampling);
168     _i(O[i].PWSSampling,_T[i].PWSSampling);
169     _i(O[i].PWSImpAdmin_pws,_T[i].PWSImpAdmin_pws);
170     _i(O[i].PWSSampling_pws,_T[i].PWSSampling_pws);
171     _i(O[i].PWSTreatmentAdmin, _T[i].PWSTreatmentAdmin);
172     _i(O[i].PWSTreatmentAdmin_pws, _T[i].PWSTreatmentAdmin_pws);
173
174     _i(O[i].PWSTreatment,_T[i].PWSTreatment);
175     _i(O[i].PWSTreatment_pws,_T[i].PWSTreatment_pws);
176     _i(O[i].CapitalCost,_T[i].CapitalCost);
177     _i(O[i].CapitalCost_pws,_T[i].CapitalCost_pws);
178     _i(O[i].OMCost,_T[i].OMCost);
179     _i(O[i].OMCost_pws,_T[i].OMCost_pws);
180   end;
181 end;
182
183 procedure TcmPFASOption3.PerformanceGACIX;
184 begin
185   var
186   PRS, PRA,
187   PRAg_PFOA, PRAg_PFHPA,
188   PRSg_PFOS, PRSg_PFHXS,
189   PRAi_PFOA, PRAi_PFHPA,
190   PRSi_PFOS, PRSi_PFHXS,
191   bvgPFOA, bvgPFOS, bvgPFHPA, bvgPFHXS,
192   bviPFOA, bviPFOS, bviPFHPA, bviPFHXS
193   : double;
194
195   PRA := (cGRPA - MCLA * MCLSafetyFactor) / cGRPA;
196   PRS := (cGRPS - MCLA * MCLSafetyFactor) / cGRPS;
197   PRM := max(PRA,PRS);
198   if PRM>0.99 then begin
199     tFV[ttGAC] := false;
200     tFV[ttIX] := false;
201     RedistributeTechForecast;
202     exit;
203   end;
204
205   //TODO laying this out the long way before we see if we can refactor
206
207   //A Group;
208   PRAg_PFOA := (CurConc[ixPFOA] + CurConc[ixPFHPA] *
209           (1 + (B_GAC[ixPFHPA] - B_GAC[ixPFOA]) / A_R) -
210           MCLA * MCLSafetyFactor) / cGRPA;
211   PRAg_PFHPA := (CurConc[ixPFHPA] + CurConc[ixPFOA] *
212           (1 + (B_GAC[ixPFOA] - B_GAC[ixPFHPA]) / A_R) -
213           MCLA * MCLSafetyFactor) / cGRPA;
214
215   if (PRAg_PFHPA>0.99) and (PRAg_PFOA>0.99) then begin
216     PRAg_PFHPA := 0.99;
217     PRAg_PFOA := 0.99;
218   end else begin
219      if PRAg_PFHPA>0.99 then begin
220        PRAg_PFHPA:=0.99;
221        PRAg_PFOA := (CurConc[ixPFOA] + CurConc[ixPFHPA] *
222             (1 - PRAg_PFHPA) -
223             MCLA * MCLSafetyFactor) / (cGRPA -CurConc[ixPFHPA]);
224      end else
```

```
225      if PRAg_PFOA>0.99 then begin
226        PRAg_PFOA:=0.99;
227        PRAg_PFHPA := (CurConc[ixPFHPA] + CurConc[ixPFOA] *
228             (1  - PRAg_PFOA) -
229             MCLA * MCLSafetyFactor) / (cGRPA - CurConc[ixPFOA]);
230      end;
231    end;
232
233    //S Group
234    PRSg_PFOS := (CurConc[ixPFOS] + CurConc[ixPFHXS] *
235            (1 + (B_GAC[ixPFHXS] - B_GAC[ixPFOS]) / A_R) -
236            MCLS * MCLSafetyFactor) / cGRPS;
237    PRSg_PFHXS := (CurConc[ixPFHXS] + CurConc[ixPFOS] *
238            (1 + (B_GAC[ixPFOS] - B_GAC[ixPFHXS]) / A_R) -
239            MCLS * MCLSafetyFactor) / cGRPS;
240
241    if (PRSg_PFHXS>0.99) and (PRSg_PFOS>0.99) then begin
242      PRSg_PFHXS := 0.99;
243      PRSg_PFOS := 0.99;
244    end else begin
245      if PRSg_PFHXS>0.99 then begin
246        PRSg_PFHXS:=0.99;
247        PRSg_PFOS := (CurConc[ixPFOS] + CurConc[ixPFHXS] *
248             (1 - PRSg_PFHXS) -
249             MCLA * MCLSafetyFactor) / (cGRPS -CurConc[ixPFHXS]);
250      end else
251      if PRSg_PFOS>0.99 then begin
252        PRSg_PFOS:=0.99;
253        PRSg_PFHXS := (CurConc[ixPFHXS] + CurConc[ixPFOS] *
254             (1  - PRSg_PFOS) -
255             MCLA * MCLSafetyFactor) / (cGRPS - CurConc[ixPFOS]);
256      end;
257    end;
258
259    //ix
260    //A Group;
261    PRAi_PFOA := (CurConc[ixPFOA] + CurConc[ixPFHPA] *
262            (1 + (B_IX[ixPFHPA] - B_IX[ixPFOA]) / A_R_IX) -
263            MCLA * MCLSafetyFactor) / cGRPA;
264    PRAi_PFHPA := (CurConc[ixPFHPA] + CurConc[ixPFOA] *
265            (1 + (B_IX[ixPFOA] - B_IX[ixPFHPA]) / A_R_IX) -
266            MCLA * MCLSafetyFactor) / cGRPA;
267
268    if (PRAi_PFHPA>0.99) and (PRAi_PFOA>0.99) then begin
269      PRAi_PFHPA := 0.99;
270      PRAi_PFOA := 0.99;
271    end else begin
272      if PRAi_PFHPA>0.99 then begin
273        PRAi_PFHPA:=0.99;
274        PRAi_PFOA := (CurConc[ixPFOA] + CurConc[ixPFHPA] *
275             (1 - PRAi_PFHPA) -
276             MCLA * MCLSafetyFactor) / (cGRPA -CurConc[ixPFHPA]);
277      end else
278      if PRAi_PFOA>0.99 then begin
279        PRAi_PFOA:=0.99;
280        PRAi_PFHPA := (CurConc[ixPFHPA] + CurConc[ixPFOA] *
281             (1  - PRAi_PFOA) -
282             MCLA * MCLSafetyFactor) / (cGRPA - CurConc[ixPFOA]);
283      end;
284    end;
285
286    //S Group
287    PRSi_PFOS := (CurConc[ixPFOS] + CurConc[ixPFHXS] *
288            (1 + (B_IX[ixPFHXS] - B_IX[ixPFOS]) / A_R_IX) -
289            MCLS * MCLSafetyFactor) / cGRPS;
290    PRSi_PFHXS := (CurConc[ixPFHXS] + CurConc[ixPFOS] *
291            (1 + (B_IX[ixPFOS] - B_IX[ixPFHXS]) / A_R_IX) -
292            MCLS * MCLSafetyFactor) / cGRPS;
293
294    if (PRSi_PFHXS>0.99) and (PRSi_PFOS>0.99) then begin
295      PRSi_PFHXS:=0.99;
296      PRSi_PFOS:=0.99;
297    end else begin
298      if PRSi_PFHXS>0.99 then begin
```

```
      PRSi_PFHXS:=0.99;
      PRSi_PFOS := (CurConc[ixPFOS] + CurConc[ixPFHXS] *
           (1 - PRSi_PFHXS) -
           MCLA * MCLSafetyFactor) / (cGRPS -CurConc[ixPFHXS]);
    end else
    if PRSi_PFOS>0.99 then begin
      PRSg_PFOS := 0.99;
      PRSi_PFHXS := (CurConc[ixPFHXS] + CurConc[ixPFOS] *
           (1  - PRSg_PFOS) -
           MCLA * MCLSafetyFactor) / (cGRPS - CurConc[ixPFOS]);
    end;
  end;

  //Bed volumes
  bvgPFOA := A_TOC * cTOC + A_R * PRAg_PFOA + B_GAC[ixPFOA];
  bvgPFHPA := A_TOC * cTOC + A_R * PRAg_PFHPA + B_GAC[ixPFHPA];
  bvgPFOS := A_TOC * cTOC + A_R * PRSg_PFOS + B_GAC[ixPFOS];
  bvgPFHXS := A_TOC * cTOC + A_R * PRSg_PFHXS + B_GAC[ixPFHXS];

  BV_GAC := min(min(bvgPFOA, bvgPFOS), min(bvgPFHPA, bvgPFHXS));
  if BV_GAC < 5000 then tFV[ttGAC] := false;

  bviPFOA := A_PFAS_IX + A_R_IX * PRAi_PFOA + B_IX[ixPFOA];
  bviPFHPA := A_PFAS_IX + A_R_IX * PRAi_PFHPA + B_IX[ixPFHPA];
  bviPFOS := A_PFAS_IX + A_R_IX * PRSi_PFOS + B_IX[ixPFOS];
  bviPFHXS := A_PFAS_IX + A_R_IX * PRSi_PFHXS + B_IX[ixPFHXS];
  BV_IX := min(min(bviPFOA, bviPFOS), min(bviPFHPA, bviPFHXS));
  if BV_IX < 20000 then tFV[ttIX] := false;

  RedistributeTechForecast;
end;

procedure TcmPFASOption3.SetPWSWideEPValues(const P: TPWSRecord);
begin
  inherited;
  GrpAT:=0;
  GrpST:=0;
end;

procedure TcmPFASOption3.SetupModule;
begin
  inherited;
  ixPFOS := FData.Occ.ContamList.IndexOf('PFOS');
  ixPFOA := FData.Occ.ContamList.IndexOf('PFOA');
  ixPFHXS := FData.Occ.ContamList.IndexOf('PFHXS');
  ixPFHPA := FData.Occ.ContamList.IndexOf('PFHPA');

  //TODO config these...
  MCLA := 4.05;
  MCLS := 4.05;
  DL[ixPFHXS] := 1.4;
  DL[ixPFHPA] := 1;

  //for option 3, the MCLs are under 70ppt so POU is eliminated
  NoPOU := true;
end;

procedure TcmPFASOption3.SetupOutputs(Metrics: TCategoryMetrics);
begin
  inherited;
  Metrics.AddMetric(fOptionName+'.GroupASumTrigger',
                    'Contams below 4, Sum > 4', @GrpAT, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.GroupSSumTrigger',
                    'Contams below 4, Sum > 4', @GrpST, mtInfo, mcSumOnly, False);
end;

end.

```

code\CostModules\cmPFASOption4.pas

```
unit cmPFASOption4;

```

```
interface
uses Sysutils, SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterCostBase,
     SafeWaterUtility, SafewaterGlobals, SafeWaterTFObject, cmPFASBase, Math,
      SafewaterMetrics;

type
  TcmPFASOption4 = class(TcmPFASBase)
    ixPFOS, ixPFOA, ixPFHXS, ixPFHPA,
    ixPFBS, ixPFNA, ixHFPODA : integer;

    TT,RTT: double;
    PRg, PRi: array[0..20] of double;

    procedure PerformanceGACIX4;
    procedure PerformanceGACIX4_1;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
    procedure SetPWSWideEPValues(const P: TPWSRecord); override;

    procedure SetupModule; override;
    procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
  end;

implementation


{ TcmPFASOption4 }

procedure TcmPFASOption4.AddEntryPointCosts(const P: TPWSRecord;
  const Conc: TContamArray; var FinalConc: TContamArray);
begin
  inherited;
  var TotYears := FData.Cfg.EndYear - FData.Cfg.PFASChangeYear;
  var OnYrChange := FData.Cfg.PFASChangeYear - FData.Cfg.StartYear;
  var c : integer;

  if c0Total<14 then TT:=0 else
  if c0Total<22 then TT:=5 else
  if c0Total<40 then TT:=7 else
  if c0Total<60 then TT:=10 else
    TT:=14;

  b_source := 0;
  b_treat := 0;
  b_exceed := 0;
  b_nondetect := 1;
  for c := 0 to CMax do begin
    if CurConc[c]>DL[c] then b_nondetect:=0;
  end;

  RTT := (c0Total - TT) / c0Total;

  var Tech: integer;
  if TT=0 then begin
    if (CurConc[ixPFOS] > MCL[ixPFOS]) or
       (CurConc[ixPFOA] > MCL[ixPFOA]) then begin
         b_exceed := 1;
         PerformanceGACIX;
         Tech := SelectTech(P);
         if Tech = ttGAC  then begin
           for c := 0 to CMax do
             PRg[c] := min((BV_GAC - A_TOC * cTOC - B_GAC[c]) / A_R, 0.99);
         end else
         if Tech = ttIX then begin
           for c := 0 to CMax do
             PRi[c] := min((BV_IX - A_PFAS_IX * c0Total - B_IX[c]) / A_R_IX, 0.99);
         end;
       end;
  end else
  if RTT>0.99 then begin
    b_exceed := 1;
    tFV[ttGAC] := false;
    tFV[ttIX] := false;
    RedistributeTechForecast;
    Tech := SelectTech(P);
  end else begin
```

```
78      b_exceed := 1;
79      PerformanceGACIX4;
80      Tech := SelectTech(P);
81    end;
82
83    {$ifdef DEBUG}
84    if _d then begin
85      FData.DebugLog.LD('costs','b_exceed', b_exceed);
86      FData.DebugLog.LD('costs','b_nondetect', b_nondetect);
87    end;
88    {$endif}
89
90    if (b_exceed <> 0) then begin
91      TookAction := True;
92      //set PRM for CRO cost...
93      PRM := max(RTT,max(
94                 (CurConc[ixPFOA] - MCL[ixPFOA] * MCLSafetyFactor) / CurConc[ixPFOA],
95                 (CurConc[ixPFOS] - MCL[ixPFOS] * MCLSafetyFactor) / CurConc[ixPFOS]
96                ));
97
98      GetWBSCosts(P, OnYrChange, TotYears, Tech);
99      if Tech < 0 then begin
100       FData.ErrLog.L(Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
101       {$ifdef DEBUG}
102       if _d then
103         FData.DebugLog.L('costs',Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
104       {$endif}
105       exit;
106       //TODO what to do?
107     end;
108
109     //setting final concentrations based on Tech for this option
110     var tmp: double;
111     var ConcSet := false;
112     case Tech of
113       ttGAC  : begin
114                  b_treat := 1;
115                  for c := 0 to CMax do
116                    PR[c] := min(PRg[c], 0.99);
117                end;
118       ttIX   : begin
119                  b_treat := 1;
120                  for c := 0 to CMax do
121                    PR[c] := min(PRi[c], 0.99);
122                end;
123       ttCRO  : begin
124                  b_treat := 1;
125                  for c := 0 to CMax do begin
126                    PR[c] := PRM;
127                  end;
128                end;
129       ttPOU  : begin
130                  b_treat := 1;
131                  tmp := (c0Total - 70) / c0Total;
132                  if tmp<0 then tmp:=1;
133                  for c := 0 to CMax do
134                    PR[c] := tmp;
135                end;
136       else     begin
137                  b_source := 1;
138                  //remove everything for other technologies
139                  for c := 0 to CMax do
140                    PR[c] := 1;
141                end;
142     end;
143     if not ConcSet then begin
144       for c := 0 to CMax do
145         FinalConc[c] := CurConc[c] * (1 - PR[c]);
146     end;
147
148     {$ifdef DEBUG}
149     if _d then begin
150       FData.DebugLog.L('costs','TECH');
151       FData.DebugLog.LD('costs','tech',FData.TechForecast.TechToText(Tech));
```

```
152       FData.DebugLog.LD('costs','df', Inf.DesignFlow);
153       FData.DebugLog.LD('costs','af', Inf.AverageFlow);
154       FData.DebugLog.LD('costs','toc', cTOC);
155       FData.DebugLog.LD('costs','bv_gac', BV_GAC);
156       FData.DebugLog.LD('costs','bv_ix', BV_IX);
157       FData.DebugLog.LD('costs','PRM', PRM);
158       FData.DebugLog.LD('costs','RTT', RTT);
159       for c := 0 to CMax do begin
160          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
161          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
162          FData.DebugLog.LD('costs','MCL',MCL[c]);
163          FData.DebugLog.LD('costs','PR',PR[c]);
164       end;
165     end;
166     {$endif}
167
168
169   end;  //b_exceed<>0
170
171   var OnYr: integer;
172   for var Yr := FData.Cfg.StartYear to FData.Cfg.EndYear do begin
173     OnYr := Yr  - FData.Cfg.StartYear;
174     if FreshPWS then
175       csOption._EvaluateSystem(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
176
177     csOption._Evaluate(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
178     {$ifdef DEBUG}
179     if _d then begin
180        (*
181       var TT := csOption._DumpCalcs;
182       FData.DebugLog.L('***WORKBOOK BY YEAR: '+OnYr.ToString);
183       for var tti := 0 to TT.Count-1 do
184         FData.DebugLog.L(TT[tti]);
185       TT.Free;
186       *)
187     end;
188     {$endif}
189     inc(OnYr);
190   end;
191   FreshPWS := False;
192   //accumulate to the system for this entry point...
193   for var i := 0 to fData.Inp.RateCount-1 do begin
194     _i(O[i].AgencyImpAdmin,_T[i].AgencyImpAdmin);
195     _i(O[i].AgencyTreatmentAdmin, _T[i].AgencyTreatmentAdmin);
196     _i(O[i].PWSImpAdmin,_T[i].PWSImpAdmin);
197     _i(O[i].AgencyImpSampling,_T[i].AgencyImpSampling);
198     _i(O[i].PWSSampling,_T[i].PWSSampling);
199     _i(O[i].PWSImpAdmin_pws,_T[i].PWSImpAdmin_pws);
200     _i(O[i].PWSSampling_pws,_T[i].PWSSampling_pws);
201     _i(O[i].PWSTreatmentAdmin, _T[i].PWSTreatmentAdmin);
202     _i(O[i].PWSTreatmentAdmin_pws, _T[i].PWSTreatmentAdmin_pws);
203
204     _i(O[i].PWSTreatment,_T[i].PWSTreatment);
205     _i(O[i].PWSTreatment_pws,_T[i].PWSTreatment_pws);
206     _i(O[i].CapitalCost,_T[i].CapitalCost);
207     _i(O[i].CapitalCost_pws,_T[i].CapitalCost_pws);
208     _i(O[i].OMCost,_T[i].OMCost);
209     _i(O[i].OMCost_pws,_T[i].OMCost_pws);
210   end;
211 end;
212
213 procedure TcmPFASOption4.PerformanceGACIX4;
214 var MRT : array[0..20,0..20] of double;
215 begin
216   var
217   bvgPFOA, bvgPFOS, bvgPFHPA, bvgPFHXS,
218   bviPFOA, bviPFOS, bviPFHPA, bviPFHXS,
219   C0T
220   : double;
221 //csl('IN');
222   var c,t: integer;
223   //set terms for GAC
224   for c := 0  to CMax do begin
225     PRg[c] := 0;
226     for t := 0  to CMax do begin
```

```
227       if c=t then continue;
228       MRT[c,t] := (B_GAC[t] - B_GAC[c]) / A_R;
229     end;
230   end;
231
232   var subcnt:integer;
233   var sub := false;
234   C0t := C0Total;
235   subcnt:=0;
236
237   for subcnt:=0 to CMax do begin
238     for c := 0  to CMax do begin
239       if PRg[c] = 0.99 then continue;
240
241       PRg[c] := CurConc[c];
242       for t := 0  to CMax do begin
243         if c=t then continue;
244 //csl('c:'+c.ToString+'  t:'+t.ToString+ ' pr1:'+PRg[c].ToString+'  mrt:'+MRT[c,t].ToString);
245         PRg[c] := PRg[c] + CurConc[t] * (1 + MRT[c,t]);
246 //csl('c:'+c.ToString+'  t:'+t.ToString+' pr2:'+PRg[c].ToString);
247       end;
248       PRg[c] := (PRg[c] - TT) / C0T;
249     end;
250
251 (*
252 csl(subcnt.ToString);
253 csl('c0t:'+c0t.ToString);
254 for c := 0  to CMax do begin
255   for t := 0  to CMax do begin
256     csl('c:'+c.ToString+'   t:'+t.ToString+'   m:'+MRT[c,t].ToString);
257   end;
258 end;
259 for c := 0  to CMax do begin
260 csl('c:'+c.ToString+'  p:'+PRg[c].ToString);
261 end;
262 *)
263     sub := false;
264     for c := 0  to CMax do begin
265       if PRg[c] > 0.99 then begin
266         C0T := C0T - CurConc[c];
267         PRg[c] := 0.99;
268         sub:=true;
269         for t := 0  to CMax do begin
270           MRT[t,c] := -0.99;
271         end;
272       end;
273     end;
274     if not sub then break;
275   end;
276
277   //set terms for IX
278   for c := 0  to CMax do begin
279     PRi[c] := 0;
280     for t := 0  to CMax do begin
281       if c=t then continue;
282       MRT[c,t] := (B_IX[t] - B_IX[c]) / A_R_IX;
283     end;
284   end;
285   C0t := C0Total;
286   subcnt:=0;
287   for subcnt:=0 to CMax do begin
288     for c := 0  to CMax do begin
289       if PRi[c] = 0.99 then continue;
290       PRi[c] := CurConc[c];
291       for t := 0  to CMax do begin
292         if c=t then continue;
293         PRi[c] := PRi[c] + CurConc[t] * (1 + MRT[c,t]);
294       end;
295       PRi[c] := (PRi[c] - TT) / C0T;
296     end;
297
298     sub := false;
299     for c := 0  to CMax do begin
300       if PRi[c] > 0.99 then begin
```

```
301         C0T := C0T - CurConc[c];
302         PRi[c] := 0.99;
303         sub:=true;
304         for t := 0  to CMax do begin
305           MRT[t,c] := -0.99;
306         end;
307       end;
308     end;
309     if not sub then break;
310   end;
311
312   //Check Option 1...
313   PerformanceGACIX4_1;
314
315   //Bed volumes
316   bvgPFOA := A_TOC * cTOC + A_R * PRg[ixPFOA] + B_GAC[ixPFOA];
317   bvgPFHPA := A_TOC * cTOC + A_R * PRg[ixPFHPA] + B_GAC[ixPFHPA];
318   bvgPFOS := A_TOC * cTOC + A_R * PRg[ixPFOS] + B_GAC[ixPFOS];
319   bvgPFHXS := A_TOC * cTOC + A_R * PRg[ixPFHXS] + B_GAC[ixPFHXS];
320
321   BV_GAC := min(BV_GAC, min(min(bvgPFOA, bvgPFOS), min(bvgPFHPA, bvgPFHXS)));
322   if BV_GAC < 5000 then tFV[ttGAC] := false;
323
324   bviPFOA := A_PFAS_IX + A_R_IX * PRi[ixPFOA] + B_IX[ixPFOA];
325   bviPFHPA := A_PFAS_IX + A_R_IX * PRi[ixPFHPA] + B_IX[ixPFHPA];
326   bviPFOS := A_PFAS_IX + A_R_IX * PRi[ixPFOS] + B_IX[ixPFOS];
327   bviPFHXS := A_PFAS_IX + A_R_IX * PRi[ixPFHXS] + B_IX[ixPFHXS];
328   BV_IX := min(BV_IX, min(min(bviPFOA, bviPFOS), min(bviPFHPA, bviPFHXS)));
329   if BV_IX < 20000 then tFV[ttIX] := false;
330
331   RedistributeTechForecast;
332 end;
333
334 function _gpr(const c0,mcl,sf: double; var PRM: double) : double;
335 begin
336   if (c0>0) and (mcl<1e30) and (c0>mcl) then
337     Result := (c0 - mcl * sf) / c0
338   else
339     Result := 0;
340   if Result > PRM then PRM := Result;
341 end;
342
343 procedure TcmPFASOption4.PerformanceGACIX4_1;
344 begin
345   var i: integer;
346   PRM := 0;
347   //TODO should these be set to large numbers in the absence of MCLs
348   for i := 0 to CMax do
349      PR[i] := _gpr(CurConc[i], MCL[i], MCLSafetyFactor,PRM);
350
351   BV_GAC := 1e30;
352   BV_IX := 1e30;
353   if PRM > 0.99 then begin
354
355     exit;
356   end;
357
358   for i := 0 to CMax do
359     BV_GAC := min(BV_GAC,A_TOC * cTOC + A_R * PR[i] + B_GAC[i]);
360
361   for i := 0 to CMax do
362     BV_IX := min(BV_IX, A_PFAS_IX * C0Total + A_R_IX * PR[i] + B_IX[i]);
363
364
365 end;
366
367 procedure TcmPFASOption4.SetPWSWideEPValues(const P: TPWSRecord);
368 begin
369   inherited;
370 end;
371
372 procedure TcmPFASOption4.SetupModule;
373 begin
374   inherited;
375   ixPFOS := FData.Occ.ContamList.IndexOf('PFOS');
```

```
  ixPFOA := FData.Occ.ContamList.IndexOf('PFOA');
  ixPFHXS := FData.Occ.ContamList.IndexOf('PFHXS');
  ixPFHPA := FData.Occ.ContamList.IndexOf('PFHPA');
  MCLSafetyFactor := 0.8;

  MCL[ixPFOS] := 4.05;
  MCL[ixPFOA] := 4.05;

  //TODO config these...
  DL[ixPFHXS] := 1.4;
  DL[ixPFHPA] := 1;

  //for option 4, the MCLs are under 70ppt so POU is eliminated
  NoPOU := true;
end;

procedure TcmPFASOption4.SetupOutputs(Metrics: TCategoryMetrics);
begin
  inherited;
end;

end.
```

code\CostModules\cmPFASOptionMCL1.pas

```
unit cmPFASOptionMCL1;

interface

uses Sysutils, Classes,SafeWaterOccurrence, SafeWaterPWSRecords, SafeWaterModelData, SafeWaterCostBase,
     SafeWaterUtility, SafewaterGlobals, SafeWaterTFObject, cmPFASBase, Math,
     SafewaterMetrics, Generics.Collections;

type
  TOCount=record
    PWS,EP, PWSPop,EPPop : double;
  end;

  TcmPFASOptionMCL1 = class(TcmPFASBase)
    aFlag: boolean;
    b_CntAnnual: integer;
    ECountT, ACountT,NDCountT: TOCount;
    //late addition to metrics - just shimming in here....
    EPY1_PFE0, EPY1_ND, EPY1_E: double;

//TODO temp debug outputs remove
DidSmpT, DidSmpO : array of double;

    constructor Create(aData : PSWModelData; GlbUnc: TGlobalUncertainty;  aNeededContams: string; aOptionName: string; aIteration: integer); override;
    procedure SetPWSWideEPValues(const P: TPWSRecord); override;
    procedure SetupModule; override;
    procedure AddEntryPointCosts(const P: TPWSRecord; const Conc : TContamArray; var FinalConc: TContamArray); override;
    procedure SetupOutputs(Metrics: TCategoryMetrics); override;
  end;

implementation


{ TcmPFASOptionMCL1 }

procedure TcmPFASOptionMCL1.AddEntryPointCosts(const P: TPWSRecord;
  const Conc: TContamArray; var FinalConc: TContamArray);
begin
  inherited;
  var TotYears := FData.Cfg.EndYear - FData.Cfg.PFASChangeYear;
  var OnYrChange := FData.Cfg.PFASChangeYear - FData.Cfg.StartYear;

  b_source := 0;
  b_treat := 0;
  b_exceed := 0;
```

```
46    b_nondetect := 1;
47    b_CntAnnual := 0;  //just  temp variable to handle annual monitoring counts - not otherwise used
48    var c : integer;
49    for c := 0 to CMax do begin
50      if (CurConc[c]>DL[c]) and (MCL[c]<1e30) then begin
51        if b_nondetect=1 then begin
52          NDCountT.PWS := 1;
53          NDCountT.PWSPop := P.Population;
54          NDCountT.EP := NDCountT.EP + 1;
55          NDCountT.EPPop := NDCountT.EPPop + CurPop;
56        end;
57        b_nondetect:=0;
58      end;
59      if CurConc[c] >= (MCL[c] * CMULT[c]) then begin
60        if b_exceed=0 then begin
61          ECountT.PWS := 1;
62          ECountT.PWSPop := P.Population;
63          ECountT.EP := ECountT.EP + 1;
64          ECountT.EPPop := ECountT.EPPop + CurPop;
65        end;
66        b_exceed:=1;
67      end;
68      if (CurConc[c] > MCL[c]/2) and (CurConc[c] < MCL[c]) and (MCL[c]<1e30) then begin
69        if b_CntAnnual=0 then begin
70          ACountT.PWS := 1;
71          ACountT.PWSPop := P.Population;
72          ACountT.EP := ACountT.EP + 1;
73          ACountT.EPPop := ACountT.EPPop + CurPop;
74        end;
75        b_CntAnnual:=1;
76      end;
77
78    end;
79    //adhoc counts for icr -  pop < 3300
80    if (P.StatePFAS=0) and (P.SystemSize < 4) then begin
81      EPY1_PFE0 := EPY1_PFE0 + 1;
82      if b_nondetect = 0 then EPY1_ND := EPY1_ND + 1;
83      if b_exceed = 1 then EPY1_E := EPY1_E + 1;
84    end;
85
86    {$ifdef DEBUG}
87    if _d then begin
88      FData.DebugLog.LD('costs','b_exceed', b_exceed);
89      FData.DebugLog.LD('costs','b_nondetect', b_nondetect);
90      FData.DebugLog.LD('costs','b_annual', b_annual);
91      FData.DebugLog.LD('costs','b_quarterly', b_quarterly);
92    end;
93    {$endif}
94
95    if (b_exceed <> 0) then begin
96      PerformanceGACIX;
97      var Tech := SelectTech(P);
98      if Tech < 0 then begin
99        FData.ErrLog.L(Format('No tech chosen: pwsid:%s, iteration:%d, option:%s',[P.PWSId, FIteration, FOptionName]));
100       exit;
101       //TODO what to do?
102     end;
103     TookAction := True;
104     GetWBSCosts(P, OnYrChange, TotYears, Tech);
105     //setting final concentrations based on Tech for this option
106     var tmp: double;
107     var ConcSet := false;
108     case Tech of
109       ttGAC  : begin
110                  b_treat := 1;
111                  for c := 0 to CMax do
112                    PR[c] := min((BV_GAC - A_TOC * cTOC - B_GAC[c]) / A_R, 0.99);
113                end;
114       ttIX   : begin
115                  b_treat := 1;
116                  for c := 0 to CMax do
117                    PR[c] := min((BV_IX - A_PFAS_IX * c0Total - B_IX[c]) / A_R_IX, 0.996);
118                end;
119       ttCRO  : begin
```

```
120                 b_treat := 1;
121                 tmp := min(PRM / 0.95,1); //B
122                 for c := 0 to CMax do
123                   if tmp = 1 then
124                     FinalConc[c] := (CurConc[c] * (1-PRM))
125                   else
126                     FinalConc[c] := (1-tmp) * CurConc[c] + tmp * (CurConc[c] * (1-0.95));
127                 ConcSet := true;
128               end;
129       ttPOU  : begin
130                 b_treat := 1;
131                 tmp := (c0Total - 70) / c0Total;
132                 if tmp<0 then tmp:=1;
133                 for c := 0 to CMax do
134                   PR[c] := tmp;
135               end;
136       else     begin
137                 b_source := 1;
138                 //remove everything for other technologies
139                 for c := 0 to CMax do
140                   PR[c] := 1;
141               end;
142     end;
143     if not ConcSet then begin
144       for c := 0 to CMax do
145         FinalConc[c] := CurConc[c] * (1 - max(PR[c],0));
146     end;
147
148     //make sure everything is under MCL
149     var OkMCL := true;
150     for c := 0 to CMax do begin
151        if FinalConc[c] > MCL[c] * CMult[c] then begin
152          OkMCL:=False;
153          break;
154        end;
155     end;
156
157     if not OkMCL then begin
158       FData.ErrLog.L('MCL Error:'+fOptionName);
159       FData.ErrLog.LD('tech',FData.TechForecast.TechToText(Tech));
160       FData.ErrLog.LD('df', Inf.DesignFlow);
161       FData.ErrLog.LD('af', Inf.AverageFlow);
162       FData.ErrLog.LD('toc', cTOC);
163       FData.ErrLog.LD('bv_gac', BV_GAC);
164       FData.ErrLog.LD('bv_ix', BV_IX);
165       for c := 0 to CMax do begin
166          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
167          FData.ErrLog.LD(FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
168          FData.ErrLog.LD(' MCL',MCL[c]);
169          FData.ErrLog.LD(' PR',PR[c]);
170       end;
171       FData.ErrLog.L('*****');
172     end;
173
174     {$ifdef DEBUG}
175     if _d then begin
176       FData.DebugLog.L('costs','TECH');
177       FData.DebugLog.LD('costs','tech',FData.TechForecast.TechToText(Tech));
178       FData.DebugLog.LD('costs','df', Inf.DesignFlow);
179       FData.DebugLog.LD('costs','af', Inf.AverageFlow);
180       FData.DebugLog.LD('costs','toc', cTOC);
181       FData.DebugLog.LD('costs','bv_gac', BV_GAC);
182       FData.DebugLog.LD('costs','bv_ix', BV_IX);
183       for c := 0 to CMax do begin
184          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' final conc',FinalConc[c]);
185          FData.DebugLog.LD('costs',FData.Occ.ContamList[c] + ' begin conc',CurConc[c]);
186          FData.DebugLog.LD('costs','MCL',MCL[c]);
187          FData.DebugLog.LD('costs','PR',PR[c]);
188       end;
189     end;
190     {$endif}
191
192
193   end;  //b_exceed<>0
194
```

```
195   var OnYr: integer;
196   var aflag2 := false;
197   var DidSample: double := 0;
198   for var Yr := FData.Cfg.StartYear to FData.Cfg.EndYear do begin
199     OnYr := Yr  - FData.Cfg.StartYear;
200
201     if OnYr = 0 then begin   ///years 1,2,3
202       b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
203     end else
204     if OnYr = 3 then begin //3,4,5 years 4,5,6
205       b_quarterly:=0;      b_triennial := 0;      b_annual:=0;
206       for c := 0 to CMax do begin
207         if (CurConc[c] > MCL[c]/2) then
208           b_quarterly := 1;
209       end;
210       if b_quarterly = 0 then begin  // all others - triennial
211          b_triennial := 1;
212       end;
213     end else
214     if OnYr = 6 then begin //year 7 - one year after treatment
215        //switch quarterly to annual in year 7 (all treating systems with get to between 1/2*mcl and mcl
216        if b_quarterly=1 then begin
217          b_quarterly := 0;
218          b_annual := 1;
219        end;
220     end;
221
222
223     if (FData.Cfg.SingleCostYear <> 0) and (Yr <> FData.Cfg.SingleCostYear) then continue;
224
225     if FreshPWS then
226       csOption._EvaluateSystem(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
227
228     csOption._Evaluate(P, OnYr, @FData.Inp, FData.Cfg.PeriodAdjust);
229     {$ifdef DEBUG}
230     (*
231     if (aFlag = false) and (b_nondetect_all_ep=1) and (P.SystemSize=2) and
232        (P.SourceWater = swGroundwater) and (FIteration=1)  then begin
233
234       var TT := csOption._DumpCalcs;
235       FData.DebugLog.L('costmicro','***WORKBOOK BY YEAR: '+OnYr.ToString+'  PWS:'+P.PWSId+'  b_nondetect:'+b_nondetect.ToString   );
236       for var tti := 0 to TT.Count-1 do
237         FData.DebugLog.L('costmicro',TT[tti]);
238       TT.Free;
239       aFlag2:=True;
240
241     end;
242     *)
243     {$endif}
244
245     if _T[0].PWSSampling-DidSample >0 then begin
246       if b_triennial = 1 then
247         DidSmpT[OnYr] := DidSmpT[OnYr] + 1
248       else
249         DidSmpO[OnYr] := DidSmpO[OnYr] + 1
250     end;
251     DidSample := _T[0].PWSSampling;
252
253   end;
254   aFlag:=aFlag2;
255
256   FreshPWS := False;
257   //accumulate to the system for this entry point...
258   for var i := 0 to fData.Inp.RateCount-1 do begin
259     _i(O[i].AgencyImpAdmin,_T[i].AgencyImpAdmin);
260     _i(O[i].AgencyTreatmentAdmin, _T[i].AgencyTreatmentAdmin);
261     _i(O[i].PWSImpAdmin,_T[i].PWSImpAdmin);
262     _i(O[i].AgencyImpSampling,_T[i].AgencyImpSampling);
263     _i(O[i].PWSSampling,_T[i].PWSSampling);
264     _i(O[i].PWSImpAdmin_pws,_T[i].PWSImpAdmin_pws);
265     _i(O[i].PWSSampling_pws,_T[i].PWSSampling_pws);
266     _i(O[i].PWSTreatmentAdmin, _T[i].PWSTreatmentAdmin);
267     _i(O[i].PWSTreatmentAdmin_pws, _T[i].PWSTreatmentAdmin_pws);
268
```

```
    _i(O[i].PWSTreatment,_T[i].PWSTreatment);
    _i(O[i].PWSTreatment_pws,_T[i].PWSTreatment_pws);
    _i(O[i].CapitalCost,_T[i].CapitalCost);
    _i(O[i].CapitalCost_pws,_T[i].CapitalCost_pws);
    _i(O[i].OMCost,_T[i].OMCost);
    _i(O[i].OMCost_pws,_T[i].OMCost_pws);
  end;
end;


constructor TcmPFASOptionMCL1.Create(aData: PSWModelData;
  GlbUnc: TGlobalUncertainty; aNeededContams, aOptionName: string;
  aIteration: integer);
begin
  inherited;
  ixPFOS := FData.Occ.ContamList.IndexOf('PFOS');
  ixPFOA := FData.Occ.ContamList.IndexOf('PFOA');
  setlength(DidSmpT, FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
  setlength(DidSmpO, FData.Cfg.EndYear - FData.Cfg.StartYear + 1);
end;

procedure TcmPFASOptionMCL1.SetPWSWideEPValues(const P: TPWSRecord);
begin
  inherited;
  ECountT := Default(TOCOunt);
  ACountT := Default(TOCOunt);
  NDCountT := Default(TOCOunt);
  EPY1_PFE0:=0;
  EPY1_ND:=0;
  EPY1_E:=0;

  for var i := 0 to High(DidSmpT) do begin
    DidSmpT[i] := 0;
    DidSmpO[i] := 0;
  end;
end;

procedure TcmPFASOptionMCL1.SetupModule;
begin
  inherited;
  var ix : integer;
  MCLSafetyFactor := 0.8;

  //reset everything
  for var i := 0 to CMax do begin
    MCL[i] := 1e30;
    DL[i]:=0;
    CMULT[i] := 1;
  end;

  if CompareText(fOptionName,'PFASMCL-PFNA')=0 then begin
    ix := FData.Occ.ContamList.IndexOf('PFNA');
    if ix>-1 then MCL[ix] := 15;
    if ix>-1 then DL[ix] := 0.7;
  end else
  if CompareText(fOptionName,'PFASMCL-HFPODA')=0 then begin
    ix := FData.Occ.ContamList.IndexOf('HFPODA');
    if ix>-1 then MCL[ix] := 15;
    if ix>-1 then DL[ix] := 1.9;
  end else
  if CompareText(fOptionName,'PFASMCL-PFHXS')=0 then begin
    ix := FData.Occ.ContamList.IndexOf('PFHXS');
    if ix>-1 then MCL[ix] := 15;
    if ix>-1 then DL[ix] := 1.4;
  end else
    raise exception.Create('Bad Name in OptionMCL1');


  //for option 1, the MCLs are under 70ppt so POU is eliminated
  NoPOU := true;
end;

procedure TcmPFASOptionMCL1.SetupOutputs(Metrics: TCategoryMetrics);
begin
  inherited;
```

```
  Metrics.AddMetric(fOptionName+'.PFASPWS',
                    'PWSs with PFAS Exceedance (Quarterly Samples)', @ECountT.PWS, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASPWSPOP',
                    'PWS Population Impacted by PFAS Exceedance (Quarterly Samples)', @ECountT.PWSPop, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASEP',
                    'EPs with PFAS Exceedance (Quarterly Samples)', @ECountT.EP, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASEPPOP',
                    'EPs Population Impacted by PFAS Exceedance (Quarterly Samples)', @ECountT.EPPop, mtInfo, mcSumOnly, False);


  Metrics.AddMetric(fOptionName+'.PFASPWS_A',
                    'PWSs with PFAS Annual Sampling', @ACountT.PWS, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASPWSPOP_A',
                    'PWS Population Impacted by PFAS Annual Sampling', @ACountT.PWSPop, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASEP_A',
                    'EPs with PFAS Annual Sampling', @ACountT.EP, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASEPPOP_A',
                    'EPs Population Impacted by PFAS Annual Sampling', @ACountT.EPPop, mtInfo, mcSumOnly, False);


  Metrics.AddMetric(fOptionName+'.PFASPWS_ND',
                    'PWSs with PFAS Detects', @NDCountT.PWS, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASPWSPOP_ND',
                    'PWS Population Impacted by PFAS Detects', @NDCountT.PWSPop, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASEP_ND',
                    'EPs with PFAS Detects', @NDCountT.EP, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.PFASEPPOP_ND',
                    'EPs Population Impacted by PFAS Detects', @NDCountT.EPPop, mtInfo, mcSumOnly, False);

  Metrics.AddMetric(fOptionName+'.EP_quarterly_yr1',
                      'All EPs in PWS pop < 3300 and with STATEPFAS = 0', @EPY1_PFE0, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EP_exceed_yr1',
                      'Number of EP in PWS pop < 3300 and with STATEPFAS = 0 above MCL for any contaminant in year 1', @EPY1_E, mtInfo, mcSumOnly, False);
  Metrics.AddMetric(fOptionName+'.EP_ nondetect_yr1',
                      'Number of EP in PWS pop < 3300 and with STATEPFAS = 0 above any contaminant detection level in year 1', @EPY1_ND, mtInfo, mcSumOnly, False);

(*
  var s: string;
  for var i := 0 to high(DidSmpT) do begin
    if i<9 then s := '0'+inttostr(i+1) else s:= inttostr(i+1);
    Metrics.AddMetric(fOptionName+'.NumEPWithSamplingCostY'+s+'_Tri',
                        'EPs Tri Sampling in year '+inttostr(i+1), @DidSmpT[i], mtInfo, mcSumOnly, False);
    Metrics.AddMetric(fOptionName+'.NumEPWithSamplingCostY'+s+'_Oth',
                        'EPs Other Sampling in year '+inttostr(i+1), @DidSmpO[i], mtInfo, mcSumOnly, False);

  end;
*)
end;

end.
```

code\PBPK\PFASPBPK.pas

```
unit PFASPBPK;

interface

uses Classes, Generics.Collections, Math, Spliner, SysUtils,
     //SafewaterGlobals just supports the direct inserting of data into safewater type arrays - can be removed
     //if RunSafeWaterStyle function is removed
     SafewaterGlobals;

type
  TBWArray = array [0..35,0..1] of double;
  PBWArray = ^TBWArray;

const
  //Body weights from ORD r code...  TODO document
  iBWF: TBWArray =
```

```
17    ((0,3.396435241),(0.246498599,5.555038682),(0.49859944,7.21368214),(0.743977591,8.500857009),(0.999439776,9.530949046),
18    (1.251540616,10.33244965),(1.496918768,10.99105309),(1.749019608,11.54969655),(1.994397759,12.05115713),(2.246498599,12.5240863),
19    (2.501960784,12.99703548),(2.747338936,13.39849607),(2.999439776,13.87142524),(3.986622074,15.95744681),(4.969899666,18.08510638),
20    (5.993311037,20.42553191),(6.996655518,22.9787234),(7.97993311,25.74468085),(8.983277592,29.04255319),(9.986622074,32.9787234),
21    (10.98996656,37.34042553),(12.03344482,41.91489362),(12.99665552,45.85106383),(14.02006689,49.46808511),(15.00334448,52.12765957),
22    (16.00668896,54.04255319),(17.03010033,55.21276596),(18.01337793,56.27659574),(19.01672241,57.34042553),
23    (20,58.29787234),(25,67.9),(35,70.2),(45,72.7),(55,73.6),(65,73.9),(75,69));
24
25    iBWM: TBWArray =
26    ((0,3.557374014),(0.249692943,6.068626175),(0.500186659,7.904900114),(0.75201612,9.286892748),(0.997443138,10.30206018),
27    (1.246340047,11.09542632),(1.500303656,11.72136754),(1.745730674,12.2278773),(1.997560135,12.66828842),(2.249389596,13.08310708),
28    (2.494816613,13.48724702),(2.746646074,13.91697941),(2.998475535,14.34137243),(3.985356373,16.1637931),(5.00382928,18.40517241),
29    (5.986607143,20.71659483),(6.987242149,23.02801724),(8.005830511,25.68965517),(9.006619458,28.56142241),(10.0075431,31.92349138),
30    (10.99076355,35.84590517),(11.99201432,40.3987069),(13.01129541,45.58189655),(14.01277709,50.97521552),(15.01425877,56.36853448),
31    (16.01552879,60.99137931),(17.01649092,64.49353448),(18.01724138,67.22521552),(18.99990379,69.11637931),(20,70.58728448),
32    (25,81.2),(35,84),(45,87.4),(55,86.9),(65,82.1),(75,75.1));
33
34  type
35    TBodyWeights = class
36    private
37      FMethod: integer;
38      procedure SetMethod(value: integer);
39    published
40      BWF, BWM: TSpliner;
41      constructor create;
42      destructor Destroy; override;
43      function GetWeight(const aSex: integer; const aAge: double) : double;
44      property Method: integer read FMethod write SetMethod;
45    end;
46
47    TPFASPBPKParms=record
48      CL,Vd,CLr,P_milk,r_f_m,pow_milk,
49      t_con, F_abs, n_i, food_dose, pow_dose,
50      t_gest, t_lact, d_m, A_m_init : double;
51      procedure Defaults;
52    end;
53
54    TPFASPBPKEq=record
55      A_mf,A_i,AUC_m,AUC_i,d_m,d_i,T_in,T_out,CLr : double;
56    end;
57
58    TPFASPBPKOut=record
59      C_m,C_i,M_mf,M_i,D_m,D_i,R_milk,r_milk_i,
60      r_m_mf,A_bal : double;
61    end;
62
63    TPFASPBPKOutput=record
64      Year : integer;
65      Age: double;
66      Data: TPFASPBPKOut;
67    end;
68
69    TPFASPBPKOutputs =  TList<TPFASPBPKOutput>;
70
71    TValueChange = record
72      Year: double;
73      Value: double;
74    end;
75
76    TPFASPBKRun = class
77      //age,year etc...
78      BirthYear,StartYear,EndYear,PFASStartYear : integer;
79      TimeStep: double; //days
80      OutputFreq: double;  // how often to output in days
81      DoseChange,
82      CLrChange: TList<TValueChange>;
83      Sex, Race: integer;
84      MaxAge: double;
85      constructor create;
86      destructor Destroy; override;
87
88      procedure Defaults;
89      procedure DefaultORDClrChange(P : TPFASPBPKParms);
90    end;
91
```

```
  TPFASPBPKBase=class(TObject)
  private
    Parms: TPFASPBPKParms;
    BW: TBodyWeights;
    CustomBodyWeights: PSubPopArray;
    UseCustomBW: boolean;


    procedure SetYearlyChanges(const OnYr: double; const onCalYr: integer; var Y :TPFASPBPKEq;
                               var OnDoseChange, OnClrChange: integer; const S: TPFASPBKRun);
    procedure AddToResult(Output : TPFASPBPKOutputs; const S: TPFASPBKRun; Data:TPFASPBPKOut; const OnTS: double);
    procedure GetState(const TS: double; var Y: TPFASPBPKEq; var YOut: TPFASPBPKOut; const S: TPFASPBKRun);
    procedure SetState(var Y: TPFASPBPKEq; const YDot: TPFASPBPKEq; const S: TPFASPBKRun);
    procedure GetDerivatives(var YDot: TPFASPBPKEq; const Y: TPFASPBPKEq; const YOut: TPFASPBPKOut);
  public
    Name: string;

    procedure SetCustomBodyWeights(P: PSubPopArray);
    function Run(const S: TPFASPBKRun) : TPFASPBPKOutputs;
    procedure RunSafeWaterStyle(var Result: TAgeArray; const S: TPFASPBKRun);

    function GetBodyWeight(const aAge: double; const aSex, aRace: integer) : double;
    constructor create;
    destructor Destroy; override;
  end;

  TPFOAPBPK = class(TPFASPBPKBase)
    constructor create;
  end;

  TPFOSPBPK = class(TPFASPBPKBase)
    constructor create;
  end;


implementation

{ TPFASPBPKBase }

procedure TPFASPBPKBase.AddToResult(Output: TPFASPBPKOutputs; const S: TPFASPBKRun; Data: TPFASPBPKOut; const OnTS: double);
begin
  var T:TPFASPBPKOutput;
  var OnYr:integer := trunc(OnTS / 365.25);
  T.Year := OnYr + S.BirthYear;
  if T.Year<S.StartYear then exit;
  T.Age := T.Year - S.BirthYear;
  T.Data := Data;
  Output.Add(T);
end;

constructor TPFASPBPKBase.create;
begin
  Parms := Default(TPFASPBPKParms);
  //initial values from init_human_p Sim_functions.r
  Parms.t_gest := 0.75 * 365;
  Parms.t_lact := 1.0 * 365;
  Parms.CL := 0.066;
  Parms.CLr := 0.095;
  Parms.Vd := 380;
  Parms.P_milk := 0.03;
  Parms.r_f_m := 1.0;
  Parms.d_m := 0;
  Parms.pow_milk := 0.0;
  Parms.A_m_init := 0;

  Parms.Defaults;
  BW := TBodyWeights.create;
  BW.Method := smLinear;
  UseCustomBW:=False;
end;

destructor TPFASPBPKBase.destroy;
begin
```

```
166   BW.Free;
167   inherited;
168 end;
169
170 function TPFASPBPKBase.GetBodyWeight(const aAge: double; const aSex, aRace: integer): double;
171 begin
172   //TODO turn custom body weights into spline
173   if UseCustomBW then
174     result := CustomBodyWeights[aSex, aRace, trunc(aAge)]
175   else
176     result := BW.GetWeight(aSex,aAge);
177 end;
178
179 procedure TPFASPBPKBase.GetDerivatives(var YDot: TPFASPBPKEq; const Y: TPFASPBPKEq; const YOut: TPFASPBPKOut);
180 begin
181   var k_mf: double;
182   if y.CLr > 0 then
183     k_mf := Y.CLr / Parms.Vd
184   else
185     k_mf := Parms.CL / Parms.Vd;
186   var k_i := Parms.CL / Parms.Vd ;
187
188   ydot.A_mf := yout.D_m - Parms.n_i * yout.r_milk_i - k_mf * y.A_mf;
189   ydot.A_i := yout.D_i + yout.r_milk_i - k_i * y.A_i ;
190   ydot.AUC_m := yout.C_m;
191   ydot.AUC_i := yout.C_i;
192   ydot.d_m := 0.0;
193   ydot.d_i := 0.0;
194   ydot.CLr := 0.0;
195   ydot.T_in := yout.D_m + Parms.n_i * yout.D_i;
196   ydot.T_out := k_mf * y.A_mf + Parms.n_i * k_i * y.A_i;
197 end;
198
199 procedure TPFASPBPKBase.GetState(const TS: double; var Y: TPFASPBPKEq; var YOut: TPFASPBPKOut; const S: TPFASPBKRun);
200 begin
201
202    var A := TS / 365.25;
203    YOut.M_mf := GetBodyWeight(A, S.Sex,S.Race);
204
205    //infant stuff - not in human adult...
206    YOut.M_i := 0;     //M_mf_in df
207    Y.d_i := 0;        //M_i_in df
208    YOut.R_milk := 0;  //R_milf_in df
209
210    YOut.R_milk := YOut.R_milk * power (YOut.M_i , Parms.pow_milk ) ;
211    //This can be calcd based on Age see ORD code...
212    YOut.r_m_mf := 1;
213
214    if YOut.M_i>0 then
215      YOut.C_m := Y.A_mf / ( Parms.Vd * YOut.M_mf )
216    else
217      YOut.C_m := (Y.A_mf * YOut.r_m_mf) / ( Parms.Vd * YOut.M_mf ) ;
218
219    if YOut.M_i>0 then
220      YOut.C_i := Y.A_i / ( Parms.Vd * YOut.M_i)
221    else
222      YOut.C_i := Parms.r_f_m * yout.C_m;
223
224    if TS<Parms.t_con then YOut.C_i := 0;
225
226    var fd: double;
227    if parms.food_dose>0 then
228      fd := 0.65 * power(yout.M_mf, 0.7919)
229    else
230      fd := power( yout.M_mf , Parms.pow_dose );
231
232    yout.D_m := Parms.F_abs * y.d_m * fd;
233
234    if parms.food_dose>0 then
235      fd := 0.65 * power(yout.M_i, 0.7919)
236    else
237      fd := power( yout.M_i , Parms.pow_dose );
238
239   yout.D_i := Parms.F_abs * y.d_i * fd;
240   yout.A_bal := y.T_in - y.A_mf - Parms.n_i * y.A_i - y.T_out;
```

```
  yout.r_milk_i := yout.R_milk * Parms.P_milk * yout.C_m;
end;

function TPFASPBPKBase.Run(const S: TPFASPBKRun): TPFASPBPKOutputs;
begin
  Result:=TPFASPBPKOutputs.Create;
  //Note: COmmented out since the Clearance parameters are currently the same
  //if S.Sex = 1 then S.DefaultORDClrChange(Parms);

  var EndTS: double := (S.EndYear - S.BirthYear + 1)* 365.25;
  var Y:TPFASPBPKEq := Default(TPFASPBPKEq);
  var YDot:TPFASPBPKEq := Default(TPFASPBPKEq);
  Y.A_mf := Parms.A_m_init;
  Y.d_m := Parms.d_m;
  var YOut:TPFASPBPKOut := Default(TPFASPBPKOut);

  var OnTS: double := 0;
  var TSCnt:double:= S.OutputFreq/2;
  var OnYr: double;
  var OnCalYr: integer;
  var OnDoseChange := 0;
  var OnClrChange := 0;

  while OnTS <= EndTS do begin
    OnYr := OnTS / 365.25;
    OnCalYr := S.BirthYear + trunc(OnYr);
    if OnCalYr > S.EndYear then break;
    if OnYr > S.MaxAge + 1 then break;

    SetYearlyChanges(OnYr,OnCalYr, Y, OnDoseChange, OnClrChange, S);

    GetState(OnTS,Y,Yout, S);
    GetDerivatives(YDot,Y,Yout);
    SetState(Y,YDot,S);
    if TSCnt >= S.OutputFreq  then begin
      AddToResult(Result,S,YOut,OnTS);
      TSCnt := 0;
      sleep(0);
    end;

    OnTs := OnTs + s.TimeStep;
    TSCnt := TSCnt + s.TimeStep;
  end;

end;

procedure TPFASPBPKBase.RunSafeWaterStyle(var Result: TAgeArray; const S: TPFASPBKRun);
begin
  //Note: COmmented out since the Clearance parameters are currently the same
  //if S.Sex = 1 then S.DefaultORDClrChange(Parms);
  fillchar(Result, sizeof(Result),0);
  var EndTS: double := (S.EndYear - S.BirthYear + 1)* 365.25;
  var Y:TPFASPBPKEq := Default(TPFASPBPKEq);
  var YDot:TPFASPBPKEq := Default(TPFASPBPKEq);
  Y.A_mf := Parms.A_m_init;
  Y.d_m := Parms.d_m;
  var YOut:TPFASPBPKOut := Default(TPFASPBPKOut);

  var OnTS: double := 0;
  var TSCnt:double:= S.OutputFreq/2;
  var OnYr: double;
  var OnCalYr: integer;
  var OnDoseChange := 0;
  var OnClrChange := 0;
  var OutCnt:integer:=0;

  while OnTS <= EndTS do begin
    OnYr := OnTS / 365.25;
    OnCalYr := S.BirthYear + trunc(OnYr);
    if OnCalYr > S.EndYear then break;
    //don't caculate past 85 for now - can be extended
    if OnYr > S.MaxAge + 1 then break;
    SetYearlyChanges(OnYr,OnCalYr, Y, OnDoseChange, OnClrChange, S);

```

```
315     GetState(OnTS,Y,Yout, S);
316     GetDerivatives(YDot,Y,Yout);
317     SetState(Y,YDot, S);
318     if TSCnt >= S.OutputFreq  then begin
319       Result[OutCnt] := YOut.C_m;
320       TSCnt := 0;
321       inc(outcnt);
322       sleep(0);
323     end;
324
325     OnTs := OnTs + s.TimeStep;
326     TSCnt := TSCnt + s.TimeStep;
327   end;
328 end;
329
330 procedure TPFASPBPKBase.SetCustomBodyWeights(P: PSubPopArray);
331 begin
332   if P = nil then
333     UseCustomBW:=False
334   else begin
335     UseCustomBW:=true;
336     CustomBodyWeights := p;
337   end;
338 end;
339
340 procedure TPFASPBPKBase.SetState(var Y: TPFASPBPKEq; const YDot: TPFASPBPKEq; const S: TPFASPBKRun);
341 begin
342   Y.A_mf := Y.A_mf + (YDot.A_mf * S.TimeStep);
343   Y.A_i := Y.A_i + (YDot.A_i * S.TimeStep);
344   Y.AUC_m := Y.AUC_m + (YDot.AUC_m * S.TimeStep);
345   Y.AUC_i := Y.AUC_i + (YDot.AUC_i * S.TimeStep);
346   Y.d_m := Y.d_m + (YDot.d_m * S.TimeStep);
347   Y.d_i := Y.d_i + (YDot.d_i* S.TimeStep);
348   Y.T_in := Y.T_in + (YDot.T_in * S.TimeStep);
349   Y.T_out := Y.T_out + (YDot.T_out * S.TimeStep);
350   Y.CLr := Y.CLr + (YDot.CLr * S.TimeStep);
351 end;
352
353 procedure TPFASPBPKBase.SetYearlyChanges(const OnYr: double; const onCalYr: integer; var Y :TPFASPBPKEq;
354            var OnDoseChange, OnClrChange: integer; const S: TPFASPBKRun);
355 begin
356
357   if (S.DoseChange.Count>0) and (OnDoseChange<S.DoseChange.Count) then begin
358     while OnCalYr >= S.DoseChange[OnDoseChange].Year do begin
359       Y.d_m := S.DoseChange[OnDoseChange].Value;
360       inc(OnDoseChange);
361       if OnDoseChange>=S.DoseChange.Count then break;
362     end;
363   end;
364
365   Y.CLr := 0;
366   if (S.CLrChange.Count>0) and (OnClrChange<S.CLrChange.Count) then begin
367     if OnYr>=S.CLrChange.items[OnClrChange].Year then begin
368       Y.CLr := S.CLrChange.items[OnClrChange].Value;
369       inc(OnClrChange);
370     end;
371   end;
372
373 end;
374
375 { TPFOAPBPK }
376
377 constructor TPFOAPBPK.create;
378 begin
379   inherited create;
380   //Set these here by PFAS so we can apply uncertainty with the engine
381   Parms.Cl := 0.120/1000;
382   Parms.Vd :=0.170;
383   Parms.CLr := 0.120/1000;
384   Parms.P_milk := 0.0502;
385   Parms.r_f_m := 0.792;
386   Parms.pow_milk := 1;
387   Name:='PFOA';
388 end;
389
```

```
390 { TPFOSPBPK }
391
392 constructor TPFOSPBPK.create;
393 begin
394   inherited;
395   Parms.Cl := 0.128/1000;
396   Parms.Vd :=0.230;
397   Parms.CLr := 0.128/1000;
398   Parms.P_milk := 0.0160;
399   Parms.r_f_m := 0.421;
400   Parms.pow_milk := 1;
401   Name:='PFOS';
402 end;
403
404 { TPFASPBKRun }
405
406 constructor TPFASPBKRun.create;
407 begin
408   ClrChange := TList<TValueChange>.create;
409   DoseChange := TList<TValueChange>.create;
410 end;
411
412 procedure TPFASPBKRun.DefaultORDClrChange(P : TPFASPBPKParms);
413 begin
414   CLrChange.Clear;
415   var T: TValueChange;
416   T.Year := 12.4;
417   T.Value := P.CLr;
418   ClrChange.Add(T);
419   T.Year := 24.25;
420   T.Value := 0;
421   ClrChange.Add(T);
422   T.Year := 26;
423   T.Value := P.CLr;
424   ClrChange.Add(T);
425   T.Year := 50;
426   T.Value := 0;
427   ClrChange.Add(T);
428 end;
429
430 procedure TPFASPBKRun.Defaults;
431 begin
432   BirthYear:=1980;
433   StartYear:=2023;
434   EndYear:=2055;
435   PFASStartYear:=1950;
436   TimeStep:=1; //days
437   OutputFreq := 365.25;
438   Sex := 1;
439   Race:=reWhite;
440   MaxAge:=cMaxAge;
441 end;
442
443 destructor TPFASPBKRun.destroy;
444 begin
445   ClrChange.Free;
446   DoseChange.Free;
447   inherited;
448 end;
449
450 { TPFASPBPKParms }
451
452 procedure TPFASPBPKParms.Defaults;
453 begin
454   t_con := 24.25 * 365; // Age (d) of conception.
455   t_gest := 0.75 * 365; // Duration (d) of pregnancy/gestation.
456   t_lact := 1.0 * 365;  // Duration (d) of lactation/nursing.
457   CL := 0.066;   // Clearance (mL/kg/d) of substance in this species.
458   CLr := 0.095;  // Clearance (mL/kg/d) for reproductive-age women
459   Vd := 380;     // Volume of distribution (mL/kg) of substance in this species.
460   P_milk := 0.03; // Milk/maternal blood concentration partition coefficient.
461   r_f_m := 1.0;   // Ratio of fetal and maternal concentrations.
462   F_abs := 1.0;   // Fraction of nominal dose that is absorbed.
463   food_dose := 0;  // A boolean flag. Dose is delivered in food if nonzero.
```

```
  d_m := 0;        // Dose rate (mg/kg/d or mg/kg) to mother.
  pow_dose := 1.0; // When 1, d_m is mg/kg/d, when 0, d_m is mg/d
  pow_milk := 0.0; // when 1, milk transfer rate is kg/kg/d, when 0, is kg/d
  A_m_init := 0;
  n_i := 1;
end;

{ TBodyWeights }

constructor TBodyWeights.create;
begin
  BWF := TSpliner.Create;
  BWM := TSpliner.Create;
  var n := high(iBWF);
  BWF.SetDataLength(n+1);
  for var i := 0 to n do begin
    BWF.AddData(iBWF[i,0],iBWF[i,1]);
  end;
  n := high(iBWM);
  BWM.SetDataLength(n+1);
  for var i := 0 to n do begin
    BWM.AddData(iBWM[i,0],iBWM[i,1]);
  end;
end;

destructor TBodyWeights.destroy;
begin
  BWF.Free;
  BWM.Free;
  inherited;
end;

function TBodyWeights.GetWeight(const aSex: integer;  const aAge: double): double;
begin
  if aSex = 0 then
    Result := BWM.SplineValue(aAge)
  else
    Result := BWF.SplineValue(aAge)
end;

procedure TBodyWeights.SetMethod(value: integer);
begin
  FMethod:=Value;
  BWM.Method := value;
  BWF.Method := value;
end;

end.

```

code\PBPK\Spliner.pas

```
unit Spliner;

interface

uses Classes, Math, SysUtils;

const
  smLinear = 0;
  smCubicSpline = 1;

type
  TDoubleArray = array of double;

  TSpliner = class
  private
    XData, YData : TDoubleArray;
    Fd2Y_dX2: TDoubleArray;
    FNoPts, PtCounter : integer;
    HaveDerivative: boolean;
    FMethod: integer;
    procedure SecondDerivative;
```

```
22    public
23      constructor create;
24
25      procedure SetDataLength(N : integer);
26      //data must be sorted by x as it arrives
27      procedure AddData(X,Y : double);
28      function SplineValue(X: Double): Double;
29
30      property Method: integer read FMethod write FMethod;
31    end;
32
33  implementation
34
35  { TSpliner }
36
37  procedure TSpliner.AddData(X, Y: double);
38  begin
39    HaveDerivative := false;
40    XData[PtCounter] := X;
41    YData[PtCounter] := Y;
42    inc(PtCounter);
43  end;
44
45  function TSpliner.SplineValue(X: Double): Double;
46  var
47    i, iLeft, iRight: Integer;
48    dX, LeftX, RightX: double;
49  begin
50    if (not HaveDerivative) and (FMethod = smCubicSpline) then
51      SecondDerivative;
52    iLeft := 0;
53    iRight := FNoPts - 1;
54    while (iRight - iLeft > 1) do  begin
55      i := (iRight + iLeft) div 2;
56      if (XData[i] > X) then
57        iRight := i
58      else
59        iLeft := i;
60    end;
61
62    dX := XData[iRight] - XData[iLeft];
63
64    if (dX = 0.0) then
65      raise ERangeError.CreateFmt('SplineValue: bad input data (dX = 0) !'
66        + 'XData[%d] = %g, XData[%d] = %g', [iRight, XData[iRight], iLeft,
67        XData[iLeft]]);
68
69    if FMethod=smLinear then begin
70      //use last value if outside range.....
71      if X>=XData[FNoPts-1] then
72        Result := YData[FNoPts-1]
73      else begin
74        var px := (X - XData[iLeft]) / dx;
75        Result := YData[iLeft] + (YData[iRight] - YData[iLeft]) * px;
76      end;
77      exit;
78    end;
79
80    RightX := (XData[iRight] - X) / dX;
81    LeftX := (X - XData[iLeft]) / dX;
82
83    Result := RightX * YData[iLeft] + LeftX * YData[iRight] +
84      ((IntPower(RightX, 3) - RightX) * Fd2Y_dX2[iLeft] +
85      (IntPower(LeftX, 3) - LeftX) * Fd2Y_dX2[iRight]) * Sqr(dX) / 6.0;
86  end;
87
88  constructor TSpliner.create;
89  begin
90    FMethod := smCubicSpline;
91  end;
92
93  procedure TSpliner.SecondDerivative;
94  var
95    i: Integer;
96    TempVar, LeftXFraction: double;
```

```
  UpperTriangle: TDoubleArray;
begin

  SetLength(Fd2Y_dX2, FNoPts);
  SetLength(UpperTriangle, FNoPts);

  Fd2Y_dX2[0] := 0;
  UpperTriangle[0] := 0;
  for i := 1 to FNoPts - 2 do begin
    LeftXFraction := (XData[i] - XData[i - 1]) / (XData[i + 1] - XData[i - 1]);
    TempVar := LeftXFraction * Fd2Y_dX2[i - 1] + 2.0;
    Fd2Y_dX2[i] := (LeftXFraction - 1.0) / TempVar;
    UpperTriangle[i] := (YData[i + 1] - YData[i]) / (XData[i + 1] - XData[i]) -
      (YData[i] - YData[i - 1]) / (XData[i] - XData[i - 1]);
    UpperTriangle[i] := (6.0 * UpperTriangle[i] / (XData[i + 1] - XData[i - 1])
      - LeftXFraction * UpperTriangle[i - 1]) / TempVar;
  end;
  Fd2Y_dX2[FNoPts - 1] := 0;
  for i := FNoPts - 2 downto 0 do
  begin
    Fd2Y_dX2[i] := Fd2Y_dX2[i] * Fd2Y_dX2[i + 1] + UpperTriangle[i];
  end;
  UpperTriangle := nil;
  HaveDerivative:=True;
end;


procedure TSpliner.SetDataLength(N: integer);
begin
  FNoPts := N;
  SetLength(XData, FNoPts);
  SetLength(YData, FNoPts);
  PtCounter:=0;
  HaveDerivative:=False;
end;

end.

```

code\PBPK\pfaspk.dpr

```
program pfaspk;

uses
  Vcl.Forms,
  wtest in 'wtest.pas' {Form1},
  PFASPBPK in 'PFASPBPK.pas',
  Spliner in 'Spliner.pas',
  SafeWaterGlobals in '..\SafeWaterGlobals.pas',
  PFASSerumCalc in '..\BenefitModules\PFASSerumCalc.pas',
  SafeWaterConfig in '..\SafeWaterConfig.pas',
  SafeWaterUtility in '..\SafeWaterUtility.pas',
  SafeWaterCohortArray in '..\BenefitModules\SafeWaterCohortArray.pas';

{$R *.res}

begin
  ReportMemoryLeaksOnShutdown := true;
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.

```

code\PBPK\wtest.dfm

```
object Form1: TForm1
  Left = 0
  Top = 0
  Caption = 'Form1'
```

```
  ClientHeight = 682
  ClientWidth = 887
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -12
  Font.Name = 'Segoe UI'
  Font.Style = []
  DesignSize = (
    887
    682)
  TextHeight = 15
  object Label1: TLabel
    Left = 24
    Top = 642
    Width = 15
    Height = 15
    Anchors = [akLeft, akBottom]
    Caption = '***'
    ExplicitTop = 634
  end
  object Label2: TLabel
    Left = 8
    Top = 209
    Width = 139
    Height = 15
    Caption = 'Dose Changes (Year,Value)'
  end
  object Label3: TLabel
    Left = 24
    Top = 656
    Width = 15
    Height = 15
    Caption = '***'
  end
  object Button1: TButton
    Left = 8
    Top = 8
    Width = 75
    Height = 25
    Caption = 'Test'
    TabOrder = 0
    OnClick = Button1Click
  end
  object Chart1: TChart
    Left = 191
    Top = 320
    Width = 674
    Height = 329
    Legend.Visible = False
    Title.Text.Strings = (
      'TChart')
    BottomAxis.AxisValuesFormat = '###0.###'
    BottomAxis.Increment = 1.000000000000000000
    BottomAxis.LabelStyle = talValue
    BottomAxis.Title.Caption = 'Year'
    LeftAxis.Title.Caption = 'Plasma Concentration (mg/L)'
    View3D = False
    View3DWalls = False
    TabOrder = 1
    DefaultCanvas = 'TGDIPlusCanvas'
    ColorPaletteIndex = 13
    object Series1: TLineSeries
      HoverElement = [heCurrent]
      Shadow.Visible = False
      Brush.BackColor = clDefault
      Pointer.InflateMargins = True
      Pointer.Style = psRectangle
      XValues.Name = 'X'
      XValues.Order = loAscending
      YValues.Name = 'Y'
      YValues.Order = loNone
    end
    object Series2: TLineSeries
      HoverElement = [heCurrent]
```

```
80        SeriesColor = 33023
81        Shadow.Visible = False
82        Brush.BackColor = clDefault
83        Pointer.InflateMargins = True
84        Pointer.Style = psRectangle
85        XValues.Name = 'X'
86        XValues.Order = loAscending
87        YValues.Name = 'Y'
88        YValues.Order = loNone
89      end
90    end
91    object Button2: TButton
92      Left = 8
93      Top = 543
94      Width = 75
95      Height = 25
96      Caption = 'Time Test'
97      TabOrder = 2
98      OnClick = Button2Click
99    end
100   object Button3: TButton
101     Left = 8
102     Top = 583
103     Width = 75
104     Height = 25
105     Caption = 'Align Test'
106     TabOrder = 3
107     OnClick = Button3Click
108   end
109   object Chart2: TChart
110     Left = 191
111     Top = 8
112     Width = 674
113     Height = 306
114     Legend.Visible = False
115     Title.Text.Strings = (
116       'TChart')
117     BottomAxis.AxisValuesFormat = '###0.###'
118     BottomAxis.Increment = 1.000000000000000000
119     BottomAxis.LabelStyle = talValue
120     BottomAxis.Title.Caption = 'Year'
121     LeftAxis.Title.Caption = 'Body Weight'
122     View3D = False
123     View3DWalls = False
124     TabOrder = 4
125     DefaultCanvas = 'TGDIPlusCanvas'
126     ColorPaletteIndex = 13
127     object LineSeries1: TLineSeries
128       HoverElement = [heCurrent]
129       Shadow.Visible = False
130       Brush.BackColor = clDefault
131       Pointer.InflateMargins = True
132       Pointer.Style = psRectangle
133       XValues.Name = 'X'
134       XValues.Order = loAscending
135       YValues.Name = 'Y'
136       YValues.Order = loNone
137     end
138   end
139   object edTS: TLabeledEdit
140     Left = 64
141     Top = 112
142     Width = 33
143     Height = 23
144     EditLabel.Width = 52
145     EditLabel.Height = 23
146     EditLabel.Caption = 'Time Step'
147     LabelPosition = lpLeft
148     TabOrder = 5
149     Text = '1'
150   end
151   object edSex: TLabeledEdit
152     Left = 32
153     Top = 141
```

```
154     Width = 26
155     Height = 23
156     EditLabel.Width = 18
157     EditLabel.Height = 23
158     EditLabel.Caption = 'Sex'
159     LabelPosition = lpLeft
160     TabOrder = 6
161     Text = '1'
162   end
163   object edAge: TLabeledEdit
164     Left = 32
165     Top = 170
166     Width = 33
167     Height = 23
168     EditLabel.Width = 21
169     EditLabel.Height = 23
170     EditLabel.Caption = 'Age'
171     LabelPosition = lpLeft
172     TabOrder = 7
173     Text = '50'
174   end
175   object memDC: TMemo
176     Left = 8
177     Top = 230
178     Width = 145
179     Height = 89
180     Lines.Strings = (
181       '1960,1')
182     ScrollBars = ssVertical
183     TabOrder = 8
184   end
185   object memDC2: TMemo
186     Left = 8
187     Top = 342
188     Width = 145
189     Height = 89
190     Lines.Strings = (
191       '1960,1')
192     ScrollBars = ssVertical
193     TabOrder = 9
194   end
195   object Button4: TButton
196     Left = 8
197     Top = 496
198     Width = 75
199     Height = 25
200     Caption = 'Curves'
201     TabOrder = 10
202     OnClick = Button4Click
203   end
204   object Button5: TButton
205     Left = 8
206     Top = 448
207     Width = 75
208     Height = 25
209     Caption = 'SerumCalc'
210     TabOrder = 11
211     OnClick = Button5Click
212   end
213   object edC: TLabeledEdit
214     Left = 16
215     Top = 57
216     Width = 49
217     Height = 23
218     EditLabel.Width = 92
219     EditLabel.Height = 15
220     EditLabel.Caption = 'PFOA=1, PFOS=2'
221     TabOrder = 12
222     Text = '1'
223   end
224   object CheckBox1: TCheckBox
225     Left = 20
226     Top = 92
227     Width = 97
228     Height = 17
```

```
    Caption = 'Custom BW'
    TabOrder = 13
  end
  object edR: TLabeledEdit
    Left = 104
    Top = 141
    Width = 26
    Height = 23
    EditLabel.Width = 25
    EditLabel.Height = 23
    EditLabel.Caption = 'Race'
    LabelPosition = lpLeft
    TabOrder = 14
    Text = '0'
  end
  object Button6: TButton
    Left = 89
    Top = 496
    Width = 75
    Height = 25
    Caption = 'Age 10'
    TabOrder = 15
    OnClick = Button6Click
  end
  object Button7: TButton
    Left = 89
    Top = 8
    Width = 75
    Height = 25
    Caption = 'Test2'
    TabOrder = 16
    OnClick = Button7Click
  end
end

```

code\PBPK\wtest.pas

```
unit wtest;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, VclTee.TeeGDIPlus,
  VCLTee.TeEngine, VCLTee.Series, Vcl.ExtCtrls, VCLTee.TeeProcs, VCLTee.Chart,
  Vcl.Mask,PFASPBPK, StrUtils, System.Threading, SafewaterGlobals, PFASSerumCalc,
  SafewaterConfig, SafewaterCohortArray, Math;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Chart1: TChart;
    Series1: TLineSeries;
    Button2: TButton;
    Label1: TLabel;
    Button3: TButton;
    Chart2: TChart;
    LineSeries1: TLineSeries;
    edTS: TLabeledEdit;
    edSex: TLabeledEdit;
    edAge: TLabeledEdit;
    Label2: TLabel;
    memDC: TMemo;
    Label3: TLabel;
    Series2: TLineSeries;
    memDC2: TMemo;
    Button4: TButton;
    Button5: TButton;
    edC: TLabeledEdit;
    CheckBox1: TCheckBox;
    edR: TLabeledEdit;
    Button6: TButton;
```

```
36      Button7: TButton;
37      procedure Button1Click(Sender: TObject);
38      procedure Button2Click(Sender: TObject);
39      procedure Button3Click(Sender: TObject);
40      procedure Button4Click(Sender: TObject);
41      procedure Button5Click(Sender: TObject);
42      procedure Button6Click(Sender: TObject);
43      procedure Button7Click(Sender: TObject);
44    private
45      { Private declarations }
46      procedure SetSettings(var T : TPFASPBKRun; adiv : integer = 1);
47    public
48      { Public declarations }
49    end;
50
51  var
52    Form1: TForm1;
53
54  implementation
55
56  {$R *.dfm}
57
58
59  procedure TForm1.Button1Click(Sender: TObject);
60  begin
61    var fconv := 1;
62    var cfg := TSafewaterConfig.create;
63    var bw : TCohortData;
64    bw.LoadFromFile(cfg.BWFile);
65
66    var P:TPFASPBPKBase;
67    var OT: TStreamWriter;
68    if edC.Text='1' then begin
69      P := TPFOAPBPK.create;
70      OT := TStreamWriter.Create('pfoaage_'+edSex.Text+'.csv');
71    end else begin
72      P := TPFOSPBPK.create;
73      OT := TStreamWriter.Create('pfosage'+edSex.Text+'.csv');
74    end;
75    OT.WriteLine('age,s,c,bw');
76
77    if CheckBox1.Checked then
78      P.SetCustomBodyWeights(@BW.V);
79
80    var Settings := TPFASPBKRun.create;
81    Settings.Defaults;
82    SetSettings(Settings);
83
84    //P.Settings.DoseChange[1].Year := 2100;
85
86    var R := P.Run(Settings);
87
88    Settings.Free;
89    Settings := TPFASPBKRun.create;
90    Settings.Defaults;
91    SetSettings(Settings,2);
92    var R2 := P.Run(Settings);
93
94    Chart1.Series[0].Clear;
95    Chart1.Series[1].Clear;
96    var i:=0;
97    for var O in R do begin
98      Chart1.Series[0].AddXY(O.Year,O.Data.C_m * FCOnv );
99      OT.Writeline(O.Year.ToString+','+O.Data.C_m.ToString+','+O.Data.d_m.ToString);
100   end;
101   for var O in R2 do begin
102     Chart1.Series[1].AddXY(O.Year,O.Data.C_m * FCOnv);
103   end;
104
105   Chart2.Series[0].Clear;
106   for i:=0 to strtoint(edAge.Text)*10 do begin
107     Chart2.Series[0].AddXY(i/10, P.GetBodyWeight(i/10, strtoint(edSex.Text), strtoint(edR.Text)));
108   end;
109
```

```
110   R.Free;
111   if edC.Text='1' then
112     (P as TPFOAPBPK).Free
113   else
114    (P as TPFOSPBPK).Free ;
115   Settings.Free;
116   Cfg.Free;
117   OT.Free;
118 end;
119
120 procedure TForm1.Button2Click(Sender: TObject);
121 begin
122   var T := GetTickCount;
123   var P := TPFOAPBPK.create;
124   var Settings := TPFASPBKRun.create;
125   Settings.Defaults;
126   SetSettings(Settings);
127
128   //simulate what safewater needs...
129   Settings.StartYear := 2023;
130   Settings.EndYear := 2050;
131
132   for var s := 0 to 1 do begin
133     for var d := 1 to 4 do begin
134       for var i := 0 to 85 do begin
135         Settings.BirthYear := Settings.StartYear - i;
136 //P.Settings.EndYear := P.Settings.BirthYear + 50;
137         Settings.Sex := s;
138 //        P.Settings.StartYear := P.Settings.BirthYear;
139 //        P.Settings.EndYear := P.Settings.StartYear + (85 - I mod 85);
140         var R := P.Run(Settings);
141         R.Free;
142       end;
143     end;
144   end;
145
146   var tt := (GetTickCount - t) /1000;
147   Label1.Caption := tt.ToString;
148   P.Free;
149
150   (*
151   T := GetTickCount;
152
153   TParallel.For(1,1,170,procedure(I:Integer)
154     begin
155       var P := TPFOAPBPK.create;
156       P.Settings.Defaults;
157       SetSettings(P.Settings);
158       P.Settings.BirthYear := 1980;
159       P.Settings.StartYear := P.Settings.BirthYear;
160       P.Settings.EndYear := P.Settings.StartYear + (85 - I mod 85);
161       var R := P.Run;
162       R.Free;
163       P.Free;
164     end);
165
166   tt := (GetTickCount - t) /1000;
167   Label3.Caption := tt.ToString;
168   *)
169
170 end;
171
172 procedure TForm1.Button3Click(Sender: TObject);
173 begin
174   var T := TStreamwriter.Create('pbkyear.csv',false,TEncoding.ASCII,4096);
175   var P := TPFOAPBPK.create;
176   var Settings := TPFASPBKRun.create;
177   Settings.Defaults;
178   SetSettings(Settings);
179 //  P.Settings.DoseChange[1].Year := 2100;
180   var R := P.Run(Settings);
181
182   //header
183   var ss:='Birth';
184   for var O in R do begin
```

```
    ss:=ss+','+O.Year.ToString;
  end;
  T.WriteLine(ss);

  ss:=Settings.BirthYear.ToString;
  for var O in R do begin
    ss:=ss+','+O.Data.C_m.ToString;
  end;
  T.WriteLine(ss);
  R.Free;

  //adding weights to the output
  ss:='Wgt';
  for var i:=1 to 86 do begin
     ss:=ss+','+ floattostr(P.GetBodyWeight(i-0.5, strtoint(edSex.Text),0));
  end;
  T.WriteLine(ss);


  Settings.BirthYear := Settings.BirthYear + 1;
  Settings.StartYear := Settings.BirthYear;
  Settings.EndYear := Settings.StartYear + strtoint(edAge.Text);
  R := P.Run(Settings);
  ss:=Settings.BirthYear.ToString+',0';
  for var O in R do begin
    ss:=ss+','+O.Data.C_m.ToString;
  end;
  T.WriteLine(ss);

  R.Free;

  Settings.BirthYear := Settings.BirthYear + 1;
  Settings.StartYear := Settings.BirthYear;
  Settings.EndYear := Settings.StartYear + strtoint(edAge.Text);
  R := P.Run(Settings);
  ss:=Settings.BirthYear.ToString+',0,0';
  for var O in R do begin
    ss:=ss+','+O.Data.C_m.ToString;
  end;
  T.WriteLine(ss);
  R.Free;

  T.Free;
  P.Free;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  var cfg := TSafewaterConfig.create;
  var bw,cons : TCohortData;
  bw.LoadFromFile(cfg.BWFile);
  cons.LoadFromFile(cfg.ConsumptionFile);
  var PS := TSerumCalc.create(pPFOA,cfg,@bw.V,@cons.V);
  PS.OutConv := 1;
  var BD,AD: TSubPopArray;
  PS.Parallel := True;
//PS.SetCustomBodyWeights(nil);
  PS.RunUnitDoses;

  //P := TSerumCalc.create(pPFOA,cfg,@bw.V,@cons.V);
  //P.Parallel := true;
  //P.RunUnitDoses;
  //P.Free;

  cfg.Free;

  var T := TStringLIst.Create;
  var str: string;
  var x: integer;
  str:='a,c1,v1,c2,v2';
  for x := 0 to 85 do str:=str+','+inttostr(2025+x);
  T.Add(Str);
  var P := TPFOAPBPK.create;
  var S := TPFASPBKRun.create;
```

```
259   var C: TValueChange;
260   var A1,A2,A3: TAgeArray;
261   S.Defaults;
262 P.SetCustomBodyWeights(@bw.V);
263
264   S.TimeStep := 1;
265   S.Sex := strtoint(edSex.Text);
266
267   C.Year := 1960;
268   C.Value := 3;
269   S.Dosechange.Add(C);
270   C.Year := 2000;
271   C.Value := 3;
272   S.Dosechange.Add(C);
273
274   for var a := 0 to 85 do begin
275     for var xx := 1 to 1 do begin
276       var cb:double := 1;
277       var ca:double := 0.5;
278
279       if xx=2 then begin
280         cb := xx*2;
281         ca := xx;
282         cb := cb + random;
283         ca := ca + random;
284       end;
285
286       C.Year := 1960;
287       C.Value := cb;
288       S.DoseChange.Items[0] :=C;
289       C.Year := 2025;
290       C.Value := cb;
291       S.DoseChange.Items[1] :=C;
292
293       S.BirthYear := 2025 - a;
294       S.StartYear := 2025 - a;
295       S.EndYear := S.StartYear + 85;
296
297
298       //Before
299       P.RunSafeWaterStyle(A1,S);
300       str := a.ToString+','+S.Dosechange.Items[0].Year.ToString+','+S.Dosechange.Items[0].Value.ToString+','+
301              S.Dosechange.Items[1].Year.ToString+','+S.Dosechange.Items[1].Value.ToString ;
302       for x  := a to high(a1) do str:=Str+','+floattostr(A1[x]);
303       T.Add(Str);
304
305       //After
306       C.Year := 1960;
307       C.Value := ca;
308       S.DoseChange.Items[0] :=C;
309       C.Year := 2025;
310       C.Value := ca;
311       S.DoseChange.Items[1] :=C;
312       P.RunSafeWaterStyle(A2,S);
313       str := a.ToString+','+S.Dosechange.Items[0].Year.ToString+','+S.Dosechange.Items[0].Value.ToString+','+
314              S.Dosechange.Items[1].Year.ToString+','+S.Dosechange.Items[1].Value.ToString ;
315       for x  := a to high(a2) do str:=Str+','+floattostr(A2[x]);
316       T.Add(Str);
317
318       //full run
319       if a=0 then begin
320         C.Year := 2000;
321         C.Value := ca;
322         S.DoseChange.Items[0] :=C;
323       end else begin
324         C.Year := 1960;
325         C.Value := cb;
326         S.DoseChange.Items[0] :=C;
327       end;
328       C.Year := 2025;
329       C.Value := ca;
330       S.DoseChange.Items[1] :=C;
331       P.RunSafeWaterStyle(A3,S);
332       str := a.ToString+','+S.Dosechange.Items[0].Year.ToString+','+S.Dosechange.Items[0].Value.ToString+','+
333              S.Dosechange.Items[1].Year.ToString+','+S.Dosechange.Items[1].Value.ToString ;
```

```
334       for x  := a to high(a3) do str:=Str+','+floattostr(A3[x]);
335       T.Add(Str);
336
337       str := a.ToString+','+S.Dosechange.Items[0].Year.ToString+',p,'+
338              S.Dosechange.Items[1].Year.ToString+',p';
339       var pp:double :=0;
340       for x := a to high(a3)-1 do begin
341         if (A1[x]>0) and (A2[x]>0) then
342           pp :=  1 - (A1[x]-A3[x])/(A1[x]-A2[x])
343         else
344           pp:=0;
345         str:=Str+','+floattostr(pp);
346       end;
347       T.Add(Str);
348
349       str := a.ToString+','+S.Dosechange.Items[0].Year.ToString+',c,'+
350              S.Dosechange.Items[1].Year.ToString+',c';
351       var str2 := a.ToString+','+S.Dosechange.Items[0].Year.ToString+',cp,'+
352              S.Dosechange.Items[1].Year.ToString+',cp';
353
354       var aa:=a;
355       for var oy :=2025 to 2110 do begin
356         PS.GetBeforeAfterSerumFromDose(BD,AD,oy-2025,cb,ca);
357         str := str+','+AD[S.Sex, S.Race,a].ToString;
358         if aa>85 then break;
359         if A3[aa]>0 then
360           str2 := str2+','+floattostr(AD[S.Sex, S.Race,a]/A3[aa])
361         else
362           str2 := str2+',-';
363         inc(aa);
364       end;
365       T.Add(Str);
366       T.Add(Str2);
367
368     end;
369   end;
370
371   T.SaveToFile('curves2.csv');
372   P.Free;
373   S.Free;
374   T.Free;
375   PS.Free;
376 end;
377
378 procedure TForm1.Button5Click(Sender: TObject);
379 begin
380   var cfg := TSafewaterConfig.create;
381   var bw,cons : TCohortData;
382   bw.LoadFromFile(cfg.BWFile);
383   cons.LoadFromFile(cfg.ConsumptionFile);
384   var P := TSerumCalc.create(pPFOA,cfg,@bw.V,@cons.V);
385   P.Parallel := false;
386   P.RunUnitDoses;
387   P.Free;
388
389   //P := TSerumCalc.create(pPFOA,cfg,@bw.V,@cons.V);
390   //P.Parallel := true;
391   //P.RunUnitDoses;
392   //P.Free;
393
394   cfg.Free;
395 end;
396
397 procedure TForm1.Button6Click(Sender: TObject);
398 begin
399   var cfg := TSafewaterConfig.create;
400   var bw,cons : TCohortData;
401   bw.LoadFromFile(cfg.BWFile);
402   cons.LoadFromFile(cfg.ConsumptionFile);
403
404   var PS :TSerumCalc;
405   if edC.Text='1' then begin
406     PS := TSerumCalc.create(pPFOA,cfg,@bw.V,@cons.V);
407   end else begin
```

```
    PS := TSerumCalc.create(pPFOS,cfg,@bw.V,@cons.V);
  end;

  var BD,AD: TSubPopArray;
  PS.Parallel := True;
  PS.SetCustomBodyWeights(nil);
  PS.RunUnitDoses;
  cfg.Free;

  var T := TStringLIst.Create;
  T.Add('age,sex,v');
  var aa:integer;

  var bdd, add: double;
  bdd:=1;
  add:=0.5;
  PS.GetBeforeAfterSerumFromDoseInd(BD,AD,10,bdd,add);
  for var s := gFemale downto gMale do begin
    for var a := 0 to CMaxAge do begin
      T.Add(a.ToString+','+s.ToString+','+Floattostr(AD[s,strtoint(edr.Text),a]/1000));
    end;
  end;

  if edC.Text='1' then begin
    T.SaveToFile('pfoaChange10.csv');
  end else begin
    T.SaveToFile('pfosChange10.csv');
  end;
  T.Free;
  PS.Free;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  var bc := 3.2 / 1000000;
  var d := 0.013;
  //bc * D * FOutConv;

end;

procedure TForm1.SetSettings(var T: TPFASPBKRun; adiv : integer = 1);
begin
  T.TimeStep := strtofloat(edTS.Text);
  T.Sex := strtoint(edSex.Text);
  T.Race := strtoint(edR.Text);

  T.BirthYear := 2023;
  T.StartYear := T.BirthYear;
  T.EndYear := T.StartYear + strtoint(edAge.Text);
  T.DoseChange.Clear;
  var C: TValueChange;

  if aDiv =1 then  begin
    for var s in memDC.Lines do begin
      var D := SplitString(s,',');
      C.Year := strtofloat(D[0]);
      C.Value := strtofloat(D[1]);
      T.Dosechange.Add(C);
    end;
  end else begin
    for var s in memDC2.Lines do begin
      var D := SplitString(s,',');
      C.Year := strtofloat(D[0]);
      C.Value := strtofloat(D[1]);
      T.Dosechange.Add(C);
    end;
  end;

end;

end.

```

code\SafeWaterCalcDist.pas

```
unit SafeWaterCalcDist;

interface

uses Math,CalcDistSpecFun,SysUtils;

const

   dNone       = 299;
   dCustom     = 300;
   dNormal     = 301;   dTriangular  = 302;   dPoisson  = 303;
   dBinomial   = 304;   dLogNormal   = 305;   dUniform  = 306;
   dExponential= 307;   dGeometric   = 308;   dWeibull  = 309;
   dGamma      = 310;   dLogistic    = 311;   dCauchy   = 314;
   dPareto     = 313;   dBeta        = 312;   dHML      = 315;
   dVariableCustom  = 316;
   dLogUniform =317;

   tiny        = 1.0e-11;
   error_value = -99.9;

{functions from Numerical Recipes, used to calc Beta, Binomial, and Poisson}

function  gammp(a,x: double): double;
function  gammq(a,x:double):double;
procedure gser(a,x:double; var gamser,gln: double);
procedure gcf(a,x: double; var gammcf,gln: double);
function  betacf(a,b,x: double): double;
function  betai(a,b,x:double):double;

{These three functions provide a generalized method of calling icdf,
 cdf, and random functions when the distribution type is variable}
function icdf(dtype:integer;y,parm1,parm2,parm3:double):double;
function cdf(dtype:integer;x,parm1,parm2,parm3:double):double;

{The following suite of functions return a randomnly selected
 value from the respective distrubtion}
function rNormal(Mean, StandardDeviation : double) : double;
function rLogNormal(GM, GSD : double) : double;
function rPoisson(Rate : double) : double;
function rExponential(Rate : double) : double;
function rTriangular(Minimum, Maximum, MostLikely : double) : double;
function rBinomial(Probability, Trials : double) : double;
function rUniform(Minimum, Maximum : double) : double;
function rGeometric(Probability : double) : double;
function rWeibull(scale,shape:double):double;
function rLogistic(Mean, Scale : double) : double;
function rGamma(a,Lamda : double) : double;
function rBeta(aa,bb, Scale : double) : double;
function rCauchy(a,b:double): double;
function rPareto(x0,Theta : double) : double;
function rHML(High,Medium,Low : double) : double;
function rLogUniform(Minimum, Maximum : double) : double;

{The next set of functions return CVM Stat of the fit for each
 distribution type and parameter estimates. }
//function cdfCustom(X : double; Stats : TDescriptiveStatistics) : single;
//function icdfCustom(Prob : double; Stats : TDescriptiveStatistics) : double;

function cdfNormal(y,mean,dev:double) : single;
function icdfNormal(Prob,mean,dev:double) : double;

function cdfLogNormal(x,GM,GSD: double): single;
function icdfLogNormal (Prob,GM,GSD:double) : double;

function  icdfTriangular(y,A,B,ML:double): single;
function  cdfTriangular(X,A,B,ML : double) : single;

function cdfUniform(X,A,B : double) : single;
function icdfUniform(Prob,A,B : double) : double;

function cdfExponential(x,lamda : double): single;
function icdfExponential(Prob,lamda : double): double;
```

```
74
75 function cdfGamma(X,A : double) : single;
76 function icdfgamma(P,A : double) : double;
77
78 function cdfWeibull(XD,scale,shape : double) : single;
79 function icdfweibull(y,scale,shape:double): double;
80
81 function cdfLogistic(x,a,b:double): single;
82 function icdfLogistic(Prob,a,b:double): double;
83
84 function cdfPoisson(xd,lambda:double) : single;
85 function icdfPoisson(y,lambda:double) : double;
86
87 function cdfBinomial(xd,p : double; n : integer): single;
88 function icdfBinomial(y,p : double; n: integer): double;
89
90 function cdfGeometric(xd,p : double): single;
91 function icdfGeometric(y,p : double): double;
92
93 function cdfBeta(x,a,b,scale:double): single;
94 function icdfBeta(y,a,b,scale:double):double;
95
96 function cdfCauchy(x,a,b:double):single;
97 function icdfCauchy(prob,a,b:double):double;
98
99 function cdfPareto(x,x0,theta:double):single;
100 function icdfPareto(y,x0,theta:double):double;
101
102 function cdfHML(x,High, Medium, Low:double):single;
103 function icdfHML(prob,High, Medium, Low:double):double;
104
105 function cdfLogUniform(x,A,B :double):single;
106 function icdfLogUniform(prob,A,B:double):double;
107
108
109 implementation
110
111 {-----------------------------------------------------------------------}
112
113 { NUMERICAL RECIPES FUNCTIONS }
114
115 procedure gser(a,x:double; var gamser,gln: double);
116    const itmax=1000; eps=3.0e-7;
117    var
118       n:integer; sum,del,ap: double;
119 begin
120   if x<0 then raise exception.Create('x<0 in CalcDist.gser');
121   gln:=lngamma(a);
122
123   if x=0 then begin
124     gamser := 0
125   end else begin
126     ap := a; sum:=1/a; del:=sum;
127     for n:=1 to itmax do begin
128       ap:= ap+1; del:=del*x/ap; sum:=sum+del;
129       if (abs(del) < abs(sum)*eps) then break;
130       if n=itmax then raise exception.Create('error in CalcDist.gser ');
131     end; {for do loop}
132     Gamser:=sum*exp(-x+a*ln(x)-gln)
133   end {else}
134 end;  {gser}
135
136 procedure gcf(a,x: double; var gammcf,gln: double);
137    const itmax=1000; eps=3.0e-7;
138    var n:integer; gold,g,fac,b1,b0,anf,ana,an,a1,a0:double;
139 begin
140   gln:=lngamma(a);
141   gold:=0.0; a0:=1.0; a1:=x; b0:=0.0; b1:=1.0; fac:=1.0;
142   for n:=1 to itmax do begin
143     an:=1.0*n; ana:=an-a; a0:=(a1+a0*ana)*fac;
144     b0:=(b1+b0*ana)*fac; anf := an*fac; a1 := x*a0+anf*a1;
145     b1:=x*b0+anf*b1;
146     if (a1<>0) then begin
147       fac:= 1.0/a1; g:=b1*fac;
```

```
      if (abs((g-gold)/g) < eps) then break;
      gold :=g
    end;
    if n=itmax then raise exception.Create('error in CalcDist.gcf');
  end;
  gammcf := exp(-x+a*ln(x)-gln)*g
end;  {gcf}

function gammp(a,x: double): double;
var
  gammcf,gln: double;
begin
   if ((x < 0.0) OR (a <= 0.0)) then begin
     raise exception.Create('calcdist.GAMMP - invalid arguments');
   end;
   if (x < (a+1.0)) then begin
     gser(a,x,gammcf,gln);
     gammp := gammcf
   end else begin
     gcf(a,x,gammcf,gln);
     gammp := 1.0-gammcf
   end
end;

function gammq(a,x:double):double;
var gamser,gln:double;
begin
  if (x<0) or (a<=0) then raise exception.Create('error in CalcDist.gammq');
  if (x<a+1) then begin
    gser(a,x,gamser,gln); gammq:=1-gamser end
  else begin
    gcf(a,x,gamser,gln);
    gammq:=gamser
  end;
end;   { gammq }


function betacf(a,b,x: double): double;
  const
     itmax=1000; eps=3.0e-7;
  var
     tem,qap,qam,qab,em,d,bz,bpp,bp,bm,az,app,am,aold,ap: double;
     m: integer;
begin
  am:=1; bm:=1; az:=1; qab:=a+b; qap:=a+1;
  qam := a-1; bz:=1-qab*x/qap;

  for m:= 1 to itmax do begin
    em:=m; tem:=em+em; d:= em*(b-m)*x/((qam+tem)*(a+tem));
    ap:=az+d*am; bp:=bz+d*bm;
    d:=-(a+em)*(qab+em)*x/((a+tem)*(qap+tem)); app:=ap+d*az;
    bpp:=bp+d*bz; aold:=az; am:=ap/bpp; bm:=bp/bpp;
    az := app/bpp; bz:=1;
    if ((abs(az-aold))<(eps*abs(az))) then break;
    if m=itmax then raise exception.Create('error in CaclDist.betacf');
  end;
  Betacf:=az;
end; {betacf}


function betai(a,b,x:double):double;
var Bt: double;
begin
  if (a<=0) or (b<=0) then
    exception.Create('error in CaclDist.betai');
  if (x<0) or (x>1) then
    exception.Create('error in CaclDist.betai');
  if (x=0) or (x=1) then
    bt :=0
  else
    bt := exp(lngamma(a+b)-lngamma(a)-lngamma(b) + a * ln(x) + b*ln(1-x));
  if ( x < ((a+1)/(a+b+2)) ) then
    betai := bt*betacf(a,b,x)/a
  else
    betai:=1-bt*betacf(b,a,1-x)/b
```

```
end;
{end of Numerical Recipes Code}
{-----------------------------------------------------------------------}

{GENERALIZED FUNCTIONS}

function icdf(dtype:integer;y,parm1,parm2,parm3:double):double;

begin
   Case Dtype of
       dGamma       : icdf:=parm1+parm3*icdfGamma(y,Parm2);
       dBeta        : icdf:=icdfbeta(y,parm1,Parm2,Parm3);
       dNormal      : icdf:=icdfNormal(y,parm1,Parm2);
       dLogNormal   : icdf:=icdfLogNormal(y,parm1,Parm2);
       dTriangular  : icdf:=icdfTriangular(y,parm1,Parm2,Parm3);
       dPoisson     : icdf:=icdfPoisson(y,parm1);
       dBinomial    : icdf:=icdfBinomial(y,parm1,round(Parm2));
       dUniform     : icdf:=icdfUniform(y,parm1,Parm2);
       dExponential : icdf:=icdfExponential(y,parm1);
       dGeometric   : icdf:=icdfGeometric(y,parm1);
       dWeibull     : icdf:=icdfWeibull(y,parm1,Parm2);
       dLogistic    : icdf:=icdfLogistic(y,parm1,Parm2);
       dCauchy      : icdf:=icdfCauchy(y,parm1,Parm2);
       dPareto      : icdf:=icdfPareto(y,parm1,Parm2);
       dHML         : icdf:=icdfHML(y,parm1,Parm2, Parm3);
       dLogUniform  : icdf:=icdfLoguniform(y,parm1,Parm2);
       dnone        : icdf:=1;
       end; {case}
end; {ICDF}

function cdf(dtype:integer;x,parm1,parm2,parm3:double):double;
begin
   Case Dtype of
       dGamma       : cdf:=cdfGamma(x/parm3-parm1,Parm2);
       dBeta        : cdf:=cdfbeta(x,parm1,Parm2,Parm3);
       dNormal      : cdf:=cdfNormal(x,parm1,Parm2);
       dLogNormal   : cdf:=cdfLogNormal(x,parm1,Parm2);
       dTriangular  : cdf:=cdfTriangular(x,parm1,Parm2,Parm3);
       dPoisson     : cdf:=cdfPoisson(x,parm1);
       dBinomial    : cdf:=cdfBinomial(x,parm1,round(Parm2));
       dUniform     : cdf:=cdfUniform(x,parm1,Parm2);
       dExponential : cdf:=cdfExponential(x,parm1);
       dGeometric   : cdf:=cdfGeometric(x,parm1);
       dWeibull     : cdf:=cdfWeibull(x,parm1,Parm2);
       dLogistic    : cdf:=cdfLogistic(x,parm1,Parm2);
       dCauchy      : cdf:=cdfCauchy(x,parm1,Parm2);
       dPareto      : cdf:=cdfPareto(x,parm1,Parm2);
       dHML         : cdf:=cdfHML(x,Parm1,Parm2,Parm3);
       dLogUniform  : cdf:=cdfLoguniform(x,parm1,Parm2);
       dnone        : cdf:=x;
       end; {case}
end; {cdf}

{-----------------------------------------------------------------------}

FUNCTION rbeta(aa,bb,scale:double): Double;

  {  Returns a single random deviate from the beta distribution with
     parameters A and B.  The density of the beta is
     x^(a-1) * (1-x)^(b-1) / B(a,b) for 0 < x < 1}

  { R. C. H. Cheng
    Generating Beta variatew with Nonintegral Shape Parameters
    Communications of the ACM, 21:317-322  (1978)
    Translated from fortran }

  { Preliminary Eyeballing of distributions of the output from this
    function seem to indicate that it's working fine.}

Const  expmax=89;
       infnty=1.0e38;

var  a,alpha,b,beta,gamma,r,s,t,u1,u2,v,w,z: double;

```

```
begin
  if ((aa=1) and (bb=1)) then begin
    rbeta:=random*scale;
    exit;
  end;
  if aa<bb then begin
    a:=aa;
    b:=bb;
  end
  else begin
    b:=aa;
    a:=bb;
  end;
  alpha := a + b;
  beta := sqrt((alpha-2.0) / (2.0*a*b-alpha));
  gamma := a + 1.0/beta;

  repeat
    u1 := random;
    u2 := random;
    v := beta*ln(u1/ (1.0-u1));
    if v>expmax then w:=infnty else w:=a*exp(v);
    z := (u1*u1)*u2;
    r := gamma*v - 1.3862944;
    s := a + r - w;
    t := ln(z)
  until (((r+alpha*ln(alpha/ (b+w)))>=t) or ((s+2.609438) >= (5.0*z)) or (s>t));

  if  aa=a then
    rbeta := scale*(w/(b+w))
  else
    rbeta := scale*(b/(b+w));
end;


function rNormal(Mean, StandardDeviation : double) : double;
var r, v1, v2, v3, dev: double;
begin
  repeat
    v1 := 2*Random-1;
    v2 := 2*Random-1;
    r := sqr(v1) + sqr(v2);
  until (r<1.0);
  if Random(2) = 1 then
    v3:=v2
  else
    v3:=v1;
  dev := v3 * SQRT(( - 2 * LN(r)) / r);
  rNormal := mean + dev * StandardDeviation;
end;


function rLogNormal(GM, GSD : double) : double;
begin
  rLogNormal:=exp(rNormal(ln(GM),ln(GSD)));
end;

(* Poisson routine taken from numerical recipes *)
(* gammln used in Poisson *)
function gammln(xx:double) : double;
const
   stp = 2.50662827465;
   half= 0.5;
   fpf = 5.5;
   cof: array[1..6] of double = (76.18009173,-86.50532033,24.01409822,
                                 -1.231739516,0.120858003e-2, -0.536382e-5);
var
   x,tmp,ser:double;
   j:integer;
begin
   x:= xx-1;
   tmp := x+fpf;
   tmp := (x+0.5)*ln(tmp)-tmp;
   ser := 1.0;
   for j := 1 to 6 do begin
```

```
      x:= x+1;
      ser := ser+cof[j]/x;
    end;
    gammln := tmp+ln(stp*ser);
end;


function rPoisson(Rate : double) : double;
var n,t:double;    (* times, ln of waiting time (1st part) *)
    limit,glsq,gllnrate,glg,y:double;

begin
   if (Rate<12) then begin
     n := -1;
     t := 1.0;
     limit := exp(-Rate);
     repeat
       n := n+1;
         t := t * random;
     until (t< limit);
   end else begin
     glsq := sqrt(2.0*Rate);
     gllnRate := ln(Rate);
     glg := Rate*gllnRate-gammln(Rate+1.0);
     repeat
       repeat
           y:= PI * Random;
         y:= sin(y)/cos(y);
         n:=glsq*y+Rate;
       until (n >= 0.0);
       n := trunc(n);
       t := 0.9*(1.0+sqr(y))*exp(n*gllnRate-gammln(n+1.0)-glg);
     until (t >= Random);
   end;
   rPoisson:=n;
end;

function rExponential(Rate : double) : double;
begin
  rExponential:= - ln(Random)/Rate;
end;


function rTriangular(Minimum, Maximum, MostLikely : double) : double;
var temp, base, part: double;
begin
  base := Maximum - Minimum;
  part := MostLikely - Minimum;
  Temp:=Random;
  if (Temp < (part/base)) then
     Temp := Minimum + Sqrt(base*part*Temp)
  else
     Temp := Maximum - Sqrt(base*(Maximum-Mostlikely)*(1-Temp));
  rTriangular := Temp;
end;

function rBinomial(Probability, Trials : double) : double;
{Probability must be in (0,1) for this function to work}

var mean,em,en,g,angle,oldg,p,pc,bnl: double;
    pclog,plog,sq,t,y:double;
    j: integer;
    n     : longint;
begin
   if (Probability<=0) or (Probability>=1) then
     raise exception.Create('error in CaclDist.rBinomial');
   n:=Trunc(Trials);
   if (Probability<=0.5) then p:=Probability else p:=1.0-Probability;
   mean := n*p;
   if (n<25) then begin
      bnl := 0.0;
      for j:=1 to n do begin
        if (Random<p) then bnl := bnl+1.0;
      end
```

```
446     end else
447     if (mean<1.0) then begin
448       g := exp(-mean);
449       t := 1.0;
450       j := 0;
451       repeat
452          t:=t*Random;
453          j:=j+1;
454       until ((t<g) OR (j=n));
455       bnl := j;
456     end else begin
457       en := n; oldg := gammln (en+1.0);
458       pc := 1.0-p; plog := ln(p); pclog := ln(pc);
459       sq := sqrt(2.0*mean*pc);
460       repeat
461         repeat
462             angle := pi * Random;
463           y := tan(angle);
464           em := sq*y+mean;
465         until ((em >=0.0) and (em<en+1.0));
466         em := trunc(em);
467         t := 1.2*sq*(1.0+sqr(y))   * exp(oldg-gammln(em+1.0) -
468              gammln(en-em+1.0)+ em * plog+(en-em)*pclog);
469       until (Random<=t);
470       bnl := em;
471     end;
472     if(p<>Probability) then bnl := n-bnl;
473     rBinomial := bnl;
474   end;
475
476
477   function rUniform(Minimum, Maximum : double) : double;
478   begin
479     rUniform:=Minimum + (Maximum-Minimum)*Random;
480   end;
481
482   function rLogUniform(Minimum, Maximum : double) : double;
483   begin
484     Result := icdfLogUniform(Random,Minimum,Maximum)
485   end;
486
487
488   function rGeometric(Probability : double) : double;
489   {Probability must be in (0,1) for this to work.}
490   begin
491     if (probability<=0) or (probability >=1) then
492       raise exception.Create('error in CaclDist.rGeometric');
493     rGeometric:=trunc(ln(1-Random)/ln(1-Probability))+1;
494   end;
495
496
497   function rWeibull(scale,shape: double) : double;
498   {Inversion of CDF given in @Risk book
499    Changed functional form 7/1/96
500    removed scale=(1/scale)^(1/shape) - BMF}
501   begin
502     rWeibull := scale * power( (ln(1/random)) , (1/shape) );
503   end;
504
505
506   (* simple inversion of cumulative distribution function *)
507   function rLogistic(Mean, Scale : double) : double;
508   begin
509     rLogistic:=Mean-Scale*ln(1/Random - 1);
510   end;
511
512   (* compared to Erlang, this gets n-th time something happens  *)
513   (* distribution lambda^n * x^(n-1) * exp(-lambda*x)/Gamma(n)  *)
514   (* Gamma is just controlling routine, calling either          *)
515   (* gamma_calc1a (when a>=1) or gamma_calc2 (when 0<a<1).      *)
516   (* gamma_calc1 is an older routine that I found first;        *)
517   (* gamma_calc1a is apparently much faster.                    *)
518   (* gamma_calc1, gamma_calc1a are for parameters a>=1          *)
519   (* gamma_calc2 is for parameters 0<a<1                        *)
520   (* controlling routine must call right one                    *)
```

```
(* this function can be called only when alpha is >1
   there is no error checking, but the results may not be good
   Based on an algorithm presented in :
   Minh, Do Le.  ACM Trans. on Math. Software,14/3
   September 1988                                          *)

function gamma_calc1a(alpha:double):double;
label top,deliver;
var a,D,Dl,x,x1,x2,x4,x5,x_el,xp,xr,u,v,w:double;
    f1,f2,f4,f5,p1,p2,p3,p4:double;
    result1:double;
begin
   a := alpha -1.0;
   D := sqrt(a);
   if (alpha <= 2) then begin
      Dl := a/2;
      x1 := 0;
      x2 := Dl;
      x_el := -1.0;
      f1 := 0;
   end else begin
      Dl := D - 0.5;
      x2 := a- Dl;
      x1 := x2 - Dl;
      x_el := 1.0 - a/x1;
      f1 := exp(a*ln(x1/a) + 2*Dl);
   end;
   f2 := exp(a*ln(x2/a)+Dl);
   x4 := a+D;
   x5 := x4 + D;
   xr := 1.0 - a/x5;
   f4 := exp(a*ln(x4/a)-D);
   f5 := exp(a*ln(x5/a) - D - D);
   p1 := 2* f4 * D;
   p2 := 2 * f2 * Dl + p1;
   p3 := f5/xr + p2;
   p4 := -f1/x_el + p3;
top:                                          { 5 in Minh }
   repeat
      u := p4 * Random;
      if (u>p1) then begin
     if (u>p2) then begin
        w := Random;
        if (u>=p3) then begin               { 10 in Minh }
           u := (p4-u)/(p4-p3);
           x := x1 - ln(u)/x_el;
           if (x<0) then goto top
               else
               if (w<=(x_el*(x1-x)+1)/u) then begin
              result1 := x;
                  goto deliver;
           end else w := w*f1*u;
        end else begin                                 {9 in Minh }
           u := (p3-u)/(p3-p2);
           x := x5 - ln(u)/xr;
           if (w<=(xr*(x5-x)+1)/u) then begin
              result1 := x;
          goto deliver;
           end;
           w := w*f5*u;
        end;
     end else begin                                     { 7 in Minh }
        w := (u-p1)/Dl - f2;
        if (w<=0) then begin
           result1 := a-(u-p1)/f2;
           goto deliver;
        end;
        if (x<=f1) then begin
           result1 := x1+(w*Dl)/f1;
           goto deliver;
        end;
        v := Random;                                { 8 in Minh }
        x := x1 + v*Dl;
        xp := 2*x2-x;
```

```
595         if (w>=f2+(f2-1)*(x-x2)/(x2-a)) then begin
596            result1 := xp;
597            goto deliver;
598         end;
599         if (w<=f2*(x-x1)/Dl) then begin
600            result1 := x;
601            goto deliver;
602         end;
603         if ((w>2*f2-1) AND (w>2*f2 - exp(a*ln(xp/a)+a-xp))) then begin
604            result1 := xp;
605            goto deliver;
606         end;
607      end;
608       end else begin                                        { 5,6 in Minh }
609          w :=  u/D - f4;
610      if (w<=0) then begin
611         result1 := a + u/f4;
612         goto deliver;
613      end;
614      if (w<=f5) then begin
615         result1 := x4 + (w*D)/f5;
616         goto deliver;
617      end;
618      v := Random;
619      x := x4 + v*D;
620      xp := 2*x4-x;
621      if (w>=f4+(f4-1)*(x-x4)/(x4-a)) then begin
622         result1 := xp;
623         goto deliver;
624      end;
625      if (w<=f4+(a/x4-1)*f4*(x-x4)) then begin
626         result1 := x;
627         goto deliver;
628      end;
629      if ((w>=2*f4-1) AND (w>=2*f4-exp(a*ln(xp/a)+a-xp))) then begin
630         result1 := xp;
631         goto deliver;
632      end;
633       end;
634    until(ln(w)<=a*ln(x/a)+a-x);                             (* 11 in Minh *)
635    result1 := x;
636 deliver:  gamma_calc1a := result1;
637 end;
638
639
640 { the routine for when 0<a<1 }
641 function gamma_calc2(a:double) : double;
642 const e_hat_minus_one = 0.3678794;
643 var b,p,check,sgamma:double;
644 begin
645   b := 1.0 + a * e_hat_minus_one;
646   repeat
647     p := b * Random;
648     if (p<1.0) then begin
649       sgamma := exp(ln(p)/a);
650       check :=  sgamma;
651     end else begin
652       sgamma := - ln((b-p)/a);
653       check := (1.0-a) * ln(sgamma);
654     end;
655   until (check <= rExponential(a));
656   gamma_calc2 := sgamma;
657 end;
658
659
660 function rGamma(a,Lamda : double) : double;
661 begin
662   if (a>=1) then rGamma := Lamda*gamma_calc1a(a)
663   else rGamma := Lamda*gamma_calc2(a);
664 end;
665
666
667 function rCauchy(a,b:double): double;
668
669 {This function usually requires at least one parameter.  In the
```

```
function we received from NRAND1, the alpha (centering) parameter
is assumed as 0, beta as 1.

Therefore, the output is extrapolated to the correct parametrization by
running cdf of the output of this function with parameters of 0 and 1 and
then running icdf of that output with the parameter that is desired.  }

var t,u:double;
begin
  u:=random;
  if (u=0) then begin
     repeat; u:=random; until (u>0);
  end;  {range of borland random is 0<=random<1 }
  t:=(u-0.5)*pi;
  rCauchy:=cdfcauchy(cdfCauchy((sin(t)/cos(t)),0,1),a,b);
end;

function rPareto(x0,theta: double):double;
{function created by inserting a random number (0-1) into the
inverse of the cumulative density function. Feb 13, 1995 }
{theta's too low cause FP overflow}

begin
  if (x0<=0) or (theta<0.02) then
    raise exception.Create('error in CaclDist.rPareto');
  rpareto:= x0+x0*(power((1-random),(-1/theta))-1);
end;

function rHML(High, Medium, Low : double):double;
begin
  case Random(2) of
    0 : rHML:=High;
    1 : rHML:=Medium;
    2 : rHML:=Low;
  end;
end;

(*
function cdfCustom(X : double; Stats : TDescriptiveStatistics) : single;
var t : integer;
    prob : double;
begin
  t:=-1;
  if X=Stats.Quantiles[0] then
    prob:=0
  else
  if X=Stats.Quantiles[100] then
    prob:=1
  else begin
    repeat
      t:=t+1;
    until (X>=Stats.Quantiles[t]) and (X<=Stats.Quantiles[t+1]) and (t<100);
    if  (Stats.Quantiles[t+1]=Stats.Quantiles[t]) then
      prob:=t*0.05+0.025
    else
      prob:=t*0.05+0.05*((X-Stats.Quantiles[t])/(Stats.Quantiles[t+1]-Stats.Quantiles[t]));
  end;
  cdfCustom:=prob;
end;


function icdfCustom(Prob : double; Stats : TDescriptiveStatistics) : double;
var t : longint;
    x : single;
begin
  t:=trunc(Prob*100) DIV 5;
  if t<100 then begin
    if (Stats.Quantiles[t+1]-Stats.Quantiles[t])>0 then
      x:=Stats.Quantiles[t]+((Prob-t*0.05)/(0.05))*(Stats.Quantiles[t+1]-Stats.Quantiles[t])
    else
      x:=Stats.Quantiles[t];
  end else
    x:=Stats.Quantiles[100];
  icdfCustom:=x;
```

```
end;
*)


function cdfNormal(y,mean,dev:double) : single;
const
        p  : extended =  0.231641900;
        b1 : extended =  0.319381530;
        b2 : extended = -0.356563782;
        b3 : extended =  1.781477937;
        b4 : extended = -1.821255978;
        b5 : extended =  1.330274429;
var
   sd,x,y1,y2,y3,y4,y5    : extended;
   t,z,r                  : extended;
begin
  sd := (y-mean)/dev;
{ check to see if the standard deviation ( sd ) is in range }
  {if sd < 0 then exit;}
  x := sd;
  r := EXP(-(x*x)/2)/2.5066282746;
{  r = frequency }
  z  := x;
  y1 := 1/(1+(p*ABS(x)));
  y2 := y1 * y1;
  y3 := y2 * y1;
  y4 := y3 * y1;
  y5 := y4 * y1;
  t  := 1 - r*(b1*y1+ b2*y2 + b3*y3 + b4*y4 + b5*y5);
  if z > 0 then
    cdfNormal := t
  else
    cdfNormal := 1 - t;
end;

function icdfNormal(prob,mean,dev:double) : double;
const
        c0 : extended = 2.515517;
        c1 : extended = 0.802853;
        c2 : extended = 0.010328;
        d1 : extended = 1.432788;
        d2 : extended = 0.189269;
        d3 : extended = 0.001308;
var
        dn,up,xp : EXTendED;
        t1,t2,t3 : EXTendED;
begin
  if prob=1 then icdfnormal:=mean+5*dev;
  if prob=0 then icdfnormal:=mean-5*dev;
  if (prob <= 0) OR (prob >= 1) then exit;
  if prob > 0.5 then
     xp := 1 - prob
  else
     xp := prob;
  t1 := SQRT(LN(1/SQR(xp)));
  t2 := t1 * t1;
  t3 := t2 * t1;
  up := c0  + c1*t1 + c2*t2;
  dn := 1 + d1*t1 + d2*t2 + d3*t3;
  xp := t1 - (up/dn);
  if prob <= 0.5 then
    icdfNormal := mean-xp*dev
  else
    icdfNormal := mean+xp*dev;
end;

function cdfLogNormal(x,GM,GSD: double): single;
var u,std: double;
begin
  if x=0 then cdflognormal:=1;
  if (GM<=1) or (GSD<=1) or (x<=0) then exit;
  u:=ln(GM); std:=ln(GSD);
  cdflognormal:=cdfnormal((ln(x)-u)/std,0,1);
end;

```

```
819 function icdfLogNormal(Prob,GM,GSD:double) : double;
820 var n,u,std: double;
821 begin
822   if (GM<=1) or (GSD<=1) then exit;
823   if Prob=0 then icdflognormal:=0;
824   if Prob=1 then icdflognormal:=1e200;
825   if (prob<=0) or (prob>=1) then exit;
826   u:=ln(GM); std:=ln(GSD);
827   n:=icdfnormal(prob,0,1);
828   if u+std*n>1e4 then icdflognormal:=1e200 else
829                       icdflognormal:=exp(u+std*n);
830 end;
831
832 function  icdfTriangular(y,A,B,ML:double): single;
833 var maba : double;
834 begin
835   icdftriangular:=0;
836   if b=a then exit;
837   maba:=(ML-A)/(B-A);
838   if (y<=maba) then
839     icdftriangular:=a + sqrt(y * (ML-a) * (B-A))
840   else
841     icdftriangular:=b - sqrt((1-y) * (B-ML) * (B-A));
842 end;
843
844 function cdfTriangular(X,A,B,ML : double) : single;
845 begin
846   cdftriangular:=0;
847   if b=a then exit;
848   if x<A then exit;
849   if x>B then begin cdftriangular:=1; exit; end;
850   if x<ML then cdfTriangular:=((ML-A)/(B-A)) * sqr((x-a)/(ml-a))
851           else cdfTriangular:=1-((B-ML)/(B-A)) * sqr((b-x)/(b-ml));
852 end;
853
854
855 function cdfUniform(X,A,B : double) : single;
856 begin
857    cdfUniform := (x-a)/(b-a);
858 end;
859 function icdfUniform(Prob,A,B : double) : double;
860 begin
861    icdfUniform:=Prob*(b-a)+a;
862 end;
863
864 function cdfLogUniform(x,A,B :double):single;
865 begin
866   if x<a then Result := 0 else
867   if x>=b then Result := 1 else
868   Result := ln(x/a)/ln(b/a);
869 end;
870 function icdfLogUniform(prob,A,B:double):double;
871 begin
872   Result := exp(ln(b/a)*prob)*a;
873 end;
874
875 function cdfExponential(x,lamda : double): single;
876 begin
877    cdfExponential:=1.0-1/exp(lamda*x);
878 end;
879 function icdfExponential(Prob,lamda : double): double;
880 begin
881    icdfExponential:=ln(1/(1-Prob))/Lamda;
882 end;
883
884 function cdfGamma(X,A : double) : single;
885 begin
886   cdfGamma:=igammap(A,X);
887 end;
888
889 function icdfgamma(p,a: double):double;
890 {RETURNS NEGATIVE ONE (-1.0) if FUNCTION NON-EVALUABLE}
891
892 {From fortran code written by A.H. Morris Jr}
```

```
Label 30,1,2,3,110,120,130,140,150,170,180,190,200,210,4,230,240,250,260;

CONST NUM_ITERATIONS=50;

var  x,q,x0,a0,a1,a2,a3,am1,amax,ap1,ap2,ap3,apn,b,b1,b2,b3,
     b4,c,c1,c2,c3,c4,c5,d,e,e2,eps,g,h,ln10,pn,qg,qn,
     r,rta,s,s2,sum,t,tol,u,w,xmax,xmin,xn,y,z: double;
     ierr,iop: integer;
     amin,bmin,dmin,emin,eps0:  array [0..2] of double;

begin
  ln10:=2.302585;
  c:=0.577215664901533 ;
  a0:=3.31125922108741 ;a1:=11.6616720288968 ;
  a2:=4.28342155967104 ;a3:=0.213623493715853 ;
  b1:=6.61053765625462 ;b2:=6.40691597760039 ;
  b3:=1.27364489782223 ;b4:=0.036117081018842 ;
  eps0[1]:=1E-10; eps0[2]:=1E-08;
  amin[1]:=500.0 ; amin[2]:=100.0 ;
  bmin[1]:=1E-28; bmin[2]:=1E-13;
  dmin[1]:=1E-06; dmin[2]:=1E-04;
  emin[1]:=2E-03; emin[2]:=6E-03;
  tol:=1E-5;
  ierr:=0;
  x0:=0;
{  E, xmin, and xmax are machine dependent constants.
    e is the smallest number for which 1.0 + e > 1.0.
    xmin is the smallest positive number and xmax is the
    largest positive number.}

  e := 1e-21;
  xmin := 1e-150;
  xmax := 9.9e150;

  x := 0.0;
  if (a<0.001) then begin icdfgamma:=-1; exit; end;
  q:=1-p;
  if (p=0.0) then begin icdfgamma:=x; exit; end;
  if (q=0.0) then begin icdfgamma := xmax; exit; end;
  if (a=1.0) then
    if (q>=0.9) then begin
      icdfgamma :=  -ln(1-p); exit;
    end else begin
      icdfgamma := -ln(q); exit;
    end;

  e2 := 2.0*e;
  amax := 0.4E-10/(sqr(e));
  iop := 1;
  if (e>1E-10)  then iop := 2;
  eps := eps0[iop];
  xn := x0;

{A<1}

  if (a>1.0) then GOTO 2;
  g := calcgamma(a+1.0);
  qg := q*g;
  if (qg=0.0) then begin
    icdfgamma:=xmax;
    exit;
  end;
  b := qg/a;
  if (qg>0.6*a) then GOTO 3;
  if not ((a>=0.30) OR (b<0.35)) then begin
    t := exp(-(b+c));
    u := t*exp(t);
    xn := t*exp(u);
    GOTO 1;
    end;
  if (b>=0.45) then GOTO 3;
  if (b=0.0) then begin
    icdfgamma:=xmax;
    exit;
```

```
  end;
  y := -ln(b);
  s := 0.5 + (0.5-a);
  z := ln(y);
  t := y - s*z;
  if (b>=0.15) then begin
    xn := y - s*ln(t) - ln(1+s/(t+1));
    GOTO 4;
  end;

  if (b>0.01) then begin
    u := ((t+2*(3-a))*t+(2-a)* (3-a))/
          ((t+(5-a))*t+2);
    xn := y - s*ln(t) - ln(u);
    GOTO 4;
  end;

30:
  c1 := -s*z;
  c2 := -s* (1+c1);
  c3 := s* ((0.5*c1+ (2.0-a))*c1+ (2.5-1.5*a));
  c4 := -s*(((c1/3+ (2.5-1.5*a))*c1+ ((a-6)*a+7))*
            c1+ ((11*a-46)*a+47)/6);
  c5 := -s* ((((-c1/4+ (11*a-17)/6)*c1+ ((-3.0*a+
       13.0)*a-13.0))*c1+0.5* (((2.0*a-25.0)*a+72.0)*a-
        61.0))*c1+ (((25.0*a-195.0)*a+477.0)*a-379.0)/
        12.0);
  xn := ((((c5/y+c4)/y+c3)/y+c2)/y+c1)+ y;
  if (a>1.0) or (b>bmin[iop]) then GOTO 4;
  icdfgamma := xn;
  EXIT;

3:    if (p>=0.9) and (a<0.01) then begin icdfgamma:=-1; exit; end;
      if (b*q<=1E-8) then xn := exp(-(q/a+c))
                          else if (p<=0.9) then xn := exp(ln(p*g)/a)
                                           else xn := exp((ln(1-q)+gammln(a))/a);

      if (xn=0) then begin icdfgamma:=-1; exit; end;
      t := 0.5 + (0.5-xn/(a+1));
      xn := xn/t;
      if xn<0 then xn:=e;
      GOTO 1;

{   Selection of the initial approximation xn of x if a>1 }


2: if (q<=0.5) then w := ln(q) else w := ln(p);
   t := sqrt(-2*w);
   s := t - (((a3*t+a2)*t+a1)*t+a0)/((((b4*t+b3)*t+b2)*t+b1)*t+1);
   if (q>0.5) then s := -s;
   rta := sqrt(a);
   s2 := sqr(s);
   xn := a + s*rta + (s2-1)/3 + s* (s2-7)/ (36*rta) -
          ((3*s2+7)*s2-16)/ (810*a) +
          s* ((9*s2+256)*s2-433)/ (38880*a*rta);
   if xn<0 then xn:=0;
   if a<amin[iop] then GOTO 110;
   x := xn;
   d := 0.5 + (0.5-x/a);
   if (abs(d)<=dmin[iop]) then begin
     icdfgamma:=x; EXIT;
   end;

110:
   if (p<=0.5) then GOTO 130;
   if (xn<3.0*a) then GOTO 4;
   y := - (w+gammln(a));
   d := a* (a-1.0);
   if d<2 then d:=2;
   if (y<ln10*d) then GOTO 120;
   s := 1 - a;
   z := ln(y);
   GOTO 30;
```

```
120:
   t := a - 1.0;
   xn := y + t*ln(xn) - ln(1+ (-t/(xn+1)));
   xn := y + t*ln(xn) - ln(1+ (-t/(xn+1)));
   GOTO 4;

130:
   ap1 := a + 1;
   if (xn>0.70*ap1) then GOTO 170;
   w := w + gammln(ap1);
   if (xn>0.15*ap1) then GOTO 140;
   ap2 := a + 2;
   ap3 := a + 3;
   x := exp((w+x)/a);
   x := exp((w+x-ln(1+ (x/ap1)* (1+x/ap2)))/a);
   x := exp((w+x-ln(1+ (x/ap1)* (1+x/ap2)))/a);
   x := exp((w+x-ln(1+ (x/ap1)* (1+ (x/ap2)* (1+ x/ap3))))/a);
   xn := x;
   if (xn>1E-2*ap1) then GOTO 140;
   if (xn<=emin[iop]*ap1) then begin
     icdfgamma:=x; EXIT;
   end;
   GOTO 170;

140:
   apn := ap1;
   t := xn/apn;
   sum := 1 + t;

150:
   apn := apn + 1;
   t := t* (xn/apn);
   sum := sum + t;
   if (t>1E-4) then GOTO 150;
   t := w - ln(sum);
   xn := exp((xn+t)/a);
   xn := xn* (1- (a*ln(xn)-xn-t)/ (a-xn));
   GOTO 170;

{  SCHRODER ITERATION USING P   }

1: if (p>0.5) then GOTO 4;
170:
   if (p<=1E10*xmin) then begin
     ICDFGAMMA:=Xn;
     EXIT;
   end;
   am1 := (a-0.5) - 0.5;
180:
   if (a<=amax) then GOTO 190;
     d := 0.5 + (0.5-xn/a);
    if (abs(d)<=e2) then begin
      ICDFGAMMA:=Xn; EXIT;
    end;

190:
   if (ierr>=NUM_ITERATIONS) then begin
     icdfgamma:=x;
     exit;
   end;
   ierr := ierr + 1;
   qn:=gammq(a,xn);
   pn:=gammp(a,xn);
   if (pn=0.0) or (qn=0.0) then begin
     ICDFGAMMA:=xn;
     EXIT;
   end;
   r := exp(-xn)*power(xn,a)/calcgamma(a);
   if (r=0) then begin
     ICDFGAMMA:=xn;
     EXIT;
   end;
   t := (pn-p)/r;
   w := 0.5* (am1-xn);
   if (abs(t)<=0.1 ) and ( abs(w*t)<=0.1) then GOTO 200;
```

```
   x := xn* (1.0-t);
   if (x<=0.0) then begin
     ICDFGAMMA:=-1;
     EXIT;
   end;
   d := abs(t);
   GOTO 210;

200:
  h := t* (1.0+w*t);
  x := xn* (1.0-h);
  if (x<=0.0) then begin
    ICDFGAMMA:=-1;
    EXIT;
  end;
  if (abs(w)>=1.0 ) and ( abs(w)*t*t<=eps) then begin
    icdfgamma:=x;
    EXIT;
  end;
  d := abs(h);

210:
  xn := x;
  if (d>tol) then GOTO 180;
  if (d<=eps) or (abs(p-pn)<=tol*p) then begin
    icdfgamma:=x;
    EXIT;
  end;
  GOTO 180;

  {  SCHRODER ITERATION USING Q  }

4:
  if (q<=1E10*xmin) then begin
    ICDFGAMMA:=xn;
    EXIT;
  end;
  am1 := (a-0.5) - 0.5;
230:
  if (a<=amax) then GOTO 240;
  d := 0.5 + (0.5-xn/a);
  if (abs(d)<=e2) then begin
    ICDFGAMMA:=xn;
    EXIT;
  end;

240:
  if (ierr>=NUM_ITERATIONS) then begin
    ICDFGAMMA:=x;
    EXIT;
  end;
  ierr := ierr + 1;
  qn:=gammq(a,xn);
  pn:=gammp(a,xn);
  if (pn=0.0) OR  (qn=0.0) then begin
    ICDFGAMMA:=xn;
    EXIT;
  end;
  r := exp(-xn)*power(xn,a)/calcgamma(a);
  if (r=0.0) then begin
    ICDFGAMMA:=xn;
    EXIT;
  end;
  t := (q-qn)/r;
  w := 0.5* (am1-xn);
  if (abs(t)<=0.1 ) and ( abs(w*t)<=0.1) then GOTO 250;
  x := xn* (1.0-t);
  if (x<=0.0) then begin
    ICDFGAMMA:=-1;
    EXIT;
  end;
  d := abs(t);
  GOTO 260;

```

```
250:
  h := t* (1.0+w*t);
  x := xn* (1.0-h);
  if (x<=0.0) then begin
    ICDFGAMMA:=-1;
    EXIT;
  end;
  if (abs(w)>=1.0) AND (abs(w)*sqr(t)<=eps) then begin
    icdfgamma:=x;
    EXIT;
  end;
  d := abs(h);

260:
  xn := x;
  if (d>tol) then GOTO 230;
  if (d<=eps) then begin
    icdfgamma:=x;
    EXIT;
  end;
  if (abs(q-qn)<=tol*q) then begin
    icdfgamma:=x;
    EXIT;
  end;
  GOTO 230;
end;  {THIS ICDFGAMMA NIGHTMARE}


function cdfWeibull(XD,Scale,Shape : double) : single;
begin
  cdfWeibull:=1 - exp ( -1* power((XD/scale),shape));
end;

function icdfweibull(y,scale,shape: double): double;
begin
  icdfWeibull:=scale*power(-LN(1-y),(1/shape));
end;


function cdfLogistic(x,a,b:double): single;
begin
   cdfLogistic:=1/(1+exp(-(x-a)/b));
end;
function icdfLogistic(Prob,a,b:double): double;
begin
   icdfLogistic:=-b*ln((1/Prob)-1) + a;
end;

function icdfPoisson(y,lambda:double) : double;

const max_iterations = 25;
{ Iterative guess technique }
{ This acuracy requires a maximum of 25 iterations under normal circs. }

var err,yofx: double;
    xtop,xbot,xguess: integer;

begin
   Xtop:=round(lambda*2);  if lambda<70 then xtop:=round(lambda*3); if lambda<10 then xtop:=30;
   xbot:=0;
   xguess:=round((xtop-xbot)/2);
   repeat
     yofx:=gammq(xguess+1,lambda);
     if yofx < y then begin
       xbot:=xguess;
       xguess:=xguess+round((xtop-xguess)/2);
     end;
     if yofx > y then begin
       xtop:=xguess;
       xguess:=xguess-round((xguess-xbot)/2);
     end;
   until (xtop-xbot<=1) or (yofx=y);
   err:=abs(yofx-y);
   if abs(gammq(xguess,lambda)-y)<err then
     xguess:=xguess-1
```

```
   else if abs(gammq(xguess+2,lambda)-y)<err then
     xguess:=xguess+1;
   icdfpoisson:= xguess;
end;


function cdfPoisson(xd,lambda:double):single;

begin
  cdfPoisson:=GammQ(xd+1,lambda);   {numerical recipes}
end;


function cdfBinomial(xd,p:double;n:integer): single;
{XD = argument}
{p = Probability}
{n=  trials}

{See numerical recipes, p169. Faster, and it works. }

begin
  if xd>=n then cdfBinomial:=1
  else if xd<0 then cdfBinomial:=0
  else cdfBinomial := 1-betai(trunc(xd)+1,n-trunc(xd),p);
end;

function icdfBinomial(y,p : double; n: integer): double;

const max_iterations = 25;
{ Iterative guess technique }
{ This acuracy requires a maximum of 25 iterations under normal circs. }

var err,yofx: double;
    xguess,xtop,xbot: integer;

begin
  if (p<0) or (p>1) or (n<0) then begin
    icdfbinomial:=-1;
    exit;
  end;

  Xtop:=n;
  xbot:=0;
  xguess:=round((xtop-xbot)/2);
  repeat
    yofx:=cdfbinomial(xguess,p,n);
    if yofx < y then begin
      xbot:=xguess;
      xguess:=xguess+round((xtop-xguess)/2);
    end;
    if yofx > y then begin
      xtop:=xguess;
      xguess:=xguess-round((xguess-xbot)/2);
    end;
  until (xtop-xbot<=1) or (yofx=y);
  err:=abs(yofx-y);
  if abs(cdfbinomial(xguess,p,n)-y)<err then
    xguess:=xguess-1
  else if abs(cdfbinomial(xguess+2,p,n)-y)<err then
    xguess:=xguess+1;
  icdfbinomial:= xguess;
end;


function cdfgeometric(xd,p:double):single;

begin
  if (p<0) or (p>1) then begin
    cdfgeometric:=-1;
    exit;
  end;
  if xd>1e100 then
    cdfgeometric:=1
  else
```

```
    cdfgeometric:=1-power((1-p),(Trunc(xd)))
end;

function icdfGeometric(y,p : double): double;

begin
  if (p>1) or (p<0) then icdfgeometric:=-1 else
    if y=1 then icdfgeometric:=1e150 else
      if p=1 then icdfgeometric:=0 else
         icdfGeometric:=trunc(ln(1-y)/ln(1-p));
end;

function cdfBeta(x,a,b,scale:double):single;
begin
  if x>scale then x:=scale;
  cdfbeta:=betai(a,b,(x/scale));
end;

function icdfBeta(y,a,b,scale:double):double;
const acurate_to = 1e-8;
      max_iterations = 25;
{ Iterative guess technique }
{ This acuracy requires a maximum of 25 iterations under normal circs. }

var xguess,xtop,xbot,yofx: double;
    iterations: integer;

begin
  iterations:=0;
  xtop:=1;
  xbot:=0;
  xguess:=0.5;
  repeat
    iterations:=iterations+1;
    yofx:=betai(a,b,xguess);
    if yofx < y then begin
      xbot:=xguess;
      xguess:=xguess+(xtop-xguess)/2;
    end;
    if yofx > y then begin
      xtop:=xguess;
      xguess:=xguess-(xguess-xbot)/2
    end;
  until (abs(yofx-y)<acurate_to) or (iterations>max_iterations);
  icdfbeta:= xguess*scale;
end;


function cdfCauchy(x,a,b:double):single;
begin
  cdfCauchy:=0.5+(1/pi)*arctan((x-a)/b);
end;
function icdfCauchy(prob,a,b:double):double;

begin
  icdfCauchy:=b*(tan(pi*(prob-0.5)))+a
end;

function cdfpareto(x,x0,theta: double): single;
begin
   cdfpareto:=1-power(1+((x-x0)/x0),-theta);
end;

function icdfPareto(y,x0,theta:double):double;
begin
     icdfPareto:=x0+x0*(power(1-y,-1/theta)-1);
end;


function cdfHML(X,High, Medium, Low : double) : single;
begin
   if x>=High then   cdfHML:=1 else
   if x>=Medium then cdfHML:=0.6666666 else
   if x>=Low then    cdfHML:=0.3333333 else
                     cdfHML:=0;
```

```
end;
function icdfHML(Prob,High,Medium,Low : double) : double;
begin
   if Prob>0.6666666 then icdfHML:=High else
   if Prob>0.3333333 then icdfHML:=Medium else
                          icdfHML:=Low;
end;

begin
end.

```

code\SafeWaterConfig.pas

```
unit SafeWaterConfig;

interface

uses Classes, Types, IniFiles, StrUtils, SafewaterGlobals, SafeWaterUtility;

type
  TSafeWaterConfig = class
  private

  public
    OccFile, BWFile, ConsumptionFile, PWSSubpopFIle, CountySubpopfile,
    NationalSubpopFile, EJProportionsFile  : string;
    TechForecastFile: string;
    PWSFiles: TStringDynArray;
    CostOptions, BenefitOptions, OutputOptions: TStringDynArray;
    Workers: integer; //0 to calculate
    CPUMult: integer;
    MaxAge: integer;
    DiscountRate: double;
    DiscountRates: string;
    SmallLargeCat,   //for changnig size categories to small,large
    ICFCF,  //flag for running in ICF cohort follow style
    DebugOut, QuickRun, Uncertainty,
    AllCategories,
    Censor, ForcePFASConc, UseWTP, PFASEJOutput : boolean;
    SingleSystemCat : integer;
    DebugList: string;
    StartYear, EndYear : integer;

    //PFAS specific [PFAS]
    IncludePBPK,PFASCostHW: boolean;
    PBPKTimeStep: double;

    PFASChangeYear : integer;
    //used to calculate costs for single year
    SingleCostYear: integer;
    YearlyOut: boolean;
    ZeroDiscYearly: boolean; //

    DW_RSC: double;  //drinking water relative source conribution

    //not saved
    StartAt: integer;
    OverrideGUID, OverrideUnc: string;
    ForceConcFileName:string;
    PeriodAdjust: integer;

    constructor create;

    procedure LoadFromFile(aFileName: string);
    procedure SaveToFile(aFileName: string);

    procedure ExpandFileNames();

  end;

implementation

```

```
60  { TSafeWaterConfig }
61
62  constructor TSafeWaterConfig.create;
63  begin
64    //set defaults;
65    OccFile := AppPath + 'data\2022-04-01_PFAS_occurrence.csv';
66    BWFile := 'data\CohortData\BodyWeights.csv';
67    ConsumptionFile := 'data\CohortData\Consumption.csv';
68    NationalSubPopFile := 'data\CohortData\NationalPopProportionsAvg.csv';
69  //  NationalSubPopFile := 'data\CohortData\NationalPopProportions.csv';
70    PWSSubpopFIle := 'data\CohortData\PWS_PopPropAvg.csv';
71    CountySubpopfile := AppPath + 'data\CohortData\FIPS_PopProp.csv';
72    EJProportionsFile := AppPath + 'data\CohortData\PWS_EJProportions.csv';
73    TechForecastFile := AppPath + 'data\WBSCostCurves\TechForecast.csv';
74    setlength(PWSFiles,1);
75    PWSFiles[0] := 'data\SDWIS_Amended_Inventory_2022-07-07.csv';
76    CostOptions := SplitString('PFASOption1a',',');
77    BenefitOptions := SplitString('PFASCVD',',');
78    OutputOptions:=[];
79    Workers:=0;
80    CPUMult:=1;
81    MaxAge:=cMaxAge; //not saved for now
82    QuickRun := False;
83    DiscountRate := 0.03;
84    DiscountRates := '2,3,7';
85    DebugOut:=false;
86    DebugList:='';
87    SmallLargeCat:=false;
88    Uncertainty := True;
89    ICFCF := True;
90    Censor:=False;
91    UseWTP := False;
92    SingleSystemCat := 0;
93
94    ForcePFASConc:=false;
95    PBPKTimeStep := 1;
96    StartYear:=2024;
97    PFASChangeYear:=2029;
98    EndYear:=2105;
99    PFASEJOutput:=False;
100   IncludePBPK:=True;
101   PFASCostHW := false;
102   ZeroDiscYearly := True;
103   YearlyOut := true;
104   AllCategories := true;
105
106   SingleCostYear:=0;
107
108   DW_RSC:=0.2;
109   PeriodAdjust := 1;
110
111   StartAt:=0;
112   OverrideGUID:='';
113   OverrideUnc:='';
114   ForceConcFileName := AppPath + 'data\PFAS_VLS_occurrence_OptionConc.csv';
115 end;
116
117
118 procedure TSafeWaterConfig.ExpandFileNames;
119 begin
120   for var i := low(PWSFIles) to high(PWSFiles) do begin
121     var ss:=PWSFiles[i];
122     CodeToRoot(ss);
123     PWSFiles[i] := ss;
124   end;
125   CodeToRoot(OccFile);
126   CodeToRoot(BWFile);
127   CodeToRoot(ConsumptionFile);
128   CodeToRoot(PWSSubpopFile);
129   CodeToRoot(CountySubpopFile);
130   CodeToRoot(TechForecastFile);
131   CodeToRoot(EJProportionsFile);
132 end;
133
```

```
procedure TSafeWaterConfig.LoadFromFile(aFileName: string);
begin
  var fIni := TIniFile.Create(aFileName);
  var s: string;

  OverrideGUID := fIni.ReadString('General','OverrideGUID',OverrideGUID);
  OverrideUnc := fIni.ReadString('General','OverrideUnc',OverrideUnc);
  StartAt := fIni.ReadInteger('General','StartAt',StartAt);
  PeriodAdjust := fIni.ReadInteger('General','PeriodAdjust',PeriodAdjust);

  OccFile := fIni.ReadString('General','OccFile',OccFile);
  BWFile := fIni.ReadString('General','BWFile',BWFile);
  ConsumptionFile := fIni.ReadString('General','ConsumptionFile',ConsumptionFile);
  PWSSubpopFile := fIni.ReadString('General','PWSSubpopFile',PWSSubpopFile);
  CountySubpopFile := fIni.ReadString('General','CountySubpopFile',CountySubpopFile);
  EJProportionsFile := fIni.ReadString('General','EJProportionsFile',EJProportionsFile);
  PWSFiles := ReadIniList(fIni,'General','PWSFiles');
  CostOptions := ReadIniList(fIni,'General','CostOptions');
  DebugList := fIni.ReadString('General','DebugList', DebugList);
  DebugOut :=  fIni.ReadBool('General','DebugOut', DebugOut);
  QuickRun :=  fIni.ReadBool('General','QuickRun', QuickRun);
  AllCategories :=  fIni.ReadBool('General','AllCategories', AllCategories);
  BenefitOptions := ReadIniList(fIni,'General','BenefitOptions');
  OutputOptions := ReadIniList(fIni,'General','OutputOptions');
  Workers := fIni.ReadInteger('General','Workers',Workers);
  CPUMult :=  fIni.ReadInteger('General','CPUMult',CPUMult);
  IncludePBPK := fIni.ReadBool('General','IncludePBPK', IncludePBPK);
  YearlyOut := fIni.ReadBool('General','YearlyOut', YearlyOut);
  UseWTP := fIni.ReadBool('General','UseWTP', UseWTP);
  SingleSystemCat := fIni.ReadInteger('General','SingleSystemCat',SingleSystemCat);
  Uncertainty := fIni.ReadBool('General','Uncertainty', Uncertainty);
  SmallLargeCat := fIni.ReadBool('General','SmallLargeCat', SmallLargeCat);
  ICFCF := fIni.ReadBool('PFAS','ICFCF', ICFCF);
  Censor := fIni.ReadBool('General','Censor', Censor);
  ZeroDiscYearly := fIni.ReadBool('General','ZeroDiscYearly', ZeroDiscYearly);
  DiscountRate := fIni.ReadFloat('General','DiscountRate',DiscountRate);
  DiscountRates := fIni.ReadString('General','DiscountRates',DiscountRates);
  ForcePFASConc := fIni.ReadBool('PFAS','ForcePFASConc', ForcePFASConc);
  PFASCostHW := fIni.ReadBool('PFAS','PFASCostHW', PFASCostHW);
  PFASEJOutput := fIni.ReadBool('PFAS','PFASEJOutput', PFASEJOutput);
  PBPKTimeStep := fIni.ReadFloat('PFAS','PBPKTimeStep',PBPKTimeStep);
  STartYear := fIni.ReadInteger('General','StartYear',STartYear);
  EndYear := fIni.ReadInteger('General','EndYear',EndYear);
  PFASChangeYear := fIni.ReadInteger('PFAS','PFASChangeYear',PFASChangeYear);

  SingleCostYear := fIni.ReadInteger('Costs','SingleCostYear',SingleCostYear);
  TechForecastFile := fIni.ReadString('Costs','TechForecastFile',TechForecastFile);
  fIni.Free;
end;

procedure TSafeWaterConfig.SaveToFile(aFileName: string);
begin
  var fIni := TIniFile.Create(aFileName);

  fIni.WriteString('General','OverrideGUID',OverrideGUID);
  fIni.WriteString('General','OverrideUnc',OverrideUnc);
  fIni.WriteInteger('General','StartAt',StartAt);
  fIni.WriteInteger('General','PeriodAdjust',PeriodAdjust);
  fIni.WriteString('General','OccFile',OccFile);
  fIni.WriteString('General','BWFile',BWFile);
  fIni.WriteString('General','ConsumptionFile',ConsumptionFile);
  fIni.WriteString('General','PWSSubpopFile',PWSSubpopFile);
  fIni.WriteString('General','CountySubpopFile',CountySubpopFile);
  fIni.WriteString('General','EJProportionsFile',EJProportionsFile);
  WriteIniList(fIni,'General','PWSFiles', PWSFIles);
  WriteIniList(fIni,'General','CostOptions', CostOptions);
  WriteIniList(fIni,'General','BenefitOptions', BenefitOptions);
  WriteIniList(fIni,'General','OutputOptions', OutputOptions);
  fIni.WriteInteger('General','Workers',Workers);
  fIni.WriteInteger('General','CPUMult',CPUMult);
  fIni.WriteString('General','DebugList', DebugList);
  fIni.WriteInteger('General','SingleSystemCat',SingleSystemCat);
  fIni.WriteBool('General','DebugOut', DebugOut);
  fIni.WriteBool('General','QuickRun', QuickRun);
  fIni.WriteBool('General','Censor', Censor);
```

```
  fIni.WriteBool('General','ZeroDiscYearly', ZeroDiscYearly);
  fIni.WriteBool('General','AllCategories', AllCategories);
  fIni.WriteBool('General','SmallLargeCat', SmallLargeCat);
  fIni.WriteBool('General','IncludePBPK', IncludePBPK);
  fIni.WriteFloat('General','DiscountRate',DiscountRate);
  fIni.WriteString('General','DiscountRates',DiscountRates);
  fIni.WriteBool('General','Uncertainty', Uncertainty);
  fIni.WriteBool('General','UseWTP', UseWTP);
  fIni.WriteBool('General','YearlyOut', YearlyOut);
  fIni.WriteBool('PFAS','ICFCF', ICFCF);
  fIni.WriteFloat('PFAS','PBPKTimeStep',PBPKTimeStep);
  fIni.WriteFloat('General','StartYear',StartYear);
  fIni.WriteFloat('General','EndYear',EndYear);
  fIni.WriteFloat('PFAS','PFASChangeYear',PFASChangeYear);
  fIni.WriteBool('PFAS','ForcePFASConc', ForcePFASConc);
  fIni.WriteBool('PFAS','PFASCostHW', PFASCostHW);
  fIni.WriteBool('PFAS','PFASEJOutput', PFASEJOutput);

  fIni.WriteInteger('Costs','SingleCostYear',SingleCostYear);
  fIni.WriteString('Costs','TechForecastFile',TechForecastFile);
  fIni.UpdateFile;
  fIni.Free;
end;

end.

```

code\SafeWaterGlobals.pas

```
unit SafeWaterGlobals;

interface

uses SysUtils, Classes, Generics.Collections,
  {ifdef DEUBUG}
  CodeSiteLogging
  {endif}
  ;

type
  TDRArray = array[0..200] of double;
  PDRArray = ^TDRArray;
  //allow 5 DR per run + 0 (No Discounting)
  TMultiDRArray = array[0..5] of TDRArray;
  PMultiDRArray = ^TMultiDRArray;

var
  //these are truly global variables
  ImInGui : boolean = false;
  AppPath, DataPath, WorkPath, OutPath, CfgPath: string;

const
  swVersion = '1.1';
  swDate = '1/2024';

  cMaxAge = 85;  //max age to size benefit cohorts
  vMissing = -999;

  PPTtomgl = 1/1000000;
  cmgPerLtongPerml = 1000;  //mg/l to ng/ml


  //--- SystemType - used to represent the type of water system (community, non-transient, etc.) ---//
  sysCWS=0;
  sysNTNC=1;
  sysTNC=2;
  sysAny=10;

  //--- Ownership - used to represent the system ownership ---//
  oPublic=0;
  oPrivate=1;
  oAny=10;

```

```
  //--- SourceWater - used to represent the source of the system's water ---//
  swGroundwater=0;
  swSurfacewater=1;
  swAny=10;

  //--- SystemSize - used to represent the size of the system in number of people served ---//
  ss0_100=0;
  ss101_500=1;
  ss501_1000=2;
  ss1001_3300=3;
  ss3301_10k=4;
  ss10k_50k=5;
  ss50k_100k=6;
  ss100k_1m=7;
  ss1mp=8;
  ssAny=10;
  //alt sizes
  ss0_10k=11;
  ss10kp=12;

  gMale=0;   gFemale=1;

  //starting with the categories per ICF model doc for PFAS
  reWhite=0;   reBlack=1;    reHispanic=2;    reOther=3;

type
  TProgressProc = procedure (const Total, Progress : int64; const msg: string) of object;

  TAgeArray = array[0..cMaxAge] of single;
  TSubPopArray = array[gMale..gFemale,reWhite..reOther]  of TAgeArray;
  PSubPopArray = ^TSubPopArray;

  TPWSIdLookup = TDictionary<string,integer>;

  TCostInfo = record
     EntryPointCount, PWSCount : double; //informational
     SystemMeanConc, SystemMeanConcFinal: array[0..20] of double; //set to max length to allow easier zeroing
     AverageFlow, DesignFlow: double;  //these are per Entry Point

     SystemIsTreating, PWSPopTreating,
     EntryPointIsTreating, EntryPointPopTreating : double;
  end;

  TCostOutputs = record
     TotalAnnualCosts,
     AgencyImpAdmin, AgencyImpSampling,
     AgencyTreatmentAdmin,
     PWSImpAdmin, PWSImpAdmin_pws,
     PWSSampling, PWSSampling_pws,
     PWSTreatment,PWSTreatment_pws,
     PWSTreatmentAdmin,PWSTreatmentAdmin_pws,
     AgencyAnnualCosts,
     PWSAnnualCosts, PWSAnnualCosts_pws,
     HHAnnualCosts, HHAnnualCosts_pws,
     PWSAnnualCosts_pws_act,
     HHAnnualCosts_pws_act : double;

     CapitalCost_pws, OMCost_pws,
     CapitalCost, OMCost: double;  // saving these seperately for info and yearly output

     CRG1, CRG3: double;  //Cost Revenue Ratio tests >1%, >3%

HHAll: double;

     DontCare: double;  // trash can for compiled costs that don't have a _pws value
     function DebugLine: string;
  end;
  TMDRCostOutputs = array of TCostOutputs;
  TMDRYearlyCostOutputs = array of array of TCostOutputs;

  TFlowParams=record
    PubGroundA,PubGroundB,D_PubGroundA,D_PubGroundB,
    PrivGroundA,PrivGroundB,D_PrivGroundA,D_PrivGroundB : double;
    PubSurfA,PubSurfB,D_PubSurfA,D_PubSurfB,
```

```
    PrivSurfA,PrivSurfB,D_PrivSurfA,D_PrivSurfB : double;
  end;




procedure CSL(msg: string);
procedure CSLS(msg: TStringList);

implementation

procedure CSL(msg: string);
begin
  {ifdef DEUBUG}
  Codesite.Send(msg);
  {endif}
end;
procedure CSLS(msg: TstringList);
begin
  {ifdef DEUBUG}
  Codesite.Send(0,'db',msg);
  {endif}
end;

{ TCostOutputs }

function TCostOutputs.DebugLine: string;
begin
  Result := Format('TotalAnnualCosts:%g  AgencyImpAdmin:%g  AgencyImpSampling:%g  '+
     'AgencyTreatmentAdmin:%g '+
     'PWSImpAdmin:%g  PWSImpAdmin_pws:%g  PWSSampling:%g  PWSSampling_pws:%g  '+
     'PWSTreatmentAdmin:%g  PWSTreatmentAdmin_pws:%g'+
     'PWSTreatment:%g  PWSTreatment_pws:%g  PWSAnnualCosts:%g  PWSAnnualCosts_pws:%g  '+
     'HHAnnualCosts:%g  HHAnnualCosts_pws:%g PWSAnnualCosts_pws_act:%g  HHAnnualCosts_pws_act:%g',[
     TotalAnnualCosts, AgencyImpAdmin, AgencyImpSampling, AgencyTreatmentAdmin,
     PWSImpAdmin, PWSImpAdmin_pws, PWSSampling, PWSSampling_pws,
     PWSTreatmentAdmin, PWSTreatmentAdmin_pws,
     PWSTreatment,PWSTreatment_pws, PWSAnnualCosts, PWSAnnualCosts_pws,
     HHAnnualCosts, HHAnnualCosts_pws, PWSAnnualCosts_pws_act, HHAnnualCosts_pws_act]);

end;

end.

```

code\SafeWaterMetrics.pas

```
unit SafeWaterMetrics;

interface

uses SysUtils,Types, Classes, SafeWaterUncertBucket, Generics.Collections,
     Math, StrUtils,Agglomerators, SafewaterUtility, System.Generics.Defaults,
     Safewaterglobals
     ;

const

 mtCost=0;
 mtBenefitDollars=1;
 mtBenefitCases=2;
 mtNetBenefits=3;
 mtInfo=4;
 mtEJOutput=5;
 mtCostExtended=6;
 mtCostYearly=7;
 mtBenefitYearly=8;
 mtAdhoc=9;

 //used to control what is tracked and ouput
 mcAll=0;
 mcMeanSum=1;
 mcMeanOnly=2;
```

```
27  mcSumOnly=3;
28
29  MetricTypeText: array[0..9] of string = ('Costs','Benefit Dollars',
30  'Benefit Cases','Net Benefits','Information','EJ Output',
31  'Additional Cost Output','Costs By Year', 'Benefits By year', 'Adhoc');
32
33  statText =
34   'P1,P10,P25,P50,P75,P90,P99,Max,Min,Mean,StdDev,Sum,N';
35
36 type
37  TMetricSummary=record
38    P1,P10,P25,P50,P75,P90,P99,Max,Min,Mean,StdDev,Sum, N: double;
39    NANInfCount: integer;
40
41  end;
42
43  TMetric = class
44  private
45     FName, FDesc: string;
46     FCType: integer;
47     FIgnoreZero: boolean;
48     MetricType: integer;
49     FDummy: double;
50     PV: PDouble;
51     PGate: PDouble;  //used to skip collection of point when value <> 0
52     SQ: TStreamQuantiles;
53     FTrials: TDoubleDynArray;
54  public
55    constructor create(aName,ADesc: string; P : PDouble; aMetricType: integer; MeanSumOnly: integer;  IgnoreZero: boolean; CollectGate: PDouble = nil;
aEpsilon: double = 0.01);
56    destructor Destroy; override;
57
58    procedure CollectPoint(const aWght: double = 1);
59    procedure AddPoint(const aV: double; aWght: double = 1);
60    //for use in draw level final output
61    procedure StoreTrial(const aV: double; const aIter, aTotalTrials: integer);
62
63    procedure SaveSummaryToStream(T: TStream);
64    procedure SaveSummaryToFile(aFileName: string);
65    function SaveSummaryToString: string;
66    function P100CSVList: string;
67    function P100CSVListHdr: string;
68    function AllCSVList: string;
69    function AllCSVListHdr: string;
70
71    property Name: string read FName write FName;
72    property Desc: string read FDesc write FDesc;
73  end;
74
75  TMetricSorter=record
76    M: TMetric;
77    Key: string;
78    RegOption,Source,Size,Ownership,SysType,MetricName,MetricDesc,MetricClass,Stat, FileTag: string;
79    procedure DefaultSortKey;
80    procedure YearlySortKey;
81    function GetCSV: string;
82    function GetCSVNoDesc: string;
83    function GetCSVSummary: string;
84  end;
85
86
87  TMetricList = class(TObjectList<TMetric>)
88    constructor Create(); reintroduce;
89    destructor Destroy; override;
90
91    function FindMetric(aName: string): TMetric;
92    procedure AddMetric(aName,aDesc: string; P : PDouble; aMetricType: integer; MeanSumOnly: integer; IgnoreZero: boolean; CollectGate: PDouble = nil;
aEpsilon: Double = 0.01);
93    function Clone: TMetricList;
94  end;
95
96  TCategoryMetrics = class
97  private
98    AllStrataCode: integer;
99    CategoryMetrics: TObjectDictionary<integer,TMetricList>;
```

```
100     FFileNameAdd: string;
101     FFileTags: TStringlist;
102     FInfo: TStringList;
103     FCollectList: TObjectDictionary<string,TMetric>;
104
105     procedure AddToCollect(const ML: TMetricList; const Strata, Name: string; const aValue: double; const FirstFile: boolean; aIter, aTotalTrials:
integer);
106     function GetCount: integer;
107     function GetTrackedCount: integer;
108     procedure iSaveUncResultFile(aFilename, aFileTag: string);
109     procedure ySaveUncResultFile(aFilename, aFileTag: string);
110     procedure iSaveNetBenefitResultFile(aFilename, aFileTag: string);
111   public
112     constructor create(aFileNameAdd: string);
113     destructor Destroy; override;
114
115     //called during intialization - fills "all" category to be cloned by others
116     procedure AddMetric(aName,aDesc: string; P : PDouble; aMetricType: integer; MeanSumOnly: integer; IgnoreZero:boolean; FileTag: string = '';
CollectGate: PDouble = nil; aEpsilon: Double = 0.01);
117     procedure CollectPoints(StrataCode: integer; aWght: double = 1);
118
119     procedure SaveIterToFile(aFileName: string);
120     function SaveIterToStrings: TStringList;
121     procedure LoadIterFromFile(aFileName: string; FirstFile: boolean; aIter, aTotalTrials: integer);
122
123     procedure SaveUncResultFile(aFilename: string; aFirstTagOnly: boolean);
124     property Info: TStringLIst read FInfo;
125     property Count: integer read getCount;
126     property TrackedCount: integer read getTrackedCount;
127   end;
128
129
130 implementation
131
132 procedure SplitUpName(Name: string; var Group,VarName,POstfix, FileTag, Year: string);
133 var s: string;
134 begin
135   s:=Name;
136   var i:=pos('.',Name);
137   Group:='';
138   if i>0 then begin
139     Group := copy(s,1,i-1);
140     delete(s,1,i);
141   end;
142
143   //find postfix -  at end
144   i:=pos('~',s);
145   PostFix:='';
146   if i>0 then begin
147     PostFix:=copy(s,i+1, 10000);
148     delete(s,i,10000);
149   end;
150
151   //find file tag
152   i:=pos('FFFF',s);
153   FileTag:='';
154   if i>0 then begin
155     FileTag:=copy(s,i+4, 10000);
156     delete(s,i,10000);
157   end;
158
159   //find year tag
160   i:=pos('YYYY',s);
161   Year:='';
162   if i>0 then begin
163     Year:=copy(s,i+4, 4);
164     delete(s,i,8);
165   end;
166
167   //varname remains
168   VarName := s;
169 end;
170
171 { TMetric }
172
```

```
procedure TMetric.AddPoint(const aV: double; aWght: double);
begin
  SQ.AddPoint(aV,aWght);
end;

procedure TMetric.CollectPoint(const aWght: double = 1);
begin
  if PGate^<>0 then
    SQ.AddPoint(PV^,aWght);
end;

constructor TMetric.create(aName,aDesc: string; P : PDouble; aMetricType: integer; MeanSumOnly: integer; IgnoreZero: boolean; CollectGate: PDouble = nil; aEpsilon: double = 0.01);
begin
  FName := aName;
  FDesc := aDesc;
  PV := P;
  FDummy := 1;
  if CollectGate<>nil then
    PGate := CollectGate
  else
    PGate := @FDummy;
  //Note: setting epsilon here to a static value - maybe introduce in config
  SQ := TStreamQuantiles.Create(aEpsilon);
  FIgnoreZero := IgnoreZero;
  SQ.IgnoreZero := IgnoreZero;
  FCtype := MeanSumOnly;
  SQ.MeanOnly := (FCType > 0);
  MetricType := aMetricType;
  setlength(FTrials,0);
end;

destructor TMetric.destroy;
begin
  SQ.Free;
  inherited;
end;

function TMetric.AllCSVList: string;
begin
  Result:='';
  if length(FTrials)=0 then exit;
  Result := FloatToStrF(FTrials[0],ffExponent,6,2);
  for var i := 1 to high(FTrials) do
   Result := Result + ',' + FloatToStrF(FTrials[i],ffExponent,6,2);
end;

function TMetric.AllCSVListHdr: string;
begin
  Result:='';
  if length(FTrials)=0 then exit;
  Result := 'I1';
  for var i := 1 to high(FTrials) do
   Result := Result + ',I' + inttostr(i+1);

end;

function TMetric.P100CSVList: string;
begin
  Result:=FloatToStrF(SQ.Min,ffExponent,6,2);
  var p:double := 0.005;
  Result:=Result+','+FloatToStrF(SQ.Quantile(p),ffExponent,6,2);
  while P<=0.995 do begin
    p:=p+0.01;
    Result:=Result+','+FloatToStrF(SQ.Quantile(p),ffExponent,6,2);
  end;
  Result:=Result+','+FloatToStrF(SQ.Max,ffExponent,6,2);
end;

function TMetric.P100CSVListHdr: string;
begin
  Result:='P000';
  var s:string;
  var p:double := 0.005;
  var i: integer;
```

```
  Result:=Result+',P005';
  while P<=0.995 do begin
    p:=p+0.01;
    i:=round(p*1000);
    if i<100 then
      Result:=Result+',P0'
    else
      Result:=Result+',P';
    Result:=Result+inttostr(i);
  end;
  Result:=Result+',P1000';

end;

procedure TMetric.SaveSummaryToStream(T: TStream);
begin
  var S: TMetricSummary;
  S.P1 := SQ.Quantile(0.01);
  S.P10 := SQ.Quantile(0.10);
  S.P25 := SQ.Quantile(0.25);
  S.P50 := SQ.Quantile(0.50);
  S.P75 := SQ.Quantile(0.75);
  S.P90 := SQ.Quantile(0.90);
  S.P99 := SQ.Quantile(0.99);
  S.Min := SQ.Min;
  S.Max := SQ.Max;
  S.Mean := SQ.Mean;
  S.Sum := SQ.Sum;
  S.StdDev := SQ.StdDev;
  S.N := SQ.NumPoints;
  S.NANInfCount := SQ.fNANInfCounter;
  T.Write(S, sizeof(s));
end;

procedure TMetric.SaveSummaryToFile(aFileName: string);
begin
  var S: TMetricSummary;
  S.P1 := SQ.Quantile(0.01);
  S.P10 := SQ.Quantile(0.10);
  S.P25 := SQ.Quantile(0.25);
  S.P50 := SQ.Quantile(0.50);
  S.P75 := SQ.Quantile(0.75);
  S.P90 := SQ.Quantile(0.90);
  S.P99 := SQ.Quantile(0.99);
  S.Min := SQ.Min;
  S.Max := SQ.Max;
  S.Mean := SQ.Mean;
  S.Sum := SQ.Sum;
  S.StdDev := SQ.StdDev;
  S.N := SQ.NumPoints;
  S.NANInfCount := SQ.fNANInfCounter;
  var T := TStreamWriter.Create(aFileName,FileExists(aFilename));
  T.Writeline(S.P1.ToString+','+  S.P10.ToString+','+  S.P25.ToString+','+
  S.P50.ToString+','+  S.P75.ToString+','+  S.P90.ToString+','+  S.P99.ToString+','+
  S.Min.ToString+','+  S.Max.ToString+','+  S.Mean.ToString+','+  S.Sum.ToString+','+
  S.StdDev.ToString+','+  S.N.ToString+','+  S.NANInfCount.ToString);
  T.Free;
end;

function TMetric.SaveSummaryToString: string;
begin
  var S: TMetricSummary;
  S.P1 := SQ.Quantile(0.01);
  S.P10 := SQ.Quantile(0.10);
  S.P25 := SQ.Quantile(0.25);
  S.P50 := SQ.Quantile(0.50);
  S.P75 := SQ.Quantile(0.75);
  S.P90 := SQ.Quantile(0.90);
  S.P99 := SQ.Quantile(0.99);
  S.Min := SQ.Min;
  S.Max := SQ.Max;
  S.Mean := SQ.Mean;
  S.Sum := SQ.Sum;
  S.StdDev := SQ.StdDev;
  S.N := SQ.NumPoints;
```

```
322   S.NANInfCount := SQ.fNANInfCounter;
323   Result := 'P1:'+S.P1.ToString+#13#10+
324   'P10:'+  S.P10.ToString+#13#10+
325   'P25:'+  S.P25.ToString+#13#10+
326   'P50:'+ S.P50.ToString+#13#10+
327   'P75:'+  S.P75.ToString+#13#10+
328   'P90:'+  S.P90.ToString+#13#10+
329   'P99:'+  S.P99.ToString+#13#10+
330   'Min:'+  S.Min.ToString+#13#10+
331   'Max:'+  S.Max.ToString+#13#10+
332   'Mn:'+  S.Mean.ToString+#13#10+
333   'Sum:'+  S.Sum.ToString+#13#10+
334   'StdDev:'+S.StdDev.ToString+#13#10+
335   'N:'+  S.N.ToString+#13#10+
336   'NanInf:'+  S.NANInfCount.ToString;
337 end;
338
339
340
341 procedure TMetric.StoreTrial(const aV: double; const aIter,
342   aTotalTrials: integer);
343 begin
344   if length(FTrials)=0 then setlength(FTrials,aTotalTrials);
345   FTrials[aIter] := aV;
346 end;
347
348 { TMetricList }
349
350 procedure TMetricList.AddMetric(aName,aDesc: string; P: PDouble; aMetricType: integer; MeanSumOnly: integer; IgnoreZero: boolean; CollectGate: PDouble
    = nil; aEpsilon: Double = 0.01);
351 begin
352   if FindMetric(aName) = nil  then
353     Add(TMetric.create(aName,aDesc,p, aMetricType, MeanSumOnly, IgnoreZero, CollectGate, aEpsilon));
354 end;
355
356 function TMetricList.Clone: TMetricList;
357 begin
358   Result := TMetricList.Create;
359   for var T in self do begin
360     Result.AddMetric(T.Name, T.Desc, T.PV, T.MetricType, T.FCType, T.FIgnoreZero, T.PGate);
361   end;
362 end;
363
364 constructor TMetricList.create;
365 begin
366   inherited create(true);
367 end;
368
369 destructor TMetricList.destroy;
370 begin
371   inherited;
372 end;
373
374 function TMetricList.FindMetric(aName: string): TMetric;
375 begin
376   Result := nil;
377   for var T in self do begin
378     if (T.FName = aName) then begin
379       Result:=T;
380       exit;
381     end;
382   end;
383 end;
384
385 { TCategoryMetrics }
386
387 procedure TCategoryMetrics.AddMetric(aName, aDesc: string; P: PDouble; aMetricType: integer; MeanSumOnly: integer; IgnoreZero:boolean; FileTag: string
    = ''; CollectGate: PDouble = nil; aEpsilon: Double = 0.01);
388 begin
389   var T: TMetricList;
390   if not CategoryMetrics.TryGetValue(AllStrataCode,T) then begin
391     T := TMetricLIst.Create;
392     CategoryMetrics.Add(AllStrataCode,T);
393   end;
394   var sName := aName;
```

```
395   if FileTag <> '' then begin
396     if FFileTags.IndexOf(FileTag)<0 then FFileTags.Add(FileTag);
397     sName := sName +'FFFF'+FileTag;
398   end;
399   T.AddMetric(sName, aDesc, P, aMetricType, MeanSumOnly, IgnoreZero, CollectGate, aEpsilon);
400 end;
401
402 procedure TCategoryMetrics.AddToCollect(const ML: TMetricList; const Strata, Name: string;
403   const aValue: double; const FirstFile: boolean; aIter, aTotalTrials: integer);
404 begin
405   var M: TMetric;
406   if not FCollectLIst.TryGetValue(Strata+Name,M) then begin
407     if not FirstFile then exit;
408     M := ML.FindMetric(Name);
409     if Assigned(M) then begin
410       FCollectLIst.Add(Strata+Name,M);
411     end else
412       exit;
413   end;
414   //TODO formalize this concept - currently just doing it for a special class of metrics...
415   if M.MetricType = mtCostExtended then
416     M.StoreTrial(aValue, aIter, aTotalTrials);
417   M.AddPoint(aValue);
418 end;
419
420 procedure TCategoryMetrics.CollectPoints(StrataCode: integer; aWght: double);
421 begin
422   var T: TMetricList;
423   if not CategoryMetrics.TryGetValue(StrataCode,T) then begin
424     T := CategoryMetrics.Items[AllStrataCode].Clone;
425     CategoryMetrics.Add(StrataCode,T);
426   end;
427   for var M in T do begin
428     M.CollectPoint(aWght);
429   end;
430 end;
431
432 constructor TCategoryMetrics.create(aFileNameAdd: string);
433 begin
434   CategoryMetrics := TObjectDictionary<integer,TMetricList>.create([doOwnsValues]);
435   AllStrataCode := StrataToCode(swAny,ssAny,oAny,sysAny);
436   FInfo:=TSTringLIst.Create;
437   FCollectList := TObjectDictionary<string,TMetric>.Create([]);
438   FFileNameAdd := aFileNameAdd;
439   FFileTags := TStringLIst.Create;
440 end;
441
442 destructor TCategoryMetrics.destroy;
443 begin
444   CategoryMetrics.Free;
445   FInfo.Free;
446   FCollectList.Free;
447   FFileTags.Free;
448   inherited;
449 end;
450
451 function TCategoryMetrics.GetCount: integer;
452 begin
453   Result := CategoryMetrics.Items[AllStrataCode].Count;
454 end;
455
456 function TCategoryMetrics.GetTrackedCount: integer;
457 begin
458   Result := CategoryMetrics.Count * CategoryMetrics.Items[AllStrataCode].Count;
459 end;
460
461 procedure TCategoryMetrics.LoadIterFromFile(aFileName: string; FirstFile: boolean; aIter, aTotalTrials: integer);
462 begin
463   //not clearing anything here because many files will be read...
464   var T := TBufferedFileStream.Create(aFileName,fmOPenRead,4096*8);
465   var MetricCount:integer;
466   var Name, Desc,Group,VarName,POstfix, FileTag, Year: string;
467   var MT:integer;
468   var MeanOnly: integer;
469   const aEpsilon = 0.001;
```

```
470
471   T.Read(MetricCount, sizeof(MetricCount));
472   for var i := 0 to MetricCount-1 do begin
473     Name := ReadStreamStr(T);
474     Desc := ReadStreamStr(T);
475     T.Read(MT,SizeOf(MT));
476     T.Read(MeanOnly,SizeOf(MeanOnly));
477     SplitUpName(Name,Group,VarName,POstfix, FileTag,Year);
478     if (FFileTags.IndexOf(FileTag)<0) and (FileTag<>'') then FFileTags.Add(FileTag);
479
480 if MT=mtCostExtended then
481   if pos('pop',Name)<2 then continue;
482
483     if MeanOnly=mcAll then begin
484       AddMetric(Name +'~P10', Desc,nil,MT,mcAll, False,'',nil,aEpsilon);
485       AddMetric(Name +'~P25', Desc,nil,MT,mcAll, False,'',nil,aEpsilon);
486       AddMetric(Name +'~P50', Desc,nil,MT,mcAll, False,'',nil,aEpsilon);
487       AddMetric(Name +'~P75', Desc,nil,MT,mcAll, False,'',nil,aEpsilon);
488       AddMetric(Name +'~P90', Desc,nil,MT,mcAll, False,'',nil,aEpsilon);
489     end;
490     if MeanOnly<>mcSumOnly then
491       AddMetric(Name +'~Mean', Desc,nil,MT,mcAll, False,'',nil,aEpsilon);
492     if MeanOnly<>mcMeanOnly then
493       AddMetric(Name +'~Sum', Desc,nil,MT,mcAll, False,'',nil,aEpsilon);
494   end;
495
496   var StrataCode: integer;
497   var S: TMetricSummary;
498   var sStrataCode: string;
499   while T.Position < T.Size do begin
500     T.Read(StrataCode, sizeof(StrataCode));
501     sStrataCode := inttostr(StrataCode);
502     var ML:TMetricList;
503     var M: TMetric;
504     if not CategoryMetrics.TryGetValue(StrataCode,ML) then begin
505       ML := CategoryMetrics.Items[AllStrataCode].Clone;
506       CategoryMetrics.Add(StrataCode,ML);
507     end;
508
509     for var i := 0 to MetricCount-1 do begin
510       Name := ReadStreamStr(T);
511       T.Read(S,SizeOf(S));
512
513       AddToCollect(ML,sStrataCode,Name+'~P10',S.P10,FirstFile, aIter, aTotalTrials);
514       AddToCollect(ML,sStrataCode,Name+'~P25',S.P25,FirstFile, aIter, aTotalTrials);
515       AddToCollect(ML,sStrataCode,Name+'~P50',S.P50,FirstFile, aIter, aTotalTrials);
516       AddToCollect(ML,sStrataCode,Name+'~P75',S.P75,FirstFile, aIter, aTotalTrials);
517       AddToCollect(ML,sStrataCode,Name+'~P90',S.P90,FirstFile, aIter, aTotalTrials);
518       AddToCollect(ML,sStrataCode,Name+'~Mean',S.Mean,FirstFile, aIter, aTotalTrials);
519       AddToCollect(ML,sStrataCode,Name+'~Sum',S.Sum,FirstFile, aIter, aTotalTrials);
520
521       if S.NANInfCount>0 then begin
522         FInfo.Add(format('NANInf detected:%s  count:%d,  file:%s  strata:%s ',
523                   [Name, S.NANInfCount, aFilename, CodeToStrataStrLong(StrataCode)]));
524       end;
525     end;
526   end;
527   T.Free;
528 end;
529
530 procedure TCategoryMetrics.SaveIterToFile(aFileName: string);
531 begin
532   var T := TBufferedFileStream.Create(aFileName,fmCreate,4096*8);
533   var MetricCount:integer := CategoryMetrics.Items[AllStrataCode].Count;
534   T.Write(MetricCount, sizeof(MetricCount));
535   for var i := 0 to MetricCount-1 do begin
536     WriteStreamStr(T,CategoryMetrics.Items[AllStrataCode].Items[i].FName);
537     WriteStreamStr(T,CategoryMetrics.Items[AllStrataCode].Items[i].FDesc);
538     T.Write(CategoryMetrics.Items[AllStrataCode].Items[i].MetricType,sizeof(CategoryMetrics.Items[AllStrataCode].Items[i].MetricType));
539     T.Write(CategoryMetrics.Items[AllStrataCode].Items[i].FCtype,sizeof(CategoryMetrics.Items[AllStrataCode].Items[i].FCType));
540   end;
541
542   for var StrataCode:integer in CategoryMetrics.Keys do begin
543     T.Write(StrataCode, sizeof(StrataCode));
```

```
544     var ML := CategoryMetrics.Items[StrataCode];
545     for var M in ML do begin
546       WriteStreamStr(T,M.FName);
547       M.SaveSummaryToStream(T);
548     end;
549   end;
550   T.Free;
551 end;
552
553 function TCategoryMetrics.SaveIterToStrings: TStringList;
554 begin
555   Result := TStringLIst.Create;
556   var Source,Size,Ownership,SysType: string;
557   for var StrataCode:integer in CategoryMetrics.Keys do begin
558     CodeToStrataStr(StrataCode,Source,Size,Ownership,SysType);
559     Result.Add(Source+','+Size+','+Ownership+','+SysType);
560     var ML := CategoryMetrics.Items[StrataCode];
561     for var M in ML do begin
562       Result.Add(M.Fname);
563       var s := M.SaveSummaryToString;
564       Result.Add(S);
565     end;
566   end;
567
568 end;
569
570 function _df(const d: double) : string;
571 begin
572   if abs(d)>1000 then
573     result:=FormatFloat('#,###',d)
574   else
575     result:=Format('%g',[d]);
576 end;
577
578
579 procedure TCategoryMetrics.ySaveUncResultFile(aFilename, aFileTag: string);
580 var BenLines, CostLines : array of array of double;
581 begin
582   var F0: boolean := aFileTag = '0';
583   var FA: string := '_'+FFileNameAdd+'('+aFileTag+').csv';
584   var FR: string := ChangeFileExt(aFilename,'');
585   var T := TStreamWriter.Create(FR + FA ,false, TEncoding.ASCII,4096*8);
586   var T2 := TStreamWriter.Create(FR+'_Summary' + FA ,false, TEncoding.ASCII,4096*8);
587
588   var RowId := TStringList.Create;
589   var CostOut := TStringList.Create;
590   var BenOut := TStringList.Create;
591   var ML: TMetricList;
592   var M: TMetric;
593   var ix: integer;
594   CostOut.Sorted := True;
595   BenOut.Sorted := True;
596   RowID.Sorted := True;
597   var Group,VarName,Postfix, FileTag, Year, rowname: string;
598   var StrataCode: integer;
599
600   for StrataCode in CategoryMetrics.Keys do begin
601     ML := CategoryMetrics.Items[Stratacode];
602     for M in ML do begin
603       if not ((M.MetricType=mtCostYearly) or (M.MetricType=mtBenefitYearly)) then continue;
604       SplitUpName(M.FName,Group,VarName,POstfix, FileTag, Year);
605       if M.MetricType=mtCostYearly then begin
606         if CostOut.IndexOf(VarName)<0 then CostOut.Add(varName)
607       end else begin
608         if BenOut.IndexOf(VarName)<0 then BenOut.Add(varName);
609       end;
610       RowName := inttostr(StrataCode) +'_'+Group+'.'+Year;
611       if RowId.IndexOf(RowName)<0 then RowId.Add(RowName);
612     end;
613   end;
614
615   setlength(BenLines,RowID.Count, BenOut.Count*3);
616   setlength(CostLines,RowID.Count, CostOut.Count*3);
617   //reread and save to new structure;
618   for StrataCode in CategoryMetrics.Keys do begin
```

```
619     ML := CategoryMetrics.Items[Stratacode];
620     for M in ML do begin
621       if not ((M.MetricType=mtCostYearly) or (M.MetricType=mtBenefitYearly)) then continue;
622       SplitUpName(M.FName,Group,VarName,POstfix, FileTag, Year);
623       if FileTag <> aFileTag then continue;
624       RowName := inttostr(StrataCode) +'_'+Group+'.'+Year;
625       var RowIX := RowId.IndexOf(RowName);
626       if M.MetricType=mtCostYearly then begin
627         ix := CostOut.IndexOf(VarName) * 3;
628         CostLines[RowIX,ix] := M.SQ.Quantile(0.05);
629         CostLines[RowIX,ix+1] := M.SQ.Mean;
630         CostLines[RowIX,ix+2] := M.SQ.Quantile(0.95);
631       end else begin
632         ix := BenOut.IndexOf(VarName) * 3;
633         BenLines[RowIX,ix] := M.SQ.Quantile(0.05);
634         BenLines[RowIX,ix+1] := M.SQ.Mean;
635         BenLines[RowIX,ix+2] := M.SQ.Quantile(0.95);
636       end;
637     end;
638   end;
639
640   var s :='RegOption,Size,Source,Ownership,Type,Year';
641   for var i := 0 to CostOut.Count -1  do begin
642     s := s + ',' + CostOut[i]+'_(5th),'+ CostOut[i]+'_(Mn),'+ CostOut[i]+'_(95th)';
643   end;
644   for var i := 0 to BenOut.Count -1  do begin
645     s := s + ',' + BenOut[i]+'_(5th),'+ BenOut[i]+'_(Mn),'+ BenOut[i]+'_(95th)';
646   end;
647   T.Writeline(s);
648
649   s :='RegOption,Year';
650   for var i := 0 to CostOut.Count -1  do begin
651     s := s + ',' + CostOut[i];
652   end;
653   for var i := 0 to BenOut.Count -1  do begin
654     s := s + ',' + BenOut[i];
655   end;
656   T2.Writeline(s);
657
658   var Source,Size,Ownership,SysType, RegOption, Yr, any: string;
659   for var i:=0 to RowId.Count-1 do begin
660     s := RowID[i];
661     ix := pos('_',s);
662     StrataCode := strtoint(copy(s,1,ix-1));
663     CodeToStrataStr(StrataCode,Source,Size,Ownership,SysType);
664     Delete(s,1,ix);
665     ix := pos('.',s);
666     RegOption := Copy(s,1,ix-1);
667     Delete(s,1,ix);
668     Yr := s;
669     s := Format('"%s","%s","%s","%s","%s",%s',
670             [RegOption,Size,Source,Ownership,SysType,Yr]);
671     for var j := 0 to high(CostLines[i])  do
672       s := s + Format(',%2.5e',[CostLines[i,j]]);
673     for var j := 0 to high(BenLines[i])  do
674       s := s + Format(',%2.5e',[BenLines[i,j]]);
675     T.Writeline(s);
676     any := lowercase(Size + Source + Ownership +SysType);
677     if any='anyanyanyany' then begin
678       s := Format('"%s",%s',
679               [RegOption,Yr]);
680       for var j := 0 to high(CostLines[i])  do begin
681         if j mod 3 = 1 then
682           s := s + Format(',%2.5e',[CostLines[i,j]]);
683       end;
684       for var j := 0 to high(BenLines[i])  do begin
685         if j mod 3 = 1 then
686           s := s + Format(',%2.5e',[BenLines[i,j]]);
687       end;
688       T2.Writeline(s);
689     end;
690
691   end;
692   T.Free;
```

```
693   T2.Free;
694   RowID.Free;
695   BenOut.Free;
696   CostOut.Free;
697 end;
698
699 procedure TCategoryMetrics.iSaveNetBenefitResultFile(aFilename, aFileTag: string);
700 begin
701   var F0: boolean := aFileTag = '0';
702   var FA: string := '_'+FFileNameAdd+'('+aFileTag+').csv';
703   var FA2: string := '_'+FFileNameAdd+'('+aFileTag+').txt';
704   var FR: string := ChangeFileExt(aFilename,'');
705   var T := TStreamWriter.Create(FR + FA ,false, TEncoding.ASCII,4096*8);
706   var T2 := TStreamWriter.Create(FR+'_National' + FA2 ,false, TEncoding.ASCII,4096*8);
707   var Group,VarName,Postfix, FileTag, Year, Source,Size,Ownership,SysType: string;
708   var i,StrataCode: integer;
709   var ML: TMetricList;
710   var M: TMetric;
711   var s,s2,any: string;
712   var p,v: double;
713   const NL = #13#10;
714
715   s:='RegOption,Size,Source,Ownership,Type';
716   for i := 1 to 100 do begin
717      s := s + ',P'+floattostr(i-0.5);
718   end;
719   T.WriteLine(s);
720   for StrataCode in CategoryMetrics.Keys do begin
721     ML := CategoryMetrics.Items[Stratacode];
722     for M in ML do begin
723       if not (M.MetricType=mtNetBenefits) then continue;
724       SplitUpName(M.FName,Group,VarName,POstfix, FileTag, Year);
725       if FileTag <> aFileTag then continue;
726       if Postfix <> 'Sum' then continue;
727
728       CodeToStrataStr(StrataCode,Source,Size,Ownership,SysType);
729       s := Format('"%s","%s","%s","%s","%s"',
730             [Group,Size,Source,Ownership,SysType]);
731       for i := 1 to 100 do begin
732         s := s + Format(',%2.5e',[M.SQ.Quantile( i/100 - 0.005)]);
733       end;
734       T.WriteLine(s);
735       any := lowercase(Size + Source + Ownership +SysType);
736       if any='anyanyanyany' then begin
737         s2 := '';
738         P:=-1;
739         for i := 1 to 100 do begin
740           v := M.SQ.Quantile( i/100 - 0.005);
741           if (p=-1) and (v>0) then begin
742             p := 1 - i/100 - 0.005;
743             p := p * 100;
744           end;
745           s2 := s2 + Format('%g,%g',[( i/100 - 0.005), v])+NL;
746         end;
747         if M.SQ.Min>0 then P:= 100 else
748         if M.SQ.Max<0 then P:= 0;
749         s:=M.FName+NL;
750         s := s+'Prob>0: '+floattostr(p) + NL +
751              'Mean: '+floattostr(M.SQ.Mean)+ NL +
752              'Min: '+floattostr(M.SQ.Min)+ NL +
753              'Max: '+floattostr(M.SQ.Max)+ NL +
754              'StdDev: '+floattostr(M.SQ.StdDev)+ NL +
755              'Var: '+floattostr(M.SQ.Variance)+ NL +
756              'Weighted N: '+floattostr(M.SQ.TotalWeight)+ NL +
757              NL + 'P,Value'+NL + S2;
758         T2.Write(s);
759       end;
760
761     end;
762   end;
763   T.Free;
764   T2.Free;
765 end;
766
767
```

```
procedure TCategoryMetrics.iSaveUncResultFile(aFilename, aFileTag: string);
begin
  var Comparison: TComparison<TMetricSorter> :=
    function(const Left, Right: TMetricSorter): Integer
    begin
      Result := CompareText(Left.Key,Right.Key);
    end;
  var Sorted := TList<TMetricSorter>.create(TComparer<TMetricSorter>.Construct(Comparison));

  var F0: boolean := aFileTag = '0';
  var FA: string := '_'+FFileNameAdd+'('+aFileTag+').csv';
  var FR: string := ChangeFileExt(aFilename,'');
  //TODO outputs are proliferating - split up....

  var T := TStreamWriter.Create(FR + FA ,false, TEncoding.ASCII,4096*8);
  var EJ := TStreamWriter.Create(FR +'_EJ' + FA ,false, TEncoding.ASCII,4096*8);
  var Sm := TStreamWriter.Create(ChangeFileExt(FR +'_Summary' + FA,'.txt'), false, TEncoding.ASCII,4096*8);

  var CE,CE100,CEAll: TStreamWriter;
  if F0 then begin
    CE := TStreamWriter.Create(FR +'_CostsAdditional' + FA,false, TEncoding.ASCII,4096*8);
    CE100 := TStreamWriter.Create(FR +'_CostsAdditional_p100' + FA,false, TEncoding.ASCII,4096*8);
    CEAll := TStreamWriter.Create(FR +'__CostsAdditional_Draws' + FA,false, TEncoding.ASCII,4096*8);
  end;

  T.Writeline('RegOption,Size,Source,Ownership,Type,Metric,MetricDesc,MetricClass,Statistic,P5,Mean,P95');
  EJ.Writeline('RegOption,Size,Source,Ownership,Type,Metric,MetricDesc,MetricClass,Statistic,P5,Mean,P95');
  if F0 then
    CE.Writeline('RegOption,Size,Source,Ownership,Type,Metric,MetricDesc,MetricClass,Statistic,P5,Mean,P95,StdDev');
  //get the metrics in a sorted order.
  for var StrataCode:integer in CategoryMetrics.Keys do begin
    var Source,Size,Ownership,SysType: string;
    CodeToStrataStr(StrataCode,Source,Size,Ownership,SysType);
    var ML := CategoryMetrics.Items[Stratacode];
    for var M in ML do begin
      //yealy results handled seperately;
      if (M.MetricType=mtCostYearly) or (M.MetricType=mtBenefitYearly) then continue;

      var MT := MetricTypeText[M.MetricType];
      var Group,VarName,Postfix, FileTag, Year: string;
      SplitUpName(M.FName,Group,VarName,POstfix, FileTag, Year);
      var MSort := Default(TMetricSorter);
      MSort.Source := Source;
      MSort.Size := Size;
      Msort.Ownership := Ownership;
      MSort.SysType := Systype;
      MSort.RegOption := group;
      MSort.FileTag := FileTag;
      if Group <> ''  then
        MSort.MetricName := Group+'.'+Varname
      else
       MSort.MetricName := Varname;
      MSort.MetricDesc := M.FDesc;
      MSort.MetricClass := MT;
      MSort.Stat := PostFix;
      MSort.M := M;
      //set sort key here - add procs to record for more
      MSort.DefaultSortKey;
      if (FileTag = aFileTag) or (FileTag = '') then
        Sorted.Add(Msort);
    end;
  end;

  Sorted.Sort;
  for var MSort in Sorted do begin
    var s := MSort.GetCSV;
    s := s + Format(',%2.5e,%2.5e,%2.5e',[MSort.M.SQ.Quantile(0.05),MSort.M.SQ.Mean,MSort.M.SQ.Quantile(0.95)]);
    if MSort.M.MetricType = mtEJOutput then
      EJ.WriteLine(s)
    else if MSort.M.MetricType = mtCostExtended then begin
      if F0 then begin
        CE.WriteLine(MSort.GetCSV + Format(',%2.5e,%2.5e,%2.5e,%2.5e',[MSort.M.SQ.Quantile(0.05),MSort.M.SQ.Mean,MSort.M.SQ.Quantile(0.95),
MSort.M.SQ.StdDev]));
        if CE100.BaseStream.Position=0 then
          CE100.Writeline('RegOption,Size,Source,Ownership,Type,Metric,MetricDesc,MetricClass,Statistic,'+MSort.M.P100CSVListHdr);
```

```
842         CE100.Writeline(MSort.GetCSV+','+MSort.M.P100CSVList);
843         if CEAll.BaseStream.Position=0 then
844           CEAll.Writeline('RegOption,Size,Source,Ownership,Type,Metric,MetricClass,Statistic,'+MSort.M.AllCSVListHdr);
845         CEAll.Writeline(MSort.GetCSVNoDesc+','+MSort.M.AllCSVList);
846       end;
847     end else begin
848       T.WriteLine(s);
849       var any := lowercase(MSort.Size + MSort.Source + MSort.Ownership +MSort.SysType);
850       if ((MSort.Stat = 'Mean') or (MSort.Stat = 'Sum')) and (any='anyanyanyany') then begin
851 (*
852 if (pos('TotalAnnualCosts',MSort.MetricName)>0) then begin
853 csl(MSort.GetCSVNoDesc+','+MSort.M.SaveSummaryToString);
854 end;
855 *)
856
857
858         s := MSort.GetCSVSummary;
859         var fd := SigFigsD(MSort.M.SQ.Quantile(0.05),6);
860         s := S + format(' %15s  %15s  %15s',[_df(SigFigsD(MSort.M.SQ.Quantile(0.05),5)),
861            _df(SigFigsD(MSort.M.SQ.Mean,5)),_df(SigFigsD(MSort.M.SQ.Quantile(0.95),5))]);
862         Sm.WriteLine(s);
863         s := StringOfChar('-', Length(s));
864         Sm.WriteLine(s);
865       end;
866     end;
867   end;
868
869   Sorted.Free;
870   T.Free;
871   EJ.Free;
872   Sm.Free;
873   if F0 then begin
874     CE.Free;
875     CE100.Free;
876     CEAll.Free;
877   end;
878 end;
879
880 procedure TCategoryMetrics.SaveUncResultFile(aFilename: string; aFirstTagOnly: boolean);
881 begin
882   var yFileName: string;
883   for var i := 0 to FFileTags.Count-1 do begin
884     iSaveUncResultFile(aFilename, FFileTags[i]);
885     yFileName := ChangeFileExt(aFilename,'')+'_NetBenefitDist';
886     iSaveNetBenefitResultFile(yFilename, FFileTags[i]);
887   end;
888
889   yFileName := ChangeFileExt(aFilename,'')+'_Yearly';
890   //determine if we only want to output first tags (0 disount rate);
891   var mix := FFileTags.Count-1;
892   if aFirstTagOnly then  mix:=0;
893   for var i := 0 to mix do begin
894     ySaveUncResultFile(yFilename, FFileTags[i]);
895   end;
896 end;
897
898 { TMetricSorter }
899
900 procedure TMetricSorter.DefaultSortKey;
901 begin
902 //  Key:=RegOption+Size+Source+Ownership+SysType+MetricClass+MetricDesc+Stat;
903   Key:=RegOption+Size+Source+Ownership+SysType+MetricClass+MetricName+Stat;
904 end;
905
906 procedure TMetricSorter.YearlySortKey;
907 begin
908 //  Key:=RegOption+Size+Source+Ownership+SysType+MetricClass+MetricDesc+Stat;
909   Key:=RegOption+Size+Source+Ownership+SysType+MetricClass+MetricName+Stat;
910 end;
911
912
913 function TMetricSorter.GetCSV: string;
914 begin
915   Result := Format('"%s","%s","%s","%s","%s","%s","%s","%s","%s"',
916             [RegOption,Size,Source,Ownership,SysType,MetricName,MetricDesc,MetricClass,Stat]);
```

```
end;

function TMetricSorter.GetCSVNoDesc: string;
begin
  Result := Format('"%s","%s","%s","%s","%s","%s","%s","%s"',
            [RegOption,Size,Source,Ownership,SysType,MetricName,MetricClass,Stat]);
end;


function TMetricSorter.GetCSVSummary: string;
begin
  Result := Format('%-50s  %-10.s  ',[MetricName,Stat]);
end;


end.

```

code\SafeWaterModel.pas

```
unit SafeWaterModel;

interface

uses Classes, SysUtils, System.Threading, SafeWaterGlobals, SafewaterPWSRecords, SafeWaterConfig,
     SafeWaterModelData, SafewaterOccurrence, SafeWaterUncertBucket, SafewaterRunner,
     SafeWaterCostModules, SafeWaterUtility, SafewaterMetrics,
     SafeWaterBenefitModules, Generics.Collections, System.SyncObjs,
     SafeWaterWBSCostCurves, SafewaterCohortArray,
     //custom objects for specific analysis
     PFASSerumCalc, SafewaterTFObject, DateUtils;



type
  TGUIProgressProc = procedure (const kind,progress: integer;const step,msg: string) of object;

  TIterQueue =  TThreadedQueue<integer>;

  TSafeWaterModel=class
  private
    fStartTime: TDateTime;
    fLastTime: TDateTime;
    fInitialized : boolean;
    fConfigFileName : string;
    fOnModelDone : TNotifyEvent;
    fStopRequest : boolean;
    fWorkPath, fOutPath: string;
    fConfig: TSafeWaterConfig;
    fNUmThreads, fTasksDone, fTotTasks: integer;
    fFileRoot: string;
    fLogFile: TStreamWriter;
    lckThreadDone: TObject;
    FQueue : TIterQueue;
    FCounter : TCountDownEvent;
    FGUIProc: TGUIProgressProc;
    FNoTime: string; //padding for no time field in log
    FOnStep: string;

    procedure SaveStartupData;

    procedure ModelFinished(Sender: TObject);
    procedure CalcElapsedAndLog(msg: string; TimeStamp: boolean = true; AddInfo: TStringLIst=nil);
    procedure Deinitialize;
    procedure InitCostsBenefitsUncertainty();
    procedure CleanUpRun();
    procedure ThreadDone(Iter: integer);
    procedure OnProgress(const Total, Progress : int64; const msg: string);
    procedure SetStep(s: string);

  public
    //LogFileName : string;
    //SaveDebugLog : boolean;
```

```
54      //UseFullLog : boolean;
55
56      Data: TSWModelData;
57      DummyProb : double;
58
59      constructor create;
60      destructor Destroy; override;
61
62      function Initialize(aOutName : string; aOptionsFile: string) : boolean;
63      procedure Run;
64      procedure RunOne;
65      procedure StopModel;
66      procedure SaveOutput(RunOK: boolean);
67
68      property OnModelDone : TNotifyEvent read FOnModelDone write FOnModelDone;
69      property GUIProc: TGUIProgressProc read FGUIProc write FGUIProc;
70    end;
71
72    TIterThread = class(TThread)
73    private
74      FM: TSafeWaterModel;
75      FQueue : TIterQueue;
76      FSignal : TCountDownEvent;
77      fReportProgress: TProgressProc;
78      FStop: boolean;
79    protected
80      procedure Execute; override;
81    public
82      constructor Create(aSafeWaterModel : TSafeWaterModel; aQueue: TIterQueue; aSignal : TCountdownEvent; ProgressProc : TProgressProc);
83      procedure Stop;
84      property ReportProgress: TProgressProc read fReportProgress;
85    end;
86
87
88  implementation
89
90  { TSafeWaterModel }
91
92  procedure TSafeWaterModel.CalcElapsedAndLog(msg: string; TimeStamp: boolean = true; AddInfo: TstringLIst=nil);
93  begin
94    var step := MillisecondsBetween(Now(), FLastTime);
95    var total := SecondsBetween(Now(), FStartTime);
96    var sstep, stotal: string;
97    var secstep := step div 1000;
98    stotal := Format('%2.2d:%2.2d:%2.2d',[total div 3600,(total div 60) mod 60,total mod 60]);
99    sstep := Format('%2.2d:%2.2d:%2.2d.%3.3d',[secstep div 3600,(secstep div 60) mod 60,secstep mod 60, step mod 1000]);
100   var s:string;
101   if TimeStamp then
102     s := Format('%s (%s): %s',[stotal, sstep,msg])
103   else
104     s := FNoTime + msg;
105   fLogFile.WriteLine(s);
106   if ImInGui then
107     FGUIProc(0,0,'',S);
108   if Assigned(Addinfo) then begin
109     var ss: string;
110     for s in AddInfo do begin
111       ss := FNoTime + s;
112       fLogFile.WriteLine(ss);
113       if ImInGui then
114         FGUIProc(0,0,'',SS);
115     end;
116   end;
117   FLastTime := Now();
118 end;
119
120 procedure TSafeWaterModel.CleanUpRun;
121 begin
122   if Data.Cfg.StartAt=0 then
123     DeleteFiles(FFileRoot+'*.rawi');
124 end;
125
126 constructor TSafeWaterModel.create;
127 begin
```

```
  fInitialized := False;
  fConfig := TSafeWaterConfig.Create;
  DummyProb := 1;
  lckThreadDone := TObject.Create;
  FNoTime := StringOfChar(' ',25);
end;

procedure TSafeWaterModel.Deinitialize;
begin
  Data.PWS.Free;
  Data.Occ.Free;
  Data.Unc.Free;
  Data.TechForecast.Free;
  Data.WBS.Free;
  Data.ErrLog.Free;
  Data.DebugLog.Free;
  Data.EJProps.Free;
  if assigned(Data.SerumCalc) then
    Data.SerumCalc.Free;
  CalcElapsedAndLog('Done');
  CalcElapsedAndLog('Results saved at:' + fOutPath, false);
  fLogFile.Free;
  fInitialized := False;
end;

destructor TSafeWaterModel.destroy;
begin
  if fInitialized then DeInitialize();
  fConfig.Free;
  lckThreadDone.Free;
  inherited;
end;

procedure TSafeWaterModel.InitCostsBenefitsUncertainty;
begin
  //read Tech forecast file
  Data.TechForecast := TAllTechnologiesObj.create;
  Data.TechForecast.LoadTech(Data.Cfg.TechForecastFile);
  if not Data.TechForecast.Active then
    raise exception.Create('Problem reading TechForecast:' +Data.Cfg.TechForecastFile);

  //in this version of safewater, we are handling uncertainty processsing differently...
  //so we make one overall master uncertainty object that gets copied and set in each
  //iteration runner.
  var G := TGlobalUncertainty.Create;
  G.SetupUncertainty(Data.Unc);

  var costs := TSafeWaterCostModules.create(@Data,G,-1);
  costs.SetActive(fConfig.CostOptions);
  if not costs.InitOK then
    if costs.Count =0 then
      raise exception.Create('No cost modules to run or error during initialization');
  costs.SetupUncertainty(Data.Unc);

  var benefits := TSafeWaterBenefitModules.create(@Data,G,-1,'none');
  benefits.SetActive(fConfig.BenefitOptions);
  if benefits.Count=0 then
    Data.ErrLog.L('Warning: No benefit modules to run');
  if not benefits.InitOK then
      raise exception.Create('Error during benefit module initialization');
  benefits.SetupUncertainty(Data.Unc);

  Data.Unc.SetAllToActiveState(fConfig.Uncertainty);
  Data.Unc.GenerateInputStream(true);
  if Data.Cfg.OverrideUnc<>'' then
    Data.Unc.SetInvarTrialsFromFile(Data.Cfg.OverrideUnc);

  costs.Free;
  benefits.Free;
  G.Free;
  CalcElapsedAndLog('Cost/Benefits/Uncertainty Setup');
end;

function TSafeWaterModel.Initialize(aOutName, aOptionsFile: string): boolean;
begin
```

```
203   if fInitialized then DeInitialize();
204   fConfig.LoadFromFile(aOptionsFile);
205   fConfig.ExpandFileNames();
206   fConfigFileName := aOptionsFile;
207   Data.Cfg := fConfig;
208
209   Data.ErrLog := TSimpleLog.Create;
210   Data.DebugLog := TMultiLog.Create;
211   //setup support data tables..
212   Data.Defaults(fConfig.DiscountRates);
213
214   fOutPath := OutPath + aOutName + '\';
215   Data.OutPath := fOutPath;
216   ForceDirectories(fOutPath);
217
218   {$IFDEF DEBUG}
219    //Data.DumpPWSDiscRates;
220    //outputing pre calculated discount/annualization vectors...
221    var l0,l1,l2,l3,l4,l5,l6: string;
222    for var drix:=0 to high(Data.Inp.Rates) do begin
223      l0:='value,dr';
224      var tg := Data.Inp.Rates[drix].ToString;
225      l1:='PreCalcDiscRate,'+tg; l2:='PreCalcAnnRate,'+tg; l3:='PreCalcDiscRateSum,'+tg;
226      l4:='PWSPreCalcDiscRate,'+tg; l5:='PWSPreCalcAnnRate,'+tg; l6:='PWSPreCalcDiscRateSum,'+tg;
227      for var i:=low(Data.Inp.PreCalcDiscRate[drix]) to high(Data.Inp.PreCalcDiscRate[drix]) do begin
228        var cy := i;
229        l0:=l0+',y'+cy.ToString;
230        l1:=l1+','+Data.Inp.PreCalcDiscRate[drix,i].ToString;
231        l2:=l2+','+Data.Inp.PreCalcAnnRate[drix,i].ToString;
232        l3:=l3+','+Data.Inp.PreCalcDiscRateSum[drix,i].ToString;
233        //these must be output by size, etc....
234        //l4:=l4+','+Data.Inp.PWSPreCalcDiscRate[drix,i].ToString;
235        //l5:=l5+','+Data.Inp.PWSPreCalcAnnRate[drix,i].ToString;
236        //l6:=l6+','+Data.Inp.PWSPreCalcDiscRateSum[drix,i].ToString;
237        //PWSPreCalcDiscRate[sw,sz], PWSPreCalcAnnRate[sw,sz], PWSPreCalcDiscRateSum[sw,sz]
238      end;
239      if DRIX=0 then
240        Data.DebugLog.L('discrates',l0);
241      Data.DebugLog.L('discrates',l1);
242      Data.DebugLog.L('discrates',l2);
243      Data.DebugLog.L('discrates',l3);
244    end;
245
246   {$ENDIF}
247
248   fLogFile := TStreamWriter.Create(fOUtPath+'run.log', true, TEncoding.ASCII,4096);
249   fLogFile.WriteLine('');
250   FStartTime := Now();
251   FLastTime := FStartTime;
252   CalcElapsedAndLog('Run Started');
253
254   if Data.Cfg.Workers<>1 then begin
255     if Data.Cfg.DebugOut then Data.ErrLog.L('Debug log turned off - Workers<>1');
256     Data.Cfg.DebugOut := false;
257   end;
258
259
260   //read cohort general data
261   //read PWS files....
262   SetStep('Reading cohort data/cost curves');
263   var CPWS := TCohortDataByIDs.create();
264   var CFIPS := TCohortDataByIDs.create();
265   var CNational: TCohortData;
266   CNational.LoadFromFile(Data.Cfg.NationalSubpopFile);
267   CFIPS.LoadFromFile(Data.Cfg.CountySubpopfile, True);
268   CPWS.LoadFromFile(Data.Cfg.PWSSubpopFile, True);
269   Data.Inp.BodyWeights.LoadFromFile(Data.Cfg.BWFile);
270   Data.Inp.DWConsumption.LoadFromFile(Data.Cfg.ConsumptionFile);
271   Data.EJProps := TEJProportionsByIDs.Create;
272   Data.EJProps.LoadFromFile(Data.Cfg.EJProportionsFile);
273
274
275   //NOTE: reading static locations here.  When generalized can be put in config
276   Data.WBS := TWBSCostCurves.create(Data.ErrLog);
```

```
277   if Data.Cfg.PFASCostHW then
278     Data.WBS.ReadFiles([AppPath+'data\WBSCostCurves\non_hw.csv',
279       AppPath+'data\WBSCostCurves\ix_hw.csv',
280       AppPath+'data\WBSCostCurves\gac_hw.csv',
281       AppPath+'data\WBSCostCurves\ro_hw.csv'])
282   else
283     Data.WBS.ReadFiles([AppPath+'data\WBSCostCurves\non.csv',
284       AppPath+'data\WBSCostCurves\ix.csv',
285       AppPath+'data\WBSCostCurves\gac.csv',
286       AppPath+'data\WBSCostCurves\ro.csv']);
287   CalcElapsedAndLog('Cohort data/cost curves read',true, Data.WBS.Info);
288
289   //load contam data
290   SetStep('Load Occurrence Data');
291   Data.Occ := TOccurrenceData.Create;
292   if ImInGui then Data.Occ.ReportProgress := OnProgress;
293   Data.Occ.LoadFromFile(fConfig.OccFile);
294   CalcElapsedAndLog('Occurrence Data Loaded', true, Data.Occ.Info);
295   var ok: boolean;
296   var T := Data.Occ.VerifyIterations(ok);
297   T.SaveToFile(fOutPath+'Occ_ErrLog.txt');
298   T.Free;
299   if not ok then
300     raise Exception.Create('Problem with Occurrence file - see '+fOutPath+'Occ_ErrLog.txt');
301
302   //read PWS files....
303   SetStep('Load PWS Records');
304   Data.PWS := TPWSRecords.Create;
305   if ImInGui then Data.PWS.ReportProgress := OnProgress;
306   var i := 0;
307   for var s in fConfig.PWSFiles do begin
308     Data.PWS.LoadFromFile(s, Data.Occ.PWSLookup, CFIPS, CPWS, CNational,Data.Cfg.SingleSystemCat,Data.Cfg.SmallLargeCat,i=0);
309     inc(i);
310   end;
311   Data.PWS.CalculateWeights;
312   CalcElapsedAndLog('PWS Records Loaded',true, Data.PWS.Info);
313   {$ifdef DEBUG}
314     Data.PWS.debugOutRandomList(fOutPath+'PWSRandom.csv');
315   {$endif}
316
317   //Subpop files no longer needed so free..
318   FreeAndNil(CFIPS);
319   FreeAndNil(CPWS);
320
321   if Data.Cfg.IncludePBPK then begin
322     Data.SerumCalc := TPFASSerum.Create(Data.Occ.ContamList, Data.Cfg,
323                       @Data.Inp.BodyWeights.V, @Data.Inp.DWConsumption.V);
324     if ImInGui then Data.SerumCalc.ReportProgress := OnProgress;
325   end;
326
327   Data.Unc := TUncertaintyStudy.Create();
328   Data.Unc.TotalTrials :=  Data.Occ.Iterations;
329
330   //setup Benefits and Costs to get uncertainty stream for Runners.
331   SetStep('Initialize Cost/Benefit/Uncertainty');
332   InitCostsBenefitsUncertainty;
333
334   if (fConfig.Workers>0) and (fConfig.Workers<73) then begin
335     fNumThreads := fConfig.Workers;
336   end else begin
337     fNumThreads := TThread.ProcessorCount * fConfig.CPUMult;
338     if (ImInGui) and (fNumThreads>2) then fNumThreads := fNumThreads - 1;
339   end;
340   fInitialized := True;
341 end;
342
343 procedure TSafeWaterModel.ModelFinished(Sender: TObject);
344 begin
345   if Assigned(FOnModelDone) then FOnModelDone(Self);
346 end;
347
348 procedure TSafeWaterModel.OnProgress(const Total, Progress: int64; const msg: string);
349 begin
350
351   if not ImInGui then exit;
```

```
352   TMonitor.Enter(lckThreadDone);
353   try
354     var p := 0;
355     if (Total>0) and (Progress>0) then
356       p := round((progress/total)*100)
357     else begin
358       inc(FTasksDone);
359       if FTotTasks>0 then
360         p := round((FTasksDone/FTotTasks)*100)
361     end;
362     FGuiProc(1, p, FOnStep, msg);
363   finally
364     TMonitor.Exit(lckThreadDone);
365   end;
366 end;
367
368 procedure TSafeWaterModel.Run;
369 begin
370   FStopRequest := false;
371   //save some qa data...
372   SetStep('Save Startup Data');
373   SaveStartupData;
374   CalcElapsedAndLog('Startup Data Saved');
375
376   //do serum precalcs for all threads... NOTE: this is for "mean" run
377   if assigned(Data.SerumCalc) then begin
378     SetStep('Calculating Unit/MCL doses');
379     FTasksDone:=0;
380     Data.SerumCalc.RunUnitDoses;
381     CalcElapsedAndLog('Unit Doses calculated',true,Data.SerumCalc.GetAllInfo);
382   end;
383
384   if Data.Cfg.StartAt>0 then
385     FFileRoot := WorkPath + Data.Cfg.OverrideGUID+'-'
386   else
387     FFileRoot := WorkPath + TGUID.NewGuid.ToString()+'-';
388
389   fQueue := TIterQueue.create(Data.Occ.Iterations);
390   fCounter := TCountDownEvent.Create(fNumThreads);
391   var fIterThreads := TList<TIterThread>.Create;
392   for var i := 0 to fNumThreads-1 do
393     fIterThreads.Add(TIterThread.Create(self, fQueue, fCounter, OnProgress));
394
395   FTasksDone:=0;
396   FTotTasks := Data.Occ.Iterations;
397   SetStep('Running iterations');
398   CalcElapsedAndLog('Started ' + Data.Occ.Iterations.ToString + ' tasks on '+fNumThreads.ToString+' cores');
399
400   for var iter := 0 to Data.Occ.Iterations - 1 do begin
401     fQueue.PushItem(iter);
402     ProcMsg;
403   end;
404
405   for var i := 1 to fNumThreads do
406     fQueue.PushItem(-1);
407
408   var RunOK := true;
409   try
410     fCounter.WaitFor;
411   except
412     on E: EAggregateException do begin
413       for var X in E do begin
414         CSL('Grand Exception:'+X.Message);
415         Data.Errlog.L('Grand Exception:'+X.Message);
416         RunOK:=False;
417       end;
418     end;
419   end;
420
421   fQueue.Free;
422   fCounter.Free;
423   fIterThreads.Free;
424   CalcElapsedAndLog('Finished tasks');
425   Data.ErrLog.Save(fOutPath+'run_errors.txt');
```

```
426   SetStep('Saving output');
427   SaveOutput(RunOK);
428   CalcElapsedAndLog('Output saved');
429   CleanUpRun;
430 end;
431
432 procedure TSafeWaterModel.RunOne;
433 var S: TSafeWaterRunnerSettings;
434    I : TSafeWaterRunner;
435 begin
436   FFileRoot := WorkPath + 'TEST_';
437   S.FId := 1;
438   S.Data := @Data;
439   S.FFileRoot := FFileRoot;
440   I := TSafeWaterRunner.create(S);
441   I.Go;
442   I.Free;
443 end;
444
445 procedure TSafeWaterModel.SaveOutput(RunOK: boolean);
446 begin
447   if RunOk then begin
448     var C := TCategoryMetrics.create('DR');
449     CalcElapsedAndLog('Collecting ' +  Data.Occ.Iterations.ToString+' iterations');
450     for var i := 0 to Data.Occ.Iterations-1 do begin
451        var iname := FFileRoot+'-'+inttostr(i)+'.rawi';
452        if not FileExists(iname) then begin
453           Data.ErrLog.LD('file not found:',FFileRoot+'-'+inttostr(i)+'.rawi');
454           continue;
455        end;
456        if ImInGui then
457          FGuiProc(1, round((i/Data.Occ.Iterations)*100), FOnStep, i.ToString+' of '+Data.Occ.Iterations.ToString);
458        C.LoadIterFromFile(iname, i=0, i, Data.Occ.Iterations);
459        if i=0 then begin
460          CalcElapsedAndLog('Metric Count:'+inttostr(C.Count), true );
461          CalcElapsedAndLog('Tracked Metric Count:'+inttostr(C.TrackedCount), true );
462        end;
463     end;
464     CalcElapsedAndLog('Finished Collecting');
465     C.SaveUncResultFile(fOutPath+'Results', Data.Cfg.ZeroDiscYearly);
466     if C.Info.Count>0 then
467      C.Info.SaveToFile(fOutPath+'NANInf.txt');
468     C.Free;
469   end;
470   Data.ErrLog.Save(fOutPath+'run_errors.txt');
471   if Data.Cfg.DebugOut then Data.DebugLog.Save(fOutPath);
472 end;
473
474 procedure TSafeWaterModel.SaveStartupData;
475 begin
476   fConfig.SaveToFile(fOUtPath+'config.ini');
477   var T: TStringList;
478   T := Data.Occ.GetEPCountList;
479   T.SaveToFile(fOUtPath+'Occ_EntryPointCount.csv');
480   T.Free;
481   T := Data.Occ.OccurrenceReport;
482   T.SaveToFile(fOUtPath+'OccurrenceReport.txt');
483   T.Free;
484
485   Data.PWS.SaveCategoryCounts(fOUtPath+'PWS_CategoryCounts.csv');
486   var TL := TStringList.Create;
487   Data.Unc.ReportInvarsToStrings(TL);
488   TL.SaveToFile(fOUtPath+'UncertainVars.csv');
489   TL.Clear;
490   Data.Unc.ReportInvarTrialsToStrings(TL);
491   TL.SaveToFile(fOUtPath+'UncertainVarsTrials.csv');
492   TL.Free;
493 end;
494
495 procedure TSafeWaterModel.SetStep(s: string);
496 begin
497   if not ImInGui then exit;
498   FOnStep := s;
499   FGuiProc(1, 0, FOnStep, '');
500 end;
```

```
501
502 procedure TSafeWaterModel.StopModel;
503 begin
504   FStopRequest := true;
505 end;
506
507 procedure TSafeWaterModel.ThreadDone(Iter: integer);
508 begin
509   TMonitor.Enter(lckThreadDone);
510   try
511     CSL(Iter.ToString);
512     inc(FTasksDone);
513   finally
514     TMonitor.Exit(lckThreadDone);
515   end;
516 end;
517
518 { TIterThread }
519
520 constructor TIterThread.Create(aSafeWaterModel : TSafeWaterModel; aQueue: TIterQueue; aSignal : TCountdownEvent; ProgressProc : TProgressProc);
521 begin
522   inherited create(false);
523   FM:=aSafeWaterModel;
524   FreeOnTerminate := True;
525   FQueue := aQueue;
526   FSignal := aSignal;
527   FReportProgress := ProgressProc;
528 end;
529
530 procedure TIterThread.Execute;
531 var
532    S: TSafeWaterRunnerSettings;
533    I : TSafeWaterRunner;
534    stp: integer;
535 begin
536   try
537     repeat
538       var L:=FQueue.PopItem;
539       if L < 0 then begin
540        // all done...
541         break;
542       end;
543       S.FId := L;
544       S.Data := @FM.Data;
545       S.FFileRoot := FM.FFileRoot;
546       try
547         stp:=0;
548         I := TSafeWaterRunner.create(S);
549         stp:=1;
550         I.Go;
551         stp:=2;
552         I.Free;
553         if Assigned(FReportProgress) then
554           FReportProgress(0,0,'iteration:'+S.FId.ToString);
555       except
556         on E:exception do begin
557           CSL('In TIterThread.Execute:'+E.Message+'  :'+stp.ToString+'  i:'+L.ToString);
558           S.Data.ErrLog.L('GRAND EXCEPTION: In TIterThread.Execute:'+E.Message+'  :'+stp.ToString+'  i:'+L.ToString);
559           raise;
560         end;
561       end;
562       ProcMsg;
563     until Terminated;
564   finally
565     FSignal.Signal;
566   end;
567 end;
568
569 procedure TIterThread.Stop;
570 begin
571   FStop := True;
572 end;
573
574 end.
```

code\SafeWaterModelData.pas

```
unit SafeWaterModelData;

interface

uses Classes, SafewaterGlobals, SafeWaterConfig, SafeWaterUncertBucket,
     SafeWaterOccurrence, SafewaterPWSRecords, SafewaterCalcDist, SafewaterCohortArray,
     SafewaterWBSCostCurves, SafeWaterUtility,
     //custom objects for specific analysis
     PFASSerumCalc, SafewaterTFObject, Math, SysUtils;

type
  //used for data that varies but is common to pws's and cost,benefit functions
  TGlobalUncertainty = class
    VSLY : array[2024..2150] of double;
    bVSL, CompLevelShift, PAF : double;
    FlowParameters: TFlowParams;
    procedure SetupUncertainty(Unc: TUncertaintyStudy);
    procedure UpdateVSLYearly;
  end;

  //used to inform benefit modules of any cost option information they might need
  //kind of a catch all that can be added to as health effects need info from costs - current example is DBP
  TCostInfo2Benefit=record
    GACTOCBin: integer;
  end;

  TSWModelInputData=record
    BodyWeights,
    DWConsumption : TCohortData;

    HouseholdConsumption : array[swGroundWater..swSurfaceWater,ss0_100..ss1Mp] of double;
    PWSRevenue : array[swGroundWater..swSurfaceWater,ss0_100..ss1Mp] of double;
    PWSCapitalCost : array[swGroundWater..swSurfaceWater,ss0_100..ss1Mp] of double;

    RateCount: integer;
    Rates : array of double; //used to tag outputs - in percent format i.e 3 not 0.03
    PreCalcDiscRate,
    PreCalcDiscRateSum,
    PreCalcAnnRate: TMultiDRArray;

    PWSPreCalcDiscRate,
    PWSPreCalcAnnRate,
    PWSPreCalcDiscRateSum : array[swGroundWater..swSurfaceWater,ss0_100..ss1Mp] of TMultiDRArray;


    procedure Defaults(DiscRates: string);
  end;
  PSWModelInputData = ^TSWModelInputData;

  //Read only data shared among all modules/threads
  TSWModelData=record
    OutPath: string;
    ErrLog: TSimpleLog;
    DebugLog: TMultiLog;
    Cfg: TSafeWaterConfig;
    Inp: TSWModelInputData;
    Unc: TUncertaintyStudy;
    Occ: TOccurrenceData;
    PWS: TPWSRecords;
    EJProps: TEJProportionsByIDs;  // additional EJ data
    WBS: TWBSCostCurves;  //cost curves

    //more specific data that may or may not be present depending on settings
    TechForecast: TAllTechnologiesObj;
    SerumCalc: TPFASSerum;
    procedure Defaults(DiscRates: string);
  end;
  PSWModelData = ^TSWModelData;

```

```
70  implementation
71
72  { TSWModelInputData }
73
74  procedure TSWModelInputData.Defaults(DiscRates: string);
75  begin
76    HouseholdConsumption[0,0]:=119;  HouseholdConsumption[0,1]:=141;
77    HouseholdConsumption[0,2]:=87;   HouseholdConsumption[0,3]:=87;
78    HouseholdConsumption[0,4]:=250;  HouseholdConsumption[0,5]:=106;
79    HouseholdConsumption[0,6]:=134;  HouseholdConsumption[0,7]:=144;
80
81    HouseholdConsumption[1,0]:=74;   HouseholdConsumption[1,1]:=109;
82    HouseholdConsumption[1,2]:=91;   HouseholdConsumption[1,3]:=91;
83    HouseholdConsumption[1,4]:=100;  HouseholdConsumption[1,5]:=100;
84    HouseholdConsumption[1,6]:=112;  HouseholdConsumption[1,7]:=111;
85
86    PWSRevenue[0,0]:=9.17;  PWSRevenue[0,1]:=4.56;
87    PWSRevenue[0,2]:=4.24;  PWSRevenue[0,3]:=3.63;
88    PWSRevenue[0,4]:=3.39;  PWSRevenue[0,5]:=4.01;
89    PWSRevenue[0,6]:=3.41;  PWSRevenue[0,7]:=2.39;
90
91    PWSRevenue[1,0]:=4.19;  PWSRevenue[1,1]:=5.21;
92    PWSRevenue[1,2]:=6.61;  PWSRevenue[1,3]:=5.61;
93    PWSRevenue[1,4]:=5.17;  PWSRevenue[1,5]:=3.2;
94    PWSRevenue[1,6]:=4.03;  PWSRevenue[1,7]:=2.41;
95
96    PWSCapitalCost[0,0]:=0.6;  PWSCapitalCost[0,1]:=3.9;
97    PWSCapitalCost[0,2]:=3.2;  PWSCapitalCost[0,3]:=3.9;
98    PWSCapitalCost[0,4]:=4.1;  PWSCapitalCost[0,5]:=4.2;
99    PWSCapitalCost[0,6]:=3.7;  PWSCapitalCost[0,7]:=4.3;
100
101   PWSCapitalCost[1,0]:=5.4;  PWSCapitalCost[1,1]:=7.3;
102   PWSCapitalCost[1,2]:=6.5;  PWSCapitalCost[1,3]:=5.9;
103   PWSCapitalCost[1,4]:=5.4;  PWSCapitalCost[1,5]:=5.4;
104   PWSCapitalCost[1,6]:=3.6;  PWSCapitalCost[1,7]:=6.2;
105
106   //get discount rates
107   var T := TStringList.create();
108   T.CommaText := DiscRates;
109   var i := T.IndexOf('0');
110   if i>-1 then T.Delete(i);
111   T.Insert(0,'0');
112   setlength(Rates,high(PreCalcDiscRate)+1);
113   for i := 0 to T.Count-1 do begin
114     if i > high(PreCalcDiscRate) then continue;
115     Rates[i] := strtofloat(T[i]);
116     RateCount := i+1;
117   end;
118   T.Free;
119
120   for i := 0 to RateCount-1 do begin
121     InitPrecalcDR2(Rates[i]/100, PreCalcDiscRate[i], PreCalcAnnRate[i], PreCalcDiscRateSum[i]);
122
123     //just repeat PWS rates for each non-zero selected discount rate - can change later if necc.
124     for var sw := swGroundWater to swSurfaceWater do begin
125       for var sz := ss0_100 to ss1Mp do begin
126         if i > 0 then
127           InitPrecalcDR2(PWSCapitalCost[sw,sz]/100,PWSPreCalcDiscRate[sw,sz,i], PWSPreCalcAnnRate[sw,sz,i], PWSPreCalcDiscRateSum[sw,sz,i])
128         else
129           InitPrecalcDR2(0,PWSPreCalcDiscRate[sw,sz,i], PWSPreCalcAnnRate[sw,sz,i], PWSPreCalcDiscRateSum[sw,sz,i]);
130       end;
131     end;
132   end;
133 end;
134
135
136 { TSWModelData }
137
138 procedure TSWModelData.Defaults(DiscRates: string);
139 begin
140   Inp.Defaults(DiscRates);
141   Unc:=nil;
142   Occ:=nil;
143   PWS:=nil;
144   SerumCalc:=nil;
```

```
  WBS:=nil;
  TechForecast:=nil;
  EJProps:=nil;
  OutPath:='';
end;


{ TGobalUncertainty }

procedure TGlobalUncertainty.SetupUncertainty(Unc: TUncertaintyStudy);
begin

  //in 2022 dollars starting in 2024
  Unc.AddInVar(@bVSL, dNone, 11529327, 0, 0, -9E99, 11529327, 0, 0,
    'Globals.VSL', 'Value of a Statistical Life');

  Unc.AddInVar(@PAF, dLogUniform, 0.0394, 0, 0, -9E99, 0.002, 0.179, 0,
   'Globals.CancerPAF', 'Cancer PAF');

  Unc.AddInVar(@FlowParameters.PubGroundA, dNone, 0.08575, 0, 0, -9E99, 0.08575, 0, 0,
      'Flow.PubGroundA', 'Parameter A');
  Unc.AddInVar(@FlowParameters.PubGroundB, dNone, 1.05839, 0, 0, -9E99, 1.05839, 0, 0,
      'Flow.PubGroundB', 'Parameter B');
  Unc.AddInVar(@FlowParameters.D_PubGroundA, dNone, 0.54992, 0, 0, -9E99, 0.54992, 0, 0,
      'Flow.D_PubGroundA', 'D Parameter A');
  Unc.AddInVar(@FlowParameters.D_PubGroundB, dNone, 0.95538, 0, 0, -9E99, 0.95538, 0, 0,
      'Flow.D_PubGroundB', 'D Parameter B');

  Unc.AddInVar(@FlowParameters.PrivGroundA, dNone, 0.06670, 0, 0, -9E99, 0.06670, 0, 0,
      'Flow.PrivGroundA', 'Parameter A');
  Unc.AddInVar(@FlowParameters.PrivGroundB, dNone, 1.06284, 0, 0, -9E99, 1.06284, 0, 0,
      'Flow.PrivGroundB', 'Parameter B');
  Unc.AddInVar(@FlowParameters.D_PrivGroundA, dNone, 0.41682, 0, 0, -9E99, 0.41682, 0, 0,
      'Flow.D_PrivGroundA', 'D Parameter A');
  Unc.AddInVar(@FlowParameters.D_PrivGroundB, dNone, 0.96078, 0, 0, -9E99, 0.96078, 0, 0,
      'Flow.D_PrivGroundB', 'D Parameter B');

  Unc.AddInVar(@FlowParameters.PubSurfA, dNone, 0.14004, 0, 0, -9E99, 0.14004, 0, 0,
      'Flow.PubSurfA', 'Parameter A');
  Unc.AddInVar(@FlowParameters.PubSurfB, dNone, 0.99703, 0, 0, -9E99, 0.99703, 0, 0,
      'Flow.PubSurfB', 'Parameter B');
  Unc.AddInVar(@FlowParameters.D_PubSurfA, dNone, 0.59028, 0, 0, -9E99, 0.59028, 0, 0,
      'Flow.D_PubSurfA', 'D Parameter A');
  Unc.AddInVar(@FlowParameters.D_PubSurfB, dNone, 0.94573, 0, 0, -9E99, 0.94573, 0, 0,
      'Flow.D_PubSurfB', 'D Parameter B');

  Unc.AddInVar(@FlowParameters.PrivSurfA, dNone, 0.09036, 0, 0, -9E99, 0.09036, 0, 0,
      'Flow.PrivSurfA', 'Parameter A');
  Unc.AddInVar(@FlowParameters.PrivSurfB, dNone, 1.03338, 0, 0, -9E99, 1.03338, 0, 0,
      'Flow.PrivSurfB', 'Parameter B');
  Unc.AddInVar(@FlowParameters.D_PrivSurfA, dNone, 0.35674, 0, 0, -9E99, 0.35674, 0, 0,
      'Flow.D_PrivSurfA', 'D Parameter A');
  Unc.AddInVar(@FlowParameters.D_PrivSurfB, dNone, 0.96188, 0, 0, -9E99, 0.96188, 0, 0,
      'Flow.D_PrivSurfB', 'D Parameter B');

  Unc.AddInVar(@CompLevelShift, dUniform, 1, 0, 1, -9E99, 0, 1, 0,
      'CompLevelShift', 'WBS Cost Curve CompLevel Shift');

end;

procedure TGlobalUncertainty.UpdateVSLYearly;
begin
  for var y := low(VSLY) to high(VSLY) do begin
    VSLY[Y]:=bVSL * power(1.00627,Y-2024);
  end;
end;

end.

```

code\SafeWaterOccurrence.pas

```
unit SafeWaterOccurrence;
```

```
2
3  interface
4
5  uses Generics.Collections, SysUtils, Types, Classes, Math, SafeWaterUtility, StrUtils,
6       System.IOUtils, SafewaterGlobals;
7
8  const cftMulti = 10;   //contaminat file types
9        cftUnknown = 99;
10
11       cftMultiFields = 'PWSID,ITERATION,ENTRY_POINT';
12
13 type
14   TContamID = string;
15   TContamPercentiles = array of single;
16   TContamArray = array of single;
17
18   TEPCountLookup = TDictionary<string, integer>;
19   TPWSIDEPLookup = TDictionary<integer,integer> ;
20
21   TOccurrenceData=class
22   private
23
24     fEPCounts: TEPCountLookup;
25     fContamBlob: TMemoryStream;
26     FN: integer;  //number of values (percentiles) per contaminant - 1 for our current files (multi)
27     FIterations: integer;
28     fFileType: integer;
29     fContamLIst: TStringLIst;
30     fContamCount: integer;
31     fReportProgress: TProgressProc;
32     FInfo: TStringList;
33     fIterLen: int64; //memory size
34     fRecLen: int64;//memory size
35     fReverseLookup : TDictionary<integer,string> ;
36     fPWSLookup : TPWSIdLookup;
37     fPWSIDEPLookup : TPWSIDEPLookup;
38
39     procedure ReadMultiContamFile(aFileName: string);
40
41     procedure LoadBinFile(aFilename: string);
42   public
43     constructor Create();
44     destructor Destroy; override;
45     function GetFileType(aFileName: string): integer;
46
47     function GetAll(const PWSID: string; const EPNumber: integer; Iteration: integer; var Conc: TContamArray):boolean; overload;
48     function GetAll(const iPWSID: integer; const EPNumber: integer; Iteration: integer; var Conc: TContamArray):boolean; overload;
49
50     function GetEPCount(const PWSID: string): integer;
51     function GetEPCountList: TStringList;
52     function VerifyIterations(out ok: boolean): TStringList;
53     function OccurrenceReport(): TStringList;
54
55     procedure LoadFromFile(aFilename: string);
56     procedure MakeSubset(aFilename: string; Iters:integer);
57
58     function DataAvailable(Contams: string) : boolean;
59
60     property N: integer read FN write FN;  // number of contam percentiles per iteration
61     property Iterations: integer read FIterations write FIterations;
62     property ContamList: TStringList read FContamList;
63     property Info: TStringList read FInfo;
64     property ReportProgress: TProgressProc read fReportProgress write fReportProgress;
65
66     property PWSLookup: TPWSIDLookup  read fPWSLookup;
67     property PWSIDEPLookup: TPWSIDEPLookup read fPWSIDEPLookup;
68   end;
69
70
71 implementation
72
73 { TOccurrenceData }
74
75 constructor TOccurrenceData.Create;
76 begin
```

```
  fEPCounts:= TEPCountLookup.Create;
  fContamList:=TstringLIst.Create;
  fInfo:=TstringLIst.Create;
  fReverseLookup := TDictionary<integer,string>.create;
  fPWSLookup := TDictionary<string, integer>.create; ;
  fPWSIDEPLookup := TDictionary<integer,integer>.create; ;
  fContamBlob := TMemoryStream.Create;
end;

function TOccurrenceData.DataAvailable(Contams: string): boolean;
begin
  Result := True;
  var L := SplitString(Contams,',');
  for var s in L do begin
    if FContamLIst.IndexOf(uppercase(s))<0 then begin
      Result := False;
      break;
    end;
  end;
end;

destructor TOccurrenceData.Destroy;
begin
  fEPCounts.Free;
  fContamList.Free;
  fInfo.Free;
  fReverseLookup.Free;
  fPWSLookup.Free;
  fPWSIDEPLookup.Free;
  fContamBlob.Free;
  inherited;
end;

//Note: Assumes Conc is properly sized...
function TOccurrenceData.GetAll(const PWSID: string; const EPNumber: integer;
  Iteration: integer; var Conc: TContamArray):boolean;
begin
  Result:=False;
  fillchar(Conc[0],length(Conc), -999);
  var iPWSID, ID, IX : integer;
  if not fPWSLookup.TryGetValue(PWSID,iPWSID) then exit;
  id := PWSEPToCode(iPWSID,EPNumber);
  if not fPWSIDEPLookup.TryGetValue(id, ix) then exit;
  Result:=True;
  var fp := (Iteration)*fIterLen + ix * fRecLen;
  fContamBlob.Position := fp;
  fContamBlob.ReadBuffer(conc[0], fRecLen);
end;

function TOccurrenceData.GetAll(const iPWSID: integer; const EPNumber: integer; Iteration: integer; var Conc: TContamArray):boolean;
begin
  Result:=False;
  fillchar(Conc[0],length(Conc), -999);
  var ID, IX : integer;
  id := PWSEPToCode(iPWSID,EPNumber);
  if not fPWSIDEPLookup.TryGetValue(id, ix) then exit;
  Result:=True;
  var fp := (Iteration)*fIterLen + ix * fRecLen;
  fContamBlob.Position := fp;
  fContamBlob.ReadBuffer(conc[0], fRecLen);
end;

function TOccurrenceData.GetEPCount(const PWSID: string): integer;
begin
  Result:=0;
  if fEPCounts.ContainsKey(PWSID) then
    Result := fEPCounts.Items[PWSID];
end;

function TOccurrenceData.GetEPCountList: TStringList;
begin
   Result := TstringList.Create;
   for var ID in fEPCounts.Keys do begin
     var n := fEPCounts.Items[id];
```

```
151      Result.Add(Format('"%s",%d',[id,n]));
152    end;
153 end;
154
155 function TOccurrenceData.GetFileType(aFileName: string): integer;
156 var F: TStreamreader;
157     ILine, Columns: TStringLIst;
158     s: string;
159     i: integer;
160     ok: boolean;
161 begin
162   Result := cftUnknown;
163   F := TStreamReader.Create(aFileName,TEncoding.ASCII,false,4096);
164   ILine := TstringList.Create;
165   Columns := TstringList.Create;
166   //read header line...
167   s:=F.ReadLine();
168   ILine.CommaText := uppercase(s);
169
170   //check file types...
171   if Result = cftUnknown then begin
172     ok := true;
173     Columns.CommaText := cftMultiFIelds;
174     for i := 0 to Columns.Count-1 do begin
175       if ILine.IndexOf(Columns[i])<0 then begin
176         ok := false;
177         break;
178       end;
179     end;
180     if ok then Result := cftMulti;
181   end;
182
183   F.Free;
184   ILine.Free;
185   Columns.Free;
186 end;
187
188 procedure TOccurrenceData.LoadBinFile(aFilename: string);
189 begin
190 //add 1 to length of datat - ipwsid starts at 1;
191   fEPCounts.Clear;
192   fContamList.Clear;
193   fPWSLookup.Clear;
194   fPWSIDEPLookup.Clear;
195   fReverseLookup.Clear;
196
197   var i,n,cc,id: integer;
198   var T := TBufferedFileStream.Create(aFilename,fmOpenRead + fmShareDenyNone);
199   t.Read(FIterations, sizeof(FIterations));
200   t.Read(FN, sizeof(FN));
201   t.Read(n, sizeof(n));
202   t.Read(cc, sizeof(cc));
203   for i:=0 to cc-1 do begin
204     var s := ReadStreamStr(T);
205     fContamList.Add(s);
206   end;
207   fRecLen := fContamlist.Count * SizeOf(Single);
208   fIterLen := n * fContamlist.Count * FN * SizeOf(Single);
209
210   fContamBlob.SetSize(fIterations*fIterLen);
211   fContamBlob.Position:=0;
212
213  // var size := FIterations*FContamlist.Count*FN * sizeof(single);
214   var iPWSID: integer;
215   var PWSID: string;
216   var hp:int64;
217   //read pws lookup data...
218   var icnt: integer;
219   T.Read(icnt,SizeOf(icnt));
220   for i := 0 to icnt - 1 do begin
221     PWSID := ReadStreamStr(T);
222     T.Read(iPWSID, sizeof(iPWSID));
223     fPWSLookup.Add(PWSID,iPWSID);
224   end;
225
```

```
226   //read pws-ep lookup data...
227   var Ix: integer;
228   for i := 0 to N-1 do begin
229     T.Read(ID,SizeOf(ID));
230     T.Read(Ix,SizeOf(Ix));
231     fPWSIDEPLookup.Add(ID,IX);
232   end;
233   fContamBlob.Clear;
234   hp:=T.Position;
235   fContamBlob.CopyFrom(T,fIterations*fIterLen);
236   T.Position:=hp;
237   (*
238   Ix:=0;
239   for var iter  := 0 to FIterations-1 do begin
240     for i := 0 to N-1 do begin
241       var d := FContamDataArray[ix];
242       for var c  := 0 to High(d) do
243         for var nn  := 0 to High(d[c,iter]) do
244             T.Read(FContamDataArray[i,c,iter,nn], sizeof(single));
245     end;
246
247     if assigned(FReportProgress) then
248       FReportProgress(iter,Fiterations,'Binary read: '+inttostr(iter) + ' of ' +inttostr(FIterations));
249   end;
250   *)
251   T.Free;
252 end;
253
254
255 procedure TOccurrenceData.LoadFromFile(aFilename: string);
256 begin
257   //Force everything to Multi so we don't have to drag around the orginal csv file
258   //fFileType := GetFileType(aFileName);
259   fFileType := cftMulti;
260   fEPCounts.Clear;
261   case fFileType of
262     cftMulti:  begin
263                  ReadMultiContamFile(aFileName);
264                end;
265     cftUnknown:begin
266                  raise exception.Create('Unkown Filetype in TOccurrenceData:'+aFileName);
267                end;
268   end;
269   fContamCount := fContamList.Count;
270 end;
271
272 procedure TOccurrenceData.MakeSubset(aFilename: string; Iters: integer);
273 begin
274   fFileType := GetFileType(aFileName);
275   fEPCounts.Clear;
276   case fFileType of
277     cftMulti:  begin
278                  ReadMultiContamFile(aFileName);
279                end;
280     cftUnknown:begin
281                  raise exception.Create('Unkown Filetype in TOccurrenceData:'+aFileName);
282                end;
283   end;
284 end;
285
286 function TOccurrenceData.OccurrenceReport: TStringList;
287   var r : array of TDoubleDynArray;
288 begin
289   Result:=TStringList.Create;
290
291   var CurConc:TContamArray;
292   setlength(CurConc,ContamList.Count);
293
294   const Bins : TDoubleDynArray = [0,1,2,3,4,5,6,7,8,9,
295                                   10,11,12,13,14,15,16,20,30,40,
296                                   50,100,200,300,400,500,1000,2000,10000,100000];
297   setlength(r,fContamList.Count);
298   for var i:= 0 to fContamList.Count-1 do begin
299     setlength(r[i],high(Bins)+1);
```

```
    for var k:= 0 to high(Bins) do begin
      r[i,k]:=0;
    end;
  end;

  for var j := 0 to fIterations-1 do begin
   for var ix := 0 to fPWSIDEPLookup.Count-1 do begin
     var fp := j*fIterLen + ix * fRecLen;
     fContamBlob.Position := fp;
     fContamBlob.ReadBuffer(CurConc[0], fRecLen);
     for var i := 0 to fContamList.Count-1 do begin
       for var b:= 0 to high(Bins) do begin
         if curconc[i] <= bins[b] then begin
            r[i,b] := r[i,b] + 1;
            break;
         end;
       end;
     end;
   end;
  end;

  var s: string;
  for var c := 0 to 5 do begin
    s:=Format('%-12s %12s %12s %12s %12s %12s',['Contam','<='+inttostr(round(bins[c*5+0])),
    '<='+inttostr(round(bins[c*5+1])),'<='+inttostr(round(bins[c*5+2])),'<='+inttostr(round(bins[c*5+3])),
    '<='+inttostr(round(bins[c*5+4]))]);
    Result.Add(S);
    s := StringOfChar('-', Length(s));
    Result.Add(S);
    for var i := 0 to fContamList.Count-1 do begin
      s:=Format('%-12s %12s %12s %12s %12s %12s',[fContamList[i],inttostr(round(r[i,c*5+0]/FIterations)),
      inttostr(round(r[i,c*5+1]/FIterations)),inttostr(round(r[i,c*5+2]/FIterations)),inttostr(round(r[i,c*5+3]/FIterations)),
      inttostr(round(r[i,c*5+4]/FIterations))]);
      Result.Add(S);
    end;
    s := StringOfChar('-', Length(s));
    Result.Add(S);
  end;
end;

procedure TOccurrenceData.ReadMultiContamFile(aFileName: string);
var F: TStreamreader;
    ILine, Columns,Header: TStringLIst;
    s,pwsid: string;
    id, id2,cnt : integer;
    startcontam, iter, iteridx, epidx: integer;
    Conc: TContamArray;
    FO: array of TFileStream;

begin
  fInfo.Clear;
  fEPCounts.Clear;
  fContamList.Clear;
  fReverseLookup.Clear;
  fPWSLookup.Clear;
  fPWSIDEPLookup.Clear;

  var bfile := ChangeFileExt(aFilename,'.bin');

  if TFile.Exists(bfile) then begin
    if not TFile.Exists(aFileName) then begin
      LoadBinFile(bfile);
      exit;
    end else
    if Fileage(bfile) > FileAge(aFilename) then begin
      LoadBinFile(bfile);
      exit;
    end;
  end;


  var sz := TFile.GetSize(aFileName);
  F := TStreamReader.Create(aFileName,TEncoding.ASCII,false,4096);
  ILine := TstringList.Create;
  Columns := TstringList.Create;
```

```
375   Header := TstringList.Create;
376   Columns.CommaText := cftMultiFields;
377   StartContam := Columns.Count;
378   //read header line...
379   s:=F.ReadLine();
380   Header.CommaText := uppercase(s);
381   if pos('DATA',uppercase(s))>0 then begin
382     Columns.Add('DATA');
383     StartContam := Columns.Count;
384   end;
385   fN := 1; //allow only one vale in these types of files
386   var i := Header.Count - Columns.Count;
387   Setlength(Conc,i);
388   //save contam idxs
389   for i:= StartContam to Header.Count-1 do
390      fContamList.Add(Header[i]);
391
392   iteridx := Header.IndexOf('ITERATION');
393   epidx := Header.IndexOf('ENTRY_POINT');
394
395   i:=0;
396   //read file to get lookups and count of iterations...
397   FIterations:=0;
398   var iPWSid := -1;
399   PWSId:='';
400   id2:=0;
401   var bread:int64 := 0;
402   cnt:=0;
403   while not F.EndOfStream do begin
404     s:=F.ReadLine();
405     bread := bread + length(s);
406     inc(cnt);
407     ILine.CommaText := s;
408     i := strtoint(ILine[iteridx]);
409     if i>FIterations then FIterations:=i;
410     if iLine[0]<>PWSId then begin
411        if not fPWSLookup.ContainsKey(iLine[0]) then begin
412          inc(iPWSID);
413          fReverseLookup.AddOrSetValue(iPWSID,iLine[0]);
414          fPWSLookup.AddOrSetValue(iLine[0], iPWSID);
415        end;
416        PWSId:= iLine[0];
417     end;
418     id :=  PWSEPToCode(iPWSID,strtoint(Iline[epidx]));
419     if not fPWSIDEPLookup.ContainsKey(id)  then begin
420       fPWSIDEPLookup.AddOrSetValue(id,id2);
421       inc(id2);
422     end;
423
424     if cnt mod 20000 = 1 then
425       if assigned(FReportProgress) then
426         FReportProgress(sz,bread,'Pass 1: '+inttostr(bread) + ' of ' +inttostr(sz));
427
428   end;
429   F.Free;
430
431   var CF := TFileStream.Create(bFile,fmCreate);
432
433   CF.Write(FIterations, sizeof(FIterations));
434   CF.Write(FN, sizeof(FN));
435   CF.Write(fPWSIDEPLookup.Count, sizeof(fPWSIDEPLookup.Count));
436   i := FContamList.Count;
437   CF.Write(i, sizeof(i));
438   for cnt:=0 to i-1 do
439     WriteStreamStr(CF,FContamLIst[cnt]);
440
441   i := fPWSLookup.Count;
442   CF.Write(i,SizeOf(i));
443   for pwsid in fPWSLookup.Keys do begin
444     WriteStreamStr(CF,PWSID);
445     iPWSID := fPWSLookup.Items[pwsid];
446     CF.Write(iPWSID,SizeOf(iPWSID));
447   end;
448   //write pws-ep lookup data...
```

```
449   for id in fPWSIDEPLookup.Keys do begin
450     i := fPWSIDEPLookup.Items[id];
451     CF.Write(ID,SizeOf(ID));
452     CF.Write(I,SizeOf(I));
453   end;
454   const HdrSize:int64 = CF.Size;
455
456   //create file and make empty
457   setlength(FO, FIterations);
458   var buf: TArray<byte>;
459   var len := fPWSIDEPLookup.Count * length(Conc) * sizeof(Conc[0]);
460   setlength(buf, len);
461   fillchar(buf[0], len, 0);
462   for i:= 0 to FIterations-1 do begin
463      CF.Write(buf,len);
464   end;
465   CF.Free;
466   CF := TFileStream.Create(bFile,fmOpenReadWrite);
467
468
469   //now read and store data
470   F := TStreamReader.Create(aFileName,TEncoding.ASCII,false,4096);
471   s:=F.ReadLine();
472   PWSId:='';
473   var LineErrors := 0;
474   var fp:int64;
475   bread:=0;
476   cnt:=0;
477   while not F.EndOfStream do begin
478     s:=F.ReadLine();
479     bread := bread + length(s);
480     inc(cnt);
481     ILine.CommaText := s;
482     if ILine.Count<>Header.Count then begin
483      inc(LineErrors);
484      continue;
485     end;
486
487     if cnt mod 20000 = 1 then
488       if assigned(FReportProgress) then
489         FReportProgress(sz,bread,'Pass 2: ' +inttostr(bread) + ' of ' +inttostr(sz));
490
491     if iLine[0]<>PWSId then begin
492        fPWSLookup.TryGetValue(iLine[0], iPWSID);
493        PWSId:= iLine[0];
494     end;
495
496     var epno := strtoint(Iline[epidx]);
497     id :=  PWSEPToCode(iPWSID,epno);
498     fPWSIDEPLookup.TryGetValue(id,id2);
499     iter := strtoint(Iline[iteridx]);
500     for i:= StartContam to Header.Count-1 do begin
501        if not TryStrToFLoat(Iline[i],conc[i - StartContam]) then conc[i - StartContam]:=-999;
502     end;
503     //calculate file position;
504     fp := HdrSize + (iter-1)*len + id2 * length(Conc) * sizeof(Conc[0]);
505     CF.Position := fp;
506     CF.Write(conc[0], length(Conc) * sizeof(Conc[0]));
507   end;
508   CF.Free;
509
510   var ss: string;
511   for i:= StartContam to Header.Count-1 do
512      ss := ss + Header[i]+',';
513   fINfo.Add('Contams: '+ss);
514   fINfo.Add('PWS Count: '+fEPCounts.Count.ToString);
515   fINfo.Add('Line Errors: '+LineErrors.ToString);
516
517   for i:=0 to FIterations-1 do
518    FO[i].Free;
519   F.Free;
520   ILine.Free;
521   Columns.Free;
522   Header.Free;
523
```

```
  LoadBinFile(bfile);
end;



function TOccurrenceData.VerifyIterations(out ok: boolean): TStringList;
var epnum: integer;
    pwsid:integer;
begin
  Result:=TStringList.Create;
  ok := true;
  (*
  for var ID in fContamData.Keys do begin
    CodeToPWSEP(ID,pwsid,epnum);
    var T := fContamData.Items[id];
    var i,j,k: integer;
    for i := 0 to fContamList.Count-1 do
      for j := 0 to fIterations-1 do
        for k:= 0 to fN-1 do begin
          if T[i,j,k] = -999 then begin
            Result.Add(format('Missing:PWSID:%s,EPNo:%d,Iteration:%d,Contam:%s,Idx:%d',[pwsid,epnum,j+1,fContamList[i],k]));
            ok := false;
          end;
        end;
  end;
  *)
end;

end.

```

code\SafeWaterOccurrence2.pas

```
unit SafeWaterOccurrence2;

interface

uses Generics.Collections, SysUtils, Types, Classes, Math, SafeWaterUtility, StrUtils,
     System.IOUtils, SafewaterGlobals;

const cftMulti = 10;   //contaminat file types
      cftUnknown = 99;
      cftMultiFields = 'PWSID,ITERATION,ENTRY_POINT';

type
  TContamID = string;
  TContamArray = array of single;
  TContamTableArray = array of TContamArray;
  TEPCountLookup = TDictionary<string, integer>;
  TPWSIDEPLookup = TDictionary<integer,integer> ;

  TOccurrenceData2=class
  private
    fContamDataArray: TContamTableArray;
    fEPCounts: TEPCountLookup;
    FN: integer;  //number of values (percentiles) per contaminant - 1 for our current files (multi)
    FIterations: integer;
    fFileType: integer;
    fContamLIst: TStringLIst;
    fContamCount: integer;
    fReportProgress: TProgressProc;
    FInfo: TStringList;
    fReverseLookup : TDictionary<integer,string> ;
    fPWSLookup : TPWSIdLookup;
    fPWSIDEPLookup : TPWSIDEPLookup;
    ODataPath:  string;

    procedure ReadMultiContamFile(aFileName: string; SubsetOut: integer=-1);

    procedure LoadBinFile(aFilename: string);
    procedure LoadIteration(aIterNo: integer);

  public
```

```
41      constructor Create();
42      destructor Destroy; override;
43      function GetFileType(aFileName: string): integer;
44
45      procedure GetAll(const PWSID: string; const EPNumber: integer; Iteration: integer; var Conc: TContamArray); overload;
46
47      function GetEPCount(const PWSID: string): integer;
48
49      function GetEPCountList: TStringList;
50      function OccurrenceReport(): TStringList;
51
52      procedure LoadFromFile(aFilename: string);
53      procedure MakeSubset(aFilename: string; Iters:integer);
54
55
56      function DataAvailable(Contams: string) : boolean;
57
58      property N: integer read FN write FN;  // number of contam percentiles per iteration
59      property Iterations: integer read FIterations write FIterations;
60      property ContamList: TStringList read FContamList;
61      property Info: TStringList read FInfo;
62      property ReportProgress: TProgressProc read fReportProgress write fReportProgress;
63
64  //    property ContamDataArray: TContamTableArray read fContamDataArray;
65      property PWSLookup: TPWSIDLookup  read fPWSLookup;
66      property PWSIDEPLookup: TPWSIDEPLookup read fPWSIDEPLookup;
67    end;
68
69
70  implementation
71
72  { TOccurrenceData2 }
73
74  constructor TOccurrenceData2.Create;
75  begin
76    fEPCounts:= TEPCountLookup.Create;
77    fContamList:=TstringLIst.Create;
78    fInfo:=TstringLIst.Create;
79    fReverseLookup := TDictionary<integer,string>.create;
80    fPWSLookup := TDictionary<string, integer>.create; ;
81    fPWSIDEPLookup := TDictionary<integer,integer>.create; ;
82  end;
83
84  function TOccurrenceData2.DataAvailable(Contams: string): boolean;
85  begin
86    Result := True;
87    var L := SplitString(Contams,',');
88    for var s in L do begin
89      if FContamLIst.IndexOf(uppercase(s))<0 then begin
90        Result := False;
91        break;
92      end;
93    end;
94  end;
95
96  destructor TOccurrenceData2.Destroy;
97  begin
98    fEPCounts.Free;
99    fContamList.Free;
100   fInfo.Free;
101   fReverseLookup.Free;
102   fPWSLookup.Free;
103   fPWSIDEPLookup.Free;
104   inherited;
105 end;
106
107
108 //Note: Assumes Conc is properly sized...
109 procedure TOccurrenceData2.GetAll(const PWSID: string; const EPNumber: integer;
110   Iteration: integer; var Conc: TContamArray);
111 begin
112   fillchar(Conc[0],length(Conc), -999);
113   var iPWSID, ID, IX : integer;
114   fPWSLookup.TryGetValue(PWSID,iPWSID);
```

```
115   id := PWSEPToCode(iPWSID,EPNumber);
116   fPWSIDEPLookup.TryGetValue(id, ix);
117   for var c:= 0 to fContamCount - 1 do
118     Conc[c] := FContamDataArray[ix][c];
119 end;
120
121 function TOccurrenceData2.GetEPCount(const PWSID: string): integer;
122 begin
123   Result:=0;
124   if fEPCounts.ContainsKey(PWSID) then
125     Result := fEPCounts.Items[PWSID];
126 end;
127
128 function TOccurrenceData2.GetEPCountList: TStringList;
129 begin
130    Result := TstringList.Create;
131    for var ID in fEPCounts.Keys do begin
132      var n := fEPCounts.Items[id];
133      Result.Add(Format('"%s",%d',[id,n]));
134    end;
135 end;
136
137 function TOccurrenceData2.GetFileType(aFileName: string): integer;
138 var F: TStreamreader;
139     ILine, Columns: TStringLIst;
140     s: string;
141     i: integer;
142     ok: boolean;
143 begin
144   Result := cftUnknown;
145   F := TStreamReader.Create(aFileName,TEncoding.ASCII,false,4096);
146   ILine := TstringList.Create;
147   Columns := TstringList.Create;
148   //read header line...
149   s:=F.ReadLine();
150   ILine.CommaText := uppercase(s);
151
152   //check file types...
153   if Result = cftUnknown then begin
154     ok := true;
155     Columns.CommaText := cftMultiFIelds;
156     for i := 0 to Columns.Count-1 do begin
157       if ILine.IndexOf(Columns[i])<0 then begin
158         ok := false;
159         break;
160       end;
161     end;
162     if ok then Result := cftMulti;
163   end;
164
165   F.Free;
166   ILine.Free;
167   Columns.Free;
168 end;
169
170 procedure TOccurrenceData2.LoadBinFile(aFilename: string);
171 begin
172   var i,ipwsid,id,cnt : integer;
173   var pwsid,tmp: string;
174   var CF := TBufferedFileStream.Create(aFilename,fmOPenRead+fmShareDenyNone);
175   var Conc: TContamArray;
176
177   fContamList.Clear;
178   fReverseLookup.Clear;
179   fPWSLookup.Clear;
180   fPWSIDEPLookup.Clear;
181   CF.Read(FIterations, sizeof(FIterations));
182   CF.Read(i, sizeof(i));
183   setlength(Conc,i);
184   for cnt:=0 to i-1 do
185     FContamLIst.Add(ReadStreamStr(CF));
186
187   //read pws lookup data...
188   CF.Read(i, sizeof(i));
189   for cnt := 0 to i-1 do begin
```

```
190     PWSId := ReadStreamStr(CF);
191     CF.Read(iPWSID, sizeof(iPWSID));
192     fPWSLookup.AddOrSetValue(PWSId,iPWSId);
193   end;
194   //read pws-ep lookup data...
195   while CF.Position<CF.Size do begin
196     CF.Read(ID,sizeof(id));
197     CF.Read(I,sizeof(I));
198     fPWSIDEPLookup.AddOrSetValue(ID,I);
199   end;
200   CF.Free;
201 end;
202
203 procedure TOccurrenceData2.LoadFromFile(aFilename: string);
204 begin
205   fFileType := GetFileType(aFileName);
206   fEPCounts.Clear;
207   case fFileType of
208     cftMulti:  begin
209                  ReadMultiContamFile(aFileName, -1);
210                end;
211     cftUnknown:begin
212                  raise exception.Create('Unkown Filetype in TOccurrenceData2:'+aFileName);
213                end;
214   end;
215   fContamCount := fContamList.Count;
216 end;
217
218 procedure TOccurrenceData2.LoadIteration(aIterNo: integer);
219 begin
220 //filename = ODataPath +iter;
221 end;
222
223 procedure TOccurrenceData2.MakeSubset(aFilename: string; Iters: integer);
224 begin
225   fFileType := GetFileType(aFileName);
226   fEPCounts.Clear;
227   case fFileType of
228     cftMulti:  begin
229                  ReadMultiContamFile(aFileName, Iters);
230                end;
231     cftUnknown:begin
232                  raise exception.Create('Unkown Filetype in TOccurrenceData2:'+aFileName);
233                end;
234   end;
235 end;
236
237
238 function TOccurrenceData2.OccurrenceReport: TStringList;
239   var r : array of TDoubleDynArray;
240 begin
241   Result:=TStringList.Create;
242   const Bins : TDoubleDynArray = [0,1,2,3,4,5,6,7,8,9,10,100,200,300,400,500,600,700,800,900,1000,
243                                   2000,3000,4000,5000,6000,7000,8000,9000,10000];
244   setlength(r,fContamList.Count);
245   for var i:= 0 to fContamList.Count-1 do begin
246     setlength(r[i],high(Bins)+1);
247     for var k:= 0 to high(Bins) do begin
248       r[i,k]:=0;
249     end;
250   end;
251
252   for var ix :=0 to high(FContamDataArray) do begin
253     var i,j,k: integer;
254     for i := 0 to fContamList.Count-1 do
255       for j := 0 to fIterations-1 do begin
256         LoadIteration(j);
257         for k:= 0 to fN-1 do begin
258           for var b:= 0 to high(Bins) do begin
259             if FContamDataArray[ix][i] > bins[b] then begin
260               r[i,b] := r[i,b] + 1;
261             end;
262           end;
263         end;
```

```
      end;
  end;

  var s: string;
  for var c := 0 to 5 do begin
    s:=Format('%-12s %12s %12s %12s %12s %12s',['Contam','>'+inttostr(round(bins[c*5+0])),
    '>'+inttostr(round(bins[c*5+1])),'>'+inttostr(round(bins[c*5+2])),'>'+inttostr(round(bins[c*5+3])),
    '>'+inttostr(round(bins[c*5+4]))]);
    Result.Add(S);
    s := StringOfChar('-', Length(s));
    Result.Add(S);
    for var i := 0 to fContamList.Count-1 do begin
      s:=Format('%-12s %12s %12s %12s %12s %12s',[fContamList[i],inttostr(round(r[i,c*5+0]/FIterations)),
      inttostr(round(r[i,c*5+1]/FIterations)),inttostr(round(r[i,c*5+2]/FIterations)),inttostr(round(r[i,c*5+3]/FIterations)),
      inttostr(round(r[i,c*5+4]/FIterations))]);
      Result.Add(S);
    end;
    s := StringOfChar('-', Length(s));
    Result.Add(S);
  end;
end;

procedure TOccurrenceData2.ReadMultiContamFile(aFileName: string; SubsetOut: integer=-1);
var F: TStreamreader;
    ILine, Columns,Header: TStringLIst;
    s,pwsid: string;
    id, id2,cnt : integer;
    startcontam, iter, epnum, iteridx, epidx: integer;
    Conc: TContamArray;
    FO: array of TFileStream;

begin
  fInfo.Clear;
  fEPCounts.Clear;
  fContamList.Clear;
  fReverseLookup.Clear;
  fPWSLookup.Clear;
  fPWSIDEPLookup.Clear;

  var bfile := ChangeFileExt(aFilename,'.bin');


  if TFile.Exists(bfile) then begin
    if not TFile.Exists(aFileName) then begin
      LoadBinFile(bfile);
      exit;
    end;
    if Fileage(bfile) > FileAge(aFilename) then begin
      LoadBinFile(bfile);
      exit;
    end;
  end;


  var sz := TFile.GetSize(aFileName);
  F := TStreamReader.Create(aFileName,TEncoding.ASCII,false,4096);
  ILine := TstringList.Create;
  Columns := TstringList.Create;
  Header := TstringList.Create;
  Columns.CommaText := cftMultiFields;
  StartContam := Columns.Count;
  //read header line...
  s:=F.ReadLine();
  Header.CommaText := uppercase(s);
  if pos('DATA',uppercase(s))>0 then begin
    Columns.Add('DATA');
    StartContam := Columns.Count;
  end;
  fN := 1; //allow only one vale in these types of files
  var i := Header.Count - Columns.Count;
  Setlength(Conc,i);
  //save contam idxs
  for i:= StartContam to Header.Count-1 do
     fContamList.Add(Header[i]);

```

```
339   iteridx := Header.IndexOf('ITERATION');
340   epidx := Header.IndexOf('ENTRY_POINT');
341
342   i:=0;
343   //read file to get lookups and count of iterations...
344   FIterations:=0;
345   var iPWSid := -1;
346   PWSId:='';
347   id2:=0;
348   while not F.EndOfStream do begin
349     s:=F.ReadLine();
350     ILine.CommaText := s;
351     i := strtoint(ILine[iteridx]);
352     if i>FIterations then FIterations:=i;
353     if iLine[0]<>PWSId then begin
354        if not fPWSLookup.ContainsKey(iLine[0]) then begin
355          inc(iPWSID);
356          fReverseLookup.AddOrSetValue(iPWSID,iLine[0]);
357          fPWSLookup.AddOrSetValue(iLine[0], iPWSID);
358        end;
359        PWSId:= iLine[0];
360     end;
361     id :=  PWSEPToCode(iPWSID,strtoint(Iline[epidx]));
362     if not fPWSIDEPLookup.ContainsKey(id)  then begin
363       fPWSIDEPLookup.AddOrSetValue(id,id2);
364       inc(id2);
365     end;
366   end;
367   F.Free;
368
369   var CF := TFileStream.Create(bFile,fmCreate);
370
371   CF.Write(FIterations, sizeof(FIterations));
372   CF.Write(FN, sizeof(FN));
373   CF.Write(fPWSIDEPLookup.Count, sizeof(fPWSIDEPLookup.Count));
374   i := FContamList.Count;
375   CF.Write(i, sizeof(i));
376   for cnt:=0 to i-1 do
377     WriteStreamStr(CF,FContamLIst[cnt]);
378
379   i := fPWSLookup.Count;
380   CF.Write(i,SizeOf(i));
381   for pwsid in fPWSLookup.Keys do begin
382     WriteStreamStr(CF,PWSID);
383     iPWSID := fPWSLookup.Items[pwsid];
384     CF.Write(iPWSID,SizeOf(iPWSID));
385   end;
386   //write pws-ep lookup data...
387   for id in fPWSIDEPLookup.Keys do begin
388     i := fPWSIDEPLookup.Items[id];
389     CF.Write(ID,SizeOf(ID));
390     CF.Write(I,SizeOf(I));
391   end;
392   const HdrSize:int64 = CF.Size;
393   CF.Free;
394
395
396   //create file and make empty
397   setlength(FO, FIterations);
398   var buf: TArray<byte>;
399   var len := fPWSIDEPLookup.Count * length(Conc) * sizeof(Conc[0]);
400   setlength(buf, len);
401   fillchar(buf[0], len, 0);
402   for i:= 0 to FIterations-1 do begin
403      CF.Write(buf,len);
404   end;
405   CF := TFileStream.Create(bFile,fmOpenReadWrite);
406
407
408   //now read and store data
409   F := TStreamReader.Create(aFileName,TEncoding.ASCII,false,4096);
410   s:=F.ReadLine();
411   PWSId:='';
412   var bread:int64 := 0;
```

```
  var LineErrors := 0;
  var fp:int64;
  while not F.EndOfStream do begin
    s:=F.ReadLine();
    bread := bread + length(s);
    inc(cnt);
    ILine.CommaText := s;
    if ILine.Count<>Header.Count then begin
     inc(LineErrors);
     continue;
    end;

    if cnt mod 20000 = 1 then
      if assigned(FReportProgress) then
        FReportProgress(sz,bread,inttostr(bread) + ' of ' +inttostr(sz));

    if iLine[0]<>PWSId then begin
       fPWSLookup.TryGetValue(iLine[0], iPWSID);
       PWSId:= iLine[0];
    end;

    var epno := strtoint(Iline[epidx]);
    id :=  PWSEPToCode(iPWSID,epno);
    fPWSIDEPLookup.TryGetValue(id,id2);
    iter := strtoint(Iline[iteridx]);
    for i:= StartContam to Header.Count-1 do begin
       if not TryStrToFLoat(Iline[i],conc[i - StartContam]) then conc[i - StartContam]:=-999;
    end;
    //calculate file position;
    fp := HdrSize + (iter-1)*len + id2 * length(Conc) * sizeof(Conc[0]);
    CF.Position := fp;
    CF.Write(conc[0], length(Conc) * sizeof(Conc[0]));
  end;
  CF.Free;

  var ss: string;
  for i:= StartContam to Header.Count-1 do
     ss := ss + Header[i]+',';
  fINfo.Add('Contams: '+ss);
  fINfo.Add('PWS Count: '+fEPCounts.Count.ToString);
  fINfo.Add('Line Errors: '+LineErrors.ToString);

  for i:=0 to FIterations-1 do
   FO[i].Free;
  F.Free;
  ILine.Free;
  Columns.Free;
  Header.Free;

  LoadBinFile(bfile);

  //SaveBinFile(changefileext(aFileName,'.bin'));
  //ProcMsg();
  //LoadBinFile(changefileext(aFileName,'.bin'));
end;



end.

```

code\SafeWaterPWSRecords.pas

```
unit SafeWaterPWSRecords;

interface

uses Classes, SysUtils, Generics.Collections, SafeWaterGlobals, SafewaterUtility, System.IOUtils,
     SafeWaterCohortArray;


const PWSHeader = '"PWS.ID","PWS.Name","EPA.Region","PWS.Type","Owner.Type","Primary.Source","Activity.Status","Population.Served.Count","
Service.Connections.Count","GW.or.SW.Code","sys_size","water_source","Entry.Points","source_info"';
```

```
10        ValidPA = 'AK,AL,AR,AS,AZ,CA,CO,CT,DC,DE,FL,GA,GU,HI,IA,ID,IL,IN,KS,KY,LA,MA,MD,ME,MI,MN,'+
11                  'MO,MP,MS,MT,NC,ND,NE,NH,NJ,NM,NN,NV,NY,OH,OK,OR,PA,PR,RI,SC,SD,TN,TX,UT,VA,VI,VT,WA,WI,WV';
12 type
13   TPWSRecord=record
14     PWSId : string[200];
15     iPWSID: integer; //interal id set from occurence data
16     State : string[2];
17     SystemSize : integer;
18     SystemType : integer;
19     Ownership : integer;
20     SourceWater : integer;
21     EPARegion : Integer;
22     EPCount: integer;
23
24     StatePFAS,       //PFAS specific element.
25     EJInd: integer;  //set to provide ej level output
26     FIPS: string;
27     StateFIPS: integer; //state fips of 0 for items outside 1..72
28
29     Population : double;
30     CohortProp: PSubPopArray;
31     CohortPop: TSubPopArray;
32     TotalRacePop: array[reWhite..reOther] of double;
33
34     AgencyWeight : double; //used to sum state level costs to proper agency
35     CategoryCount: integer;
36     CategoryCode : array of integer;  //category membership of PWS
37
38     //set up some static random numbers for picks over iterations
39      rnCLS : double;  //use this for a common shift for the WBS model
40      //Note: 5000 used as upper bound can be changed.
41      //Note:  This can go away if we run discount rates together and accept the nature of a simulation.
42      rn: array[0..5000] of double;  //static array of random numbers per pws
43
44      function OutData: string;
45      function CopyAdjustPop(UsePop: double): TPWSRecord;
46   end;
47
48   TPWSRecords = class(TObject)
49   private
50     fPWSRecords : TList<TPWSRecord>;
51     fPWSInternalIds: TPWSIdLookup;
52     fPWSCategoryCount : TDictionary<integer,integer>;
53     fReportProgress: TProgressProc;
54     FInfo: TStringList;
55     FRandSeed: integer;
56     OthNational: TSubpopArray;
57
58     function ConvertToSystemSize(pop:double; AltSize: boolean): integer;
59     function getPWSRecord(I: Integer): TPWSRecord;
60     function getPWSRecordCount: Integer;
61     function getCatCount(StrataCode: Integer): integer;
62   public
63
64     constructor Create();
65     destructor Destroy();override;
66     procedure LoadFromFile(FName: string; InternalIDS: TPWSIDLookup; CountyCohorts, PWSCohorts : TCohortDataByIds; NationalProp: TCohortData;
SingleSystemCat: integer; AllCategories, UseAltSize: boolean; ClearRecs:boolean = true);
67     procedure WriteToFile(FName: string);
68     procedure debugOutRandomList(FName: string);
69     procedure CalculateWeights;
70
71     procedure SaveCategoryCounts(aFileName: string; Strings: boolean=true);
72
73     property PWSRecord[I: Integer]: TPWSRecord read getPWSRecord;
74     property CategoryCount[StrataCode: Integer]: integer read getCatCount;
75     property Count: Integer read getPWSRecordCount;
76     property PWSInternalIDs: TPWSIdLookup read fPWSInternalIds;
77     property ReportProgress: TProgressProc read fReportProgress write fReportProgress;
78     property Info: TStringList read FInfo;
79   end;
80
81 implementation
82
83 type
```

```
84  AgencyCount = record
85    totalpop, totalep, cnt : double;
86  end;
87
88  { TPWSRecords }
89
90  procedure TPWSRecords.CalculateWeights;
91  begin
92    var TAC := TDictionary<string,AgencyCount>.create;
93    var AC: AgencyCount;
94    var PWS: TPWSREcord;
95    var VPA := TStringList.Create;
96    VPA.CommaText := ValidPA;
97    VPA.Sorted := True;
98    for var i := 0 to fPWSRecords.Count-1 do begin
99      PWS := fPWSRecords.Items[i];
100     if VPA.IndexOf(PWS.State)<0 then continue;
101     if not TAC.TryGetValue(PWS.State, AC) then AC := Default(AgencyCount);
102     AC.totalep := AC.totalep + PWS.EPCount;
103     AC.totalpop := AC.totalpop + PWS.Population;
104     AC.cnt := AC.cnt + 1;
105     TAC.AddOrSetValue(PWS.State,AC);
106   end;
107
108   //now set Agency Weights...
109   var asum: double := 0.0;
110   for var i := 0 to fPWSRecords.Count-1 do begin
111     PWS := fPWSRecords.Items[i];
112     if TAC.TryGetValue(PWS.State,AC) then begin
113       if AC.cnt > 0  then //choose weight - currently equal weight all
114         PWS.AgencyWeight := 1 / AC.cnt
115       else
116         PWS.AgencyWeight := 0;
117     end else
118       PWS.AgencyWeight := 0;
119     asum := asum + PWS.AgencyWeight;
120     fPWSRecords.Items[i] := PWS;
121   end;
122   TAC.Free;
123   VPA.Free;
124   fInfo.Add('Sum of Agency Weights: '+floattostr(asum));
125 end;
126
127 function TPWSRecords.ConvertToSystemSize(pop: double; AltSize: boolean): integer;
128 begin
129   if AltSize then begin
130     if (pop <= 10000) then
131       Result := ss0_10k
132     else
133       Result := ss10kp;
134   end else begin
135     if (pop <= 100) then
136       Result := ss0_100
137     else if (pop <= 500) then
138       Result := ss101_500
139     else if (pop <= 1000) then
140       Result := ss501_1000
141     else if (pop <= 3300) then
142       Result := ss1001_3300
143     else if (pop <= 10000) then
144       Result := ss3301_10k
145     else if (pop <= 50000) then
146       Result := ss10k_50k
147     else if (pop <= 100000) then
148       Result := ss50k_100k
149     else if (pop <= 1000000) then
150       Result := ss100k_1m
151     else
152       Result := ss1Mp;
153   end;
154 end;
155
156 constructor TPWSRecords.Create;
157 begin
158   fPWSRecords := TList<TPWSRecord>.create;
```

```
  fPWSCategoryCount := TDictionary<integer,integer>.Create;
  fPWSInternalIds := TPWSIdLookup.Create;
  fReportProgress := nil;
  fInfo := TStringLIst.Create;
  FRandSeed := 123;
end;

procedure TPWSRecords.debugOutRandomList(FName: string);
begin
  var T := TStreamWriter.Create(FName);
  var s: string;
  for var ipws := 0 to Count-1 do begin
    var PWS := PWSRecord[ipws];
    s := Format('%s,%.5f',[PWS.PWSId, PWS.rnCLS]);
    for var i := low(PWS.rn) to 10 do
      s := s+Format(',%.5f',[PWS.rn[i]]);
    T.WriteLine(s);
  end;
  T.Free;
end;

destructor TPWSRecords.Destroy;
begin
  fPWSRecords.Free;
  fPWSCategoryCount.Free;
  fPWSInternalIds.Free;
  fInfo.Free;
  inherited;
end;

function TPWSRecords.getCatCount(StrataCode: Integer): integer;
begin
  fPWSCategoryCount.TryGetValue(StrataCode,Result);
end;

function TPWSRecords.getPWSRecord(I: Integer): TPWSRecord;
begin
  Result := fPWSRecords.items[i];
end;

function TPWSRecords.getPWSRecordCount: Integer;
begin
  Result := fPWSRecords.count;
end;

procedure TPWSRecords.LoadFromFile(FName: string; InternalIDS: TPWSIDLookup; CountyCohorts, PWSCohorts : TCohortDataByIds; NationalProp: TCohortData;
SingleSystemCat: integer; AllCategories,UseAltSize: boolean; ClearRecs:boolean = true);
begin
  //TODO - WARNING - Major temp code here!!!
  var TUCMR := TStringList.Create;
  TUCMR.LoadFromFile(datapath+'SDWISUCMR3.txt');
  TUCMR.Sorted:=True;

  //TODO hacking other distribution for tribal pws without FIPS
  DefaultSubpopData(OthNational);
  for var sx:=0 to 1 do begin
    for var r:=0 to 3 do begin
      for var a:=0 to 85 do begin
        OthNational[sx,3,a]:=OthNational[sx,3,a]+NationalProp.V[sx,r,a];
      end;
    end;
  end;

  if ClearRecs then begin
    fPWSRecords.Clear;
    fPWSCategoryCount.Clear;
  end;

  RandSeed := FRandSeed;
  var sz := TFile.GetSize(FName);
  fInfo.Add('Input file: '+FName);
  fInfo.Add('Input file size: '+sz.ToString);
  var T := TStreamReader.Create(FName,TEncoding.ANSI,false,4096);
  var ILine := TStringList.Create;
  ILine.StrictDelimiter := true;
```

```
233   ILine.CommaText := T.ReadLine;
234   const iPWSID = ILine.IndexOf('PWS.ID');
235   const iRegion = ILine.IndexOf('EPA.Region');
236   const iPWSType = ILine.IndexOf('PWS.Type');
237   const iOwnerType = ILine.IndexOf('Owner.Type');
238   const iPop = ILine.IndexOf('Population.Served.Count');
239   const iSource = ILine.IndexOf('GW.or.SW.Code');
240   const iEP = ILine.IndexOf('Entry.Points');
241   const iSP = ILine.IndexOf('STATEPFAS');
242   const iFIPS = ILine.IndexOf('FIPS');
243
244   var CatCnt: integer;
245   if AllCategories then begin
246     if UseAltSize then  //adds additional size breakout small and large systems
247       CatCnt := 24
248     else
249       CatCnt := 16;
250   end else begin
251     CatCnt := 1;
252   end;
253
254   var PWS: TPWSRecord;
255   var tmp: string;
256   var i := 1;
257   var bread:int64 := 0;
258   var NCCnt := 0;
259   var CCCnt := 0;
260   var EJCnt := 0;
261   var PWSCCnt := 0;
262   var MissingFips := TStringlist.Create;
263   MissingFips.Sorted := true;
264   while not T.EndOfStream do begin
265     inc(i);
266     ILine.CommaText := T.ReadLine;
267     bread := bread + length(ILine.Text);
268     PWS := Default(TPWSRecord);
269     PWS.CategoryCount := CatCnt;
270     SetLength(PWS.CategoryCode, CatCnt);
271
272     PWS.PWSId := ILine[iPWSID];
273     //cases where leading zero stripped in csv export = should fix the export
274     //if PWS.PWSId[1] in ['1'..'9'] then
275     //  PWS.PWSId := '0'+ PWS.PWSId;
276     PWS.State := copy(PWS.PWSId,1,2);
277     tmp := ILine[iRegion];
278     Delete(tmp,1,7);
279     PWS.EPARegion := strtoint(tmp);
280     PWS.SystemType := StrToTSystemType(ILine[iPWSType]);
281     PWS.Ownership := StrToTOwnership(ILine[iOwnerType]);
282     PWS.Population := strtofloat(ILine[iPop]);
283     PWS.EPCount := strtoint(ILine[iEP]);
284     PWS.AgencyWeight := 1;
285     PWS.StatePFAS := 0;
286     if iSP>=0 then PWS.StatePFAS := strtoint(ILine[iSP]);
287     PWS.FIPS:='88888';
288     if iFIPS>=0 then PWS.FIPS := ILine[iFIPS];
289     PWS.StateFIPS := FIPSLookup(PWS.State);
290
291 //PWS.State:='WA';
292 //PWS.FIPS:='88888';
293 //PWS.FIPS:='8041';
294 //PWS.FIPS:='00000';
295
296     if assigned(FReportProgress) then
297      if i mod 5000 = 2 then
298        FReportProgress(sz,bread,PWS.PWSId);
299
300     PWS.rnCLS := random;
301     for var rr := low(PWS.rn) to high(PWS.rn) do
302       PWS.rn[rr] := random;
303     PWS.SystemSize := ConvertToSystemSize(PWS.Population, False);
304     PWS.SourceWater := StrToTSourceWater(ILine[iSource]);
305
306     if SingleSystemCat>0 then
307       if PWS.SystemSize<>SingleSystemCat then continue;
```

```
308
309     var bb:=0;
310     var isw,ist,io,iss,ts:integer;
311     if AllCategories then begin
312       for ist := sysCWS to sysAny do begin
313          if (ist<>PWS.SystemType) and (ist<>sysAny) then continue;
314          for isw := swGroundwater to swAny do begin
315            if (isw<>PWS.SourceWater) and (isw<>swAny) then continue;
316            for io := oPublic to oAny do begin
317              if (io<>PWS.Ownership) and (io<>OAny) then continue;
318              for iss := ss0_100 to ssAny do begin
319                if (iss<>PWS.SystemSize) and (iss<>ssAny) then continue;
320                PWS.CategoryCode[bb] := StrataToCode(isw,iss,io,ist);
321                inc(bb);
322                if (UseAltSize) and (iss=PWS.SystemSize) then begin
323                  ts := ConvertToSystemSize(PWS.Population, True);
324                  PWS.CategoryCode[bb] := StrataToCode(isw,ts,io,ist);
325                  inc(bb);
326                end;
327              end;
328            end;
329          end;
330       end;
331     end else begin
332       //just "all" category"
333       PWS.CategoryCode[bb] := StrataToCode(swAny,ssAny,oAny,sysAny);
334     end;
335
336     for var j := Low(PWS.CategoryCode) to High(PWS.CategoryCode) do begin
337       var cnt:integer := 0;
338       fPWSCategoryCount.TryGetValue(PWS.CategoryCode[j],cnt);
339       inc(cnt);
340       fPWSCategoryCount.AddOrSetValue(PWS.CategoryCode[j],cnt);
341     end;
342
343     //Note: Cohort Pops are by Entry Point - assume equal dist of pop among entry points
344     PWS.CohortProp := nil;
345     if Assigned(CountyCohorts) and Assigned(PWSCohorts) then begin
346       PWS.CohortProp := PWSCohorts.GetCohortData(PWS.PWSId);
347       PWS.EJInd := 1;
348       if not Assigned(PWS.CohortProp) then begin
349         PWS.EJInd := 0;
350         PWS.CohortProp := CountyCohorts.GetCohortData(PWS.FIPS);
351         if Assigned(PWS.CohortProp) then
352           inc(CCCnt)
353         else begin
354           //national average
355           inc(NCCnt);
356           if MissingFips.IndexOf(PWS.FIPS)<0 then
357             MissingFips.Add(PWS.FIPS);
358           if PWS.FIPS='99999' then
359             PWS.CohortProp := @OthNational
360           else
361             PWS.CohortProp := @NationalProp.V;  //this changed to .v 7/31/23
362         end;
363       end else begin
364         if not TUCMR.IndexOf(PWS.PWSId)>=0 then
365           PWS.EJInd:=0
366         else
367           inc(EJCnt);
368         inc(PWSCCnt);
369       end;
370     end;
371
372     //TODO Error if PWS.CohortProp = nil
373
374     if Assigned(PWS.CohortProp) then begin
375       var sum: double := 0;
376       for var s := gMale to gFemale do
377         for var r := reWhite to reOther do
378           for var a := 0 to CMaxAge do begin
379             PWS.CohortPop[s,r,a] := (PWS.Population * PWS.CohortProp[s,r,a]) / PWS.EPCount;
380             PWS.TotalRacePop[r] := PWS.TotalRacePop[r] + (PWS.Population * PWS.CohortProp[s,r,a]);
381             sum:=sum+PWS.CohortProp[s,r,a];
```

```
382          end;
383        if (sum<0.98) or (sum>1.02) then
384          fInfo.Add(Format('Bad sum %s,  %g',[PWS.PWSId, sum]));
385      end else
386        DefaultSubPopData(PWS.CohortPop);
387
388      var id: integer;
389      if Assigned(InternalIDS) then begin
390        if not InternalIDS.TryGetValue(PWS.PWSId,id) then id:=-1;
391        PWS.iPWSID := id;
392      end else
393        PWS.iPWSID := -1;
394      fPWSRecords.Add(PWS);
395    end;
396
397    fInfo.Add('Records read: '+inttostr(i-1));
398    fInfo.Add('Current PWS records: '+inttostr(fPWSRecords.Count));
399    fInfo.Add('Count using PWSID Cohorts: '+inttostr(PWSCCnt));
400    fInfo.Add('Count using County Cohorts: '+inttostr(CCCnt));
401    fInfo.Add('Count using National Cohorts: '+inttostr(NCCnt));
402    fInfo.Add('Count - EJ PWS: '+inttostr(EJCnt));
403    if MissingFips.Count>0 then begin
404      fInfo.Add('Missing FIPS List (not found in cohort file):' + inttostr(MissingFips.Count));
405      for var ix := 0 to MissingFips.Count-1 do
406        fInfo.Add(MissingFips[ix]);
407    end;
408    ILine.Free;
409    MissingFips.Free;
410    T.Free;
411    TUCMR.Free;
412  end;
413
414  procedure TPWSRecords.SaveCategoryCounts(aFileName: string; Strings: boolean);
415  begin
416    var OutLines := TStringList.Create;
417    var src,sz,own,typ: integer;
418    var ssrc,ssz,sown,styp: string;
419    for var code in fPWSCategoryCount.Keys do begin
420      var cnt := fPWSCategoryCount.Items[code];
421      if Strings then begin
422        CodeToStrataStr(code,ssrc,ssz,sown,styp);
423        OUtlines.Add(Format('"%s","%s","%s","%s",%d',[ssz,ssrc,sown,styp,cnt]));
424      end else begin
425        CodeToStrata(code,src,sz,own,typ);
426        OUtlines.Add(Format('%d,%d,%d,%d,%d',[sz,src,own,typ,cnt]));
427      end;
428    end;
429    Outlines.Sort;
430    Outlines.Insert(0,'size,source,ownership,type,count');
431    OutLines.SaveToFile(aFileName);
432    OutLines.Free;
433  end;
434
435  procedure TPWSRecords.WriteToFile(FName: string);
436  begin
437
438  end;
439
440  { TPWSRecord }
441
442  function TPWSRecord.CopyAdjustPop(UsePop: double): TPWSRecord;
443  begin
444    Result.PWSId := PWSId;
445    Result.iPWSID := iPWSID;
446    Result.State := State;
447    Result.SystemSize := SystemSize;
448    Result.SystemType := SystemType;
449    Result.Ownership := Ownership;
450    Result.SourceWater := SourceWater;
451    Result.EPARegion := EPARegion;
452  //  Result.EPCount := EPCount;
453    Result.EPCount := 1;
454    Result.StatePFAS := StatePFAS;
455    Result.EJInd := EJInd;
456    Result.FIPS := FIPS;
```

```
  Result.StateFIPS := StateFIPS;
  var a := (UsePop/Population) * EPCount;
  Result.Population := UsePop;
  Result.CohortProp := CohortProp;
  Result.AgencyWeight := AgencyWeight;
  Result.rnCLS := rnCLS;
  var i: integer;
  for i := low(TotalRacePop) to high(TotalRacePop) do
    Result.TotalRacePop[i] := TotalRacePop[i] * a;
  for i := low(CategoryCode) to high(CategoryCode) do
    Result.CategoryCode[i] := CategoryCode[i];
  for i := low(rn) to high(rn) do
    Result.rn[i] := rn[i];
  Result.CohortPop := ScaleSubpopData(CohortPop,a);
end;

function TPWSRecord.OutData: string;
begin
  Result := Format('pwsid:%s, size:%s, type:%s, sw:%s, o:%s, ep:%d, fips:%s, aw:%g',[PWSID,
            SystemSizeToStr(SystemSize),
            SystemTypeToStr(SystemType),
            SourcewaterToStr(SourceWater),
            OwnershipToStr(Ownership),
            EPCount,
            FIPS,
            AgencyWeight]);
end;

end.

```

code\SafeWaterRunner.pas

```
unit SafeWaterRunner;

interface

uses Classes, SysUtils, SafeWaterGlobals, SafewaterPWSRecords, SafeWaterConfig,
     SafeWaterModelData, SafewaterOccurrence, SafeWaterUncertBucket, SafewaterUtility,
     SafeWaterMetrics, SafeWaterCostModules, SafeWaterBenefitModules,
     SafewaterCostBase, System.IOUtils;

type
  TSafeWaterRunnerSettings = record
     Data: PSWModelData;
     FFileRoot: string;
     FId: integer;  //iteration number
  end;

  TSafeWaterRunner = class
  private
    _s: TSafeWaterRunnerSettings;
    Glb: TGlobalUncertainty;
    Unc : TUncertaintyStudy;
    Metrics: TCategoryMetrics;
    benefits: array of TSafeWaterBenefitModules;
    costs: TSafeWaterCostModules;

    MissingEPCount,
    HasEJData,
    TotalPop: double;
    TotalPopBlack,TotalPopWhite,TotalPopHisp,TotalPopOther: double;

    TotalPop_EJ: double;
    TotalPopBlack_EJ,TotalPopWhite_EJ,TotalPopHisp_EJ,TotalPopOther_EJ: double;
    EJPropTotals : array of double;

    NetBenefits, TotalBenefits: array of array of double;
    Skip: boolean;
  public
    constructor create(a:TSafeWaterRunnerSettings);
    destructor Destroy; override;

```

```
41      procedure Go;
42    end;
43
44  implementation
45
46  { TSafeWaterRunner }
47
48  constructor TSafeWaterRunner.create(a: TSafeWaterRunnerSettings);
49  begin
50    _s := a;
51    Skip:=False;
52
53    if _s.Data.Cfg.StartAt>0 then begin
54      if TFile.Exists(_s.FFileRoot +'-'+inttostr(_s.FId)+'.rawi') then begin
55        Skip:=True;
56        exit;
57      end;
58    end;
59
60    Metrics := TCategoryMetrics.create('DR');
61
62    //create cost, benefit modules and copy uncertainty...
63    Unc := TUncertaintyStudy.Create;
64    Unc.TotalTrials := _s.Data.Occ.Iterations;
65
66    //TODO this should be similar to _s.Data.Unc - consistent across iterations
67    //could be just part of _s.Data.Unc maybe
68    Glb := TGlobalUncertainty.create;
69    Glb.SetupUncertainty(Unc);
70
71    costs := TSafeWaterCostModules.create(_s.Data, Glb, _s.FId);
72    costs.SetActive(_s.Data.Cfg.CostOptions);
73    costs.SetupUncertainty(Unc);
74    costs.SetupOutputs(Metrics);
75    //TODO handle if this fails...
76    var ok := costs.InitOk;
77
78    setlength(benefits, costs.ActiveCount);
79    var _cost: TSafeWaterCostBase;
80    var ix:=0;
81    for _cost in costs do begin
82      if not _cost.Active then continue;
83      benefits[ix] := TSafeWaterBenefitModules.create(_s.Data, Glb, _s.FId, _cost.OptionName);
84      benefits[ix].SetActive(_s.Data.Cfg.BenefitOptions);
85      benefits[ix].SetupUncertainty(Unc);
86      //TODO handle if this fails...
87      ok := benefits[ix].InitOk;
88      benefits[ix].SetupOutputs(Metrics);
89      inc(ix);
90    end;
91
92     //get the master arrangment of uncertain parameters...
93    Unc.CloneInputStream(_s.Data.Unc);
94    //set the appropiate iteration
95    Unc.FetchInputTrial(_s.FId + 1);
96    //update VSL (and any other appropriate values)
97    Glb.UpdateVSLYearly();
98
99    //TODO setup cost + benefit metrics...
100   setlength(NetBenefits, _s.Data.Inp.RateCount);
101   setlength(TotalBenefits, _s.Data.Inp.RateCount);
102   for var i := 0 to _s.Data.Inp.RateCount-1 do begin
103     setlength(NetBenefits[i], Costs.Count);
104     setlength(TotalBenefits[i], Costs.Count);
105   end;
106
107   for var drix := 0 to _s.Data.Inp.RateCount-1 do begin
108     var tg := floattostr(_s.Data.Inp.Rates[drix]);
109     ix:=0;
110     for var i := 0 to Costs.Count -  1 do begin
111       if Costs.Items[i].Active then begin
112         Metrics.AddMetric(Costs.Items[i].OptionName+'.NetBenefits',
113                           'Annual Net Benefits', @NetBenefits[drix][ix], mtNetBenefits, mcMeanSum, False, tg);
114         Metrics.AddMetric(Costs.Items[i].OptionName+'.TotalBenefits',
```

```
115                         'Total Annual Benefits', @TotalBenefits[drix][ix], mtBenefitDollars, mcMeanSum, False, tg);
116         inc(ix);
117       end;
118     end;
119   end;
120
121   Metrics.AddMetric('General.TotalEPPop',
122                       'Total Pop All Entry Points', @TotalPop, mtInfo, mcMeanSum, False);
123   Metrics.AddMetric('General.TotalEPPopBlack',
124                       'Total Black Pop All Entry Points', @TotalPopBlack, mtInfo, mcMeanSum, False);
125   Metrics.AddMetric('General.TotalEPPopWhite',
126                       'Total White Pop All Entry Points', @TotalPopWhite, mtInfo, mcMeanSum, False);
127   Metrics.AddMetric('General.TotalEPPopHisp',
128                       'Total Hispanic Pop All Entry Points', @TotalPopHisp, mtInfo, mcMeanSum, False);
129   Metrics.AddMetric('General.TotalEPPopOther',
130                       'Total Other Pop All Entry Points', @TotalPopOther, mtInfo, mcMeanSum, False);
131
132   if _s.Data.Cfg.PFASEJOutput then begin
133     Metrics.AddMetric('General.TotalEPPop_EJ',
134                         'Total Pop EJ Entry Points', @TotalPop_EJ, mtEJOutput, mcMeanSum, False, '', @HasEJData);
135     Metrics.AddMetric('General.TotalEPPopBlack_EJ',
136                         'Total Black Pop EJ Entry Points', @TotalPopBlack_EJ, mtEJOutput, mcMeanSum,False,'',@HasEJData);
137     Metrics.AddMetric('General.TotalEPPopWhite_EJ',
138                         'Total White Pop EJ Entry Points', @TotalPopWhite_EJ, mtEJOutput, mcMeanSum, False,'',@HasEJData);
139     Metrics.AddMetric('General.TotalEPPopHisp_EJ',
140                         'Total Hispanic Pop EJ Entry Points', @TotalPopHisp_EJ, mtEJOutput, mcMeanSum, False,'',@HasEJData);
141     Metrics.AddMetric('General.TotalEPPopOther_EJ',
142                         'Total Other Pop EJ Entry Points', @TotalPopOther_EJ, mtEJOutput, mcMeanSum, False,'',@HasEJData);
143   end;
144
145   Metrics.AddMetric('General.MissingEPConc',
146                       'Entry Points where occurence could not be found', @MissingEPCount, mtInfo, mcSumOnly, False);
147
148   Metrics.AddMetric('General.EJPWSCount',
149                       'PWS with EJ Data availaible', @HasEJData, mtInfo, mcSumOnly, False);
150
151   setlength(EJPropTotals,length(_s.Data.EJProps.Names));
152   if _s.Data.Cfg.PFASEJOutput then begin
153     for var i := 1 to length(_s.Data.EJProps.Names)-1 do begin
154       Metrics.AddMetric('General.'+_s.Data.EJProps.Names[i],
155                         _s.Data.EJProps.Names[i] +' Pop EJ Entry Points', @EJPropTotals[i], mtEJOutput, mcMeanSum, False, '',@HasEJData);
156     end;
157   end;
158 end;
159
160 destructor TSafeWaterRunner.destroy;
161 begin
162   //globals are managed by the iteration - so free
163   if not Skip then begin
164     Glb.Free;
165     Unc.Free;
166     Metrics.Free;
167     for var i := 0 to costs.ActiveCount-1 do
168       benefits[i].Free;
169     costs.Free;
170   end;
171   inherited;
172 end;
173
174 procedure TSafeWaterRunner.Go;
175 var PWS, tPWS : TPWSRecord;
176     ipws,epno: integer;
177 begin
178   const ItNo = _s.FId;
179
180   if Skip then exit;
181
182   var conc, FinalConc : TContamArray;
183   setlength(conc, _s.Data.Occ.ContamList.Count);
184   setlength(FinalConc, _s.Data.Occ.ContamList.Count);
185   const BC = costs.ActiveCount - 1;
186   var i,ix: integer;
187
188   for ipws := 0 to _s.Data.PWS.Count-1 do begin
189 //try
```

```
190     if _s.Data.Cfg.QuickRun then
191       if ipws mod 1000 <> 0 then continue;
192
193     PWS := _s.Data.PWS.PWSRecord[ipws];
194
195 //here
196 //if PWS.PWSId<>'VA6059050' then continue;
197
198     if PWS.iPWSID<0 then begin
199       //TODO log this - means that occurence data doesn't have values for this PWS
200       _S.Data.ErrLog.L(Format('PWS missing occurrence - iter: %d,  pwsid: %s,  EPCount:%d',[_S.FId, PWS.PWSId, PWS.EPCount]));
201       continue;
202     end;
203
204     //zero out costs/benefits before iterating through entry points
205     MissingEPCount := 0;
206     costs.ResetCosts();
207     costs.SetPWSWideEPValues(PWS);
208     for i := 0 to BC do begin
209       benefits[i].ResetBenefits();   //TODO check if this is still needed - set on next line
210       benefits[i].SetPWSWideEPValues(PWS);
211     end;
212
213     var _cost: TSafeWaterCostBase;
214     for epno := 1 to PWS.EPCount do begin
215       //_s.Data.Occ.GetAll(PWS.PWSId, epno, ItNo, conc);
216       if not _s.Data.Occ.GetAll(PWS.iPWSId, epno, ItNo, conc) then begin
217         MissingEPCount := MissingEPCount+1;
218         {$ifdef DEBUG}
219         _s.Data.DebugLog.L('ep_notfound',Format('id:%s,  ItNo:%d, epno:%d,  epcount:%d',[PWS.PWSId,ItNo, epNo, PWS.EPCount]));
220         {$endif}
221         //continue;
222       end;
223
224       ix:=0;
225       var AdjPop: double := 0;
226       for _cost in costs do begin
227         if not _cost.Active then continue;
228         //todo - extend this concept to allow each option to transform the conc independently
229         _cost.CapInputConc(PWS,conc, epno, AdjPop);
230         if ix = 0 then begin
231           if AdjPop> 0 then  //Note: this forces EPCount to 1 - just used for custom EP VLS data at the moment
232             tPWS := PWS.CopyAdjustPop(AdjPop)
233           else
234             tPWS := PWS;
235         end;
236         _cost.AddEntryPointCosts(tPWS,conc,FinalConc);
237         _cost.ForceOutputConc(tPWS,FinalConc,epno);
238         _cost.AddEntryPointCostsPostProcess(tPWS,conc,FinalConc);
239         benefits[ix].AddEntryPointBenefits(tPWS,conc,FinalConc, _cost.CostInfo);
240         benefits[ix].AddEntryPointBenefitsPostProcess(tPWS,conc,FinalConc);
241         inc(ix);
242       end;
243     end;
244
245     costs.CollectPWSWideEPValues(PWS);
246     //join cost options by sum of benefits for net benefits calc
247     var B: double;
248     ix:=0; i:=-1;
249     for _cost in costs do begin
250       inc(i);
251       if not _cost.Active then continue;
252       benefits[ix].CollectPWSWideEPValues(PWS);
253       for var drix := 0 to _s.Data.Inp.RateCount-1 do begin
254         NetBenefits[drix,ix]:=0;
255         TotalBenefits[drix,ix]:=0;
256         B := benefits[ix].GetTotalBenefit(drix);
257         NetBenefits[drix,ix] :=  B - _cost.Output[drix].TotalAnnualCosts;
258         TotalBenefits[drix, ix] := B;
259       end;
260       inc(ix);
261     end;
262     TotalPop := PWS.Population;
263     TotalPopBlack := PWS.TotalRacePop[reBlack];
```

```
    TotalPopWhite := PWS.TotalRacePop[reWhite];
    TotalPopHisp := PWS.TotalRacePop[reHispanic];
    TotalPopOther := PWS.TotalRacePop[reOther];

    HasEJData:=0;
    if PWS.EJInd=1 then begin
      HasEJData:=1;
      var EJP := _s.Data.EJProps.GetAllProportionData(PWS.PWSId);
      TotalPop_EJ := PWS.Population;
      TotalPopBlack_EJ := PWS.TotalRacePop[reBlack];
      TotalPopWhite_EJ := PWS.TotalRacePop[reWhite];
      TotalPopHisp_EJ := PWS.TotalRacePop[reHispanic];
      TotalPopOther_EJ := PWS.TotalRacePop[reOther];
      if Assigned(EJP) then
        for i := 1 to length(EJPropTotals) -1 do EJPropTotals[i] := PWS.Population * EJP.fValues[i]
      else
        for i := 1 to length(EJPropTotals) -1 do EJPropTotals[i] :=0
    end else begin
      //TODO: Gate might not working as intended - zero these out for now.  remove when gate is verified to be working
      TotalPop_EJ := 0;
      TotalPopBlack_EJ := 0;
      TotalPopWhite_EJ := 0;
      TotalPopHisp_EJ := 0;
      TotalPopOther_EJ := 0;
      for i := 1 to length(EJPropTotals) -1 do EJPropTotals[i] :=0
    end;

    //collect metrics at the pws level (not entry point)
    for var c := 0 to PWS.CategoryCount-1 do
      Metrics.CollectPoints(PWS.CategoryCode[c]);
  end;

  //save this complete iteration to a file to be collected at the end for uncertainty
  Metrics.SaveIterToFile(_s.FFileRoot +'-'+inttostr(_s.FId)+'.rawi');

(*
CSL(ItNo.ToString);
var rr := Metrics.SaveIterToStrings;
rr.Free;
CSLS(rr);
*)

end;

end.

```

code\SafeWaterUncertBucket.pas

```
unit SafeWaterUncertBucket;

interface
uses  SysUtils, Classes, Variants,Math, SafeWaterCalcDist,
      System.Contnrs;

const
   MaxAttempts    = 100;    {Max tries to get lower<trial value<upper}
   MinusInfinity  = -9e33;
   Infinity       = 9e33;

type
  TVec = array of double;

  TAllPercentiles = array[0..100] of single;
  TPercentiles = array[0..99] of double;
  PPercentiles = ^TPercentiles;

  TDescriptiveStatistics = record
     N                                      : Longint;
     Mean, StdDev, Skewness, Kurtosis,P1,P99   : single;
     Quantiles                              : TAllPercentiles;
  end;

```

```
25    TDistributed = class(TObject)
26        fValue    : double;
27        DistType  : word;       {Type of distribution (see above)}
28        Value     : pDouble;    {pointer to distributed variable}
29        AllowChange,Active : boolean;
30        PTiles    : ppercentiles;
31        CVM       : double;     {goodness-of-fit stat}
32        //Dists     : TDistFits;  {results of all distribution fit tests}
33        OrigValue,              {User entered default (mean) value}
34        Lower,                  {Lower bound of selection}
35        Upper,                  {Upper bound of selection}
36        Censor,                 {Value at which to censor distribution}
37        Parm1,Parm2,Parm3       {Distribution specific Parameters}
38              : double;
39        CustomPoints : TVec; {Points for custome distribution}
40        VCVMember : boolean; //variable is memebr of a VCV matrix;
41        VCVName   : string; //for UI convenience
42        Points    : array of double;      {Data points - input or output}
43        Stats     : TDescriptiveStatistics;   {Record containing Desc. Statistics}
44        Name      : string[100]; {User supplied variable name}
45        Description : string[200];
46        VCFile    : string[200];  {file containing sets of percentils for VariableCustom}
47
48        constructor Create(nV : pDouble; nType : word; nOrig,nMin,nMax,nC,nP1,nP2,nP3 : double; aName,aDesc : string;
49                           CustomList : TStringList; Size : integer; AllowDistChange : boolean=true; VUFile : string=''; pP : PPercentiles=nil);
50        destructor  Destroy; override;
51        procedure   Toss;
52        procedure   StorePoint(const OnTrial : integer);
53        procedure GenerateStats;
54        procedure MakeDist(NPoints : integer; var OutPoints : TVec);
55
56        procedure LoadInputStreamFromList(AList : TObjectList);
57        procedure LoadFromStream(AStream : TStream);
58        procedure SaveToStream(AStream : TStream);
59    end;
60
61    TVCV = class(TObject)
62      Name : string;
63      Members : TStringList;
64      Vars : array of TDistributed;
65      VCV : array of array of double;
66
67      constructor Create(aName : string; CSVMembers : string; Filename: string = '');
68      destructor  Destroy; override;
69
70      procedure SetVar(varname : string; D : TDistributed);
71      function GetRowCol(const r,c : integer) : double;
72      procedure SetRowCol(const r,c : integer; const v : double);
73      procedure SetRow(const r: integer; const v: array of double);
74      function GetIdx(const aName : string) : integer;
75
76      procedure LoadFromStream(AStream : TStream);
77      procedure SaveToStream(AStream : TStream);
78
79    end;
80
81    TTrial=class(TObject)
82        R : double;
83        constructor Create(nR : double);
84    end;
85
86
87    TUncertaintyStudy=class(Tobject)
88        private
89          FLock: TObject;
90          fTotalTrials : integer;
91          procedure setTotalTrials(aValue : integer);
92          procedure SetupVCV;
93        public
94        VCVs,
95        SimpleVars,           { Collection of other misc variables that dont vary with uncertinty vars}
96        InVars,              {Collection of input variables,distributions and points}
97        OutVars    : TObjectList;  {Collection of output variables,distributions, and points}
98        RandomStream : TObjectList; {single randmon number gnerated for each trial}
99        NumTrials  : integer;
```

```
100       StoreInputs: boolean;
101
102       constructor Create;
103       destructor  Destroy; override;
104       function AddInVar(nV : pDouble; nType : integer; nOrig,nMin,nMax,nC,nP1,nP2,nP3 : double; aName,aDesc : string;  AllowDistChange : boolean=
    true; CustomList : TStringList=nil; VUFile : string=''; pP : PPercentiles=nil) : TDistributed;
105       function AddOutVar(nV : pDouble; aName,aDesc : string) : TDistributed;
106       function AddVCVMatrix(aName : string; fMembers : string; Filename: string='') : TVCV;
107       procedure FindVCVMatrix(aName : string; var VCIdx,MemberIX : integer);
108       procedure ClearAll;
109       procedure ResetValues;
110       procedure NewTrial;
111       procedure StoreInputTrial(const OnTrial : integer);
112       procedure StoreOutputTrial(const OnTrial : integer);
113       procedure StoreTrial(const OnTrial : integer);
114       procedure GenerateInputStream(Latin : boolean);
115       procedure SetAllToActiveState(state: boolean);
116
117       procedure GenerateRandomStream;
118       procedure FetchInputTrial(Trial : integer);
119       function  GetRandomFromStream(Trial : integer) : double;
120       procedure SumSingleOutputTrial(T : TDistributed; Trial : integer);
121       procedure SumOutputTrial(Trial : integer);
122
123       procedure GenerateStats;
124       procedure DumbPrintToStrings(T : TStrings);
125       procedure DumbPrintToCSVStrings(T : TStrings);
126       procedure ResetRun;
127       function FindInVar(Name : string; var ix : integer) : boolean;
128       function FindOutVar(Name : string; var ix : integer) : boolean;
129       function FindInVarPointer(Name : string) : pDouble;
130
131       procedure ReportInvarsToStrings(T : TStrings);
132       procedure ReportInvarTrialsToStrings(T : TStrings);
133       procedure SetInvarTrialsFromFile(aFilename: string);
134       procedure CloneInputStream(Src: TUncertaintyStudy);
135
136       procedure LoadFromStream(AStream : TStream);
137       procedure SaveToStream(AStream : TStream);
138
139
140       property TotalTrials : integer read fTotalTrials write setTotalTrials;
141    end;
142
143
144 function  StrDist(D : word) : string;
145 function  StrDistToCode(N : string) : integer;
146 function  StrParm(D,P : word) : string;
147
148 var CountExceptions : word;
149
150 implementation
151
152 function StrDist(D : word) : string;
153 var s : string;
154 begin
155    case D of
156
157    dNone       : s:='None';         dCustom     : s:='Custom';
158    dNormal     : s:='Normal';       dTriangular : s:='Triangular';
159    dPoisson    : s:='Poisson';      dBinomial   : s:='Binomial';
160    dLogNormal  : s:='LogNormal';    dUniform    : s:='Uniform';
161    dExponential: s:='Exponential';  dGeometric  : s:='Geometric';
162    dWeibull    : s:='Weibull';      dGamma      : s:='Gamma';
163    dLogistic   : s:='Logistic';     dCauchy     : s:='Cauchy';
164    dPareto     : s:='Pareto';       dBeta       : s:='Beta';
165    dHML        : s:='H-M-L';        dVariableCustom: s:='VariableCustom';
166    dLogUniform: s:='LogUniform';
167    end;
168    StrDist:=S;
169 end;
170
171 function  StrDistToCode(N : string) : integer;
172 var S : string;
173 begin
```

```
174   S:=UpperCase(N);
175   if s='CUSTOM' then result:=dCustom else
176   if s='NORMAL' then result:=dNormal else
177   if s='TRIANGULAR' then result:=dTriangular else
178   if s='POISSON' then result:=dPoisson else
179   if s='BINOMIAL' then result:=dBinomial else
180   if s='LOGNORMAL' then result:=dLogNormal else
181   if s='UNIFORM' then result:=dUniform else
182   if s='EXPONENTIAL' then result:=dExponential else
183   if s='GEOMETRIC' then result:=dGeometric else
184   if s='WEIBULL' then result:=dWeibull else
185   if s='GAMMA' then result:=dGamma else
186   if s='LOGISTIC' then result:=dLogistic else
187   if s='CAUCHY' then result:=dCauchy else
188   if s='PARETO' then result:=dPareto else
189   if s='BETA' then result:=dBeta else
190   if s='H-M-L' then result:=dHML else
191   if s='VARIABLECUSTOM' then result:=dVariableCustom else
192   if s='LOGUNIFORM' then result:=dLogUniform else
193      result:=dNone;
194 end;
195
196 function StrParm(D,P : word) : string;
197 var s : string;
198 begin
199    case D of
200    299 : case P of 1 : s:='Point Value'; 2: s:='';3 : s:=''; end;
201    300 : s:='';
202    305 : case P of 1 : s:='Geom. Mn'; 2: s:='STD';3 : s:=''; end;
203    301 : case P of 1 : s:='Mean'; 2 : s:='STD'; 3 : s:=''; end;
204    303,307 : case P of 1 : s:='Rate'; 2 : s:=''; 3 : s:=''; end;
205    302 : case P of 1 : s:='Min'; 2 : s:='Max'; 3 : s:='Most Likely'; end;
206    304 : case P of 1 : s:='Prob'; 2 : s:='Trials'; 3 : s:=''; end;
207    306 : case P of 1 : s:='Min'; 2 : s:='Max'; 3 : s:= ''; end;
208    308 : case P of 1 : s:='Prob'; 2 : s:=''; 3 : s:=''; end;
209    310 : case P of 1 : s:='Location'; 2 : s:='Alpha'; 3 : s:='Lambda'; end;
210    309 : case P of 1 : s:='Scale'; 2 : s:='Shape'; 3 : s:=''; end;
211    311 : case P of 1 : s:='Mean'; 2 : s:='Scale'; 3: s:=''; end;
212    312 : case P of 1 : s:='Alpha'; 2 : s:='Beta'; 3 : s:='Scale'; end;
213    313 : case P of 1 : s:='x0'; 2 : s:='Theta'; 3 : s:=''; end;
214    314 : case P of 1 : s:='Alpha'; 2 : s:='Beta'; 3 : s:=''; end;
215    315 : case P of 1 : s:='High'; 2 : s:='Medium'; 3 : s:='Low'; end;
216    316 : case P of 1 : s:='Mean'; 2 : s:='STD'; 3 : s:='DF'; end;
217    317 : case P of 1 : s:='Min'; 2 : s:='Max'; 3 : s:=''; end;
218    end;
219    StrParm:=S;
220 end;
221
222 function RandComp(Data1, Data2 : pointer) : integer;
223 begin
224   //Radomizer...
225   Result:=-1+Random(3);
226 end;
227
228 procedure WriteStreamStr(Stream : TStream; Str : string);
229 var
230  StrLen : integer;
231 begin
232  StrLen := Length(Str);
233  Stream.WriteBuffer(StrLen, SizeOf(Integer));
234  Stream.WriteBuffer(Pointer(Str)^, StrLen * SizeOf(Char)); //All Delphi versions compatible
235 end;
236
237 function ReadStreamStr(Stream : TStream) : string;
238 var
239   StrLen : integer;
240   TempStr : string;
241 begin
242   TempStr := '';
243   Stream.ReadBuffer(StrLen, SizeOf(Integer));
244   if StrLen > -1 then begin
245     SetLength(TempStr, StrLen);
246     Stream.ReadBuffer(Pointer(TempStr)^, StrLen * SizeOf(Char));
247     result := TempStr;
248   end else Result := '';
```

```
end;


function RandD(dtype:integer;parm1,parm2,parm3:double):double;
var y : double;
begin
   y:=random;
   Result:=icdf(dtype,y,parm1,parm2,parm3);
end;

constructor TTrial.Create(nR : double);
begin
   R:=nR;
end;

constructor TDistributed.Create(nV : pDouble; nType : word; nOrig,nMin,nMax,nC,nP1,nP2,nP3 : double; aName,aDesc : string;
             CustomList : TStringList; Size : integer;  AllowDistChange : boolean=true; VUFile : string=''; pP : PPercentiles=nil);
var i : integer;
begin
   inherited create;
   Active:=True;
   DistType:=nType;
   Parm1:=nP1;       Parm2:=nP2;   Parm3:=nP3;
   OrigValue:=nOrig; Lower:=nMin;  Upper:=nMax;
   Value:=nV;
   AllowChange:=AllowDistChange;
   VCVMember:=False;
   VCVName:='';

   if DistType=dVariableCustom then begin
      VCFile:=VUFile;
      PTiles:=pP;
   end else
   //if both addresses are nil, use the local var
   if (Value=nil) then
      Value:=@fValue;

   Name:=aName;
   Description:=aDesc;
   Censor:=nC;
   if (Lower=Upper) then begin
      Lower:=MinusInfinity;
      Upper:=Infinity;
   end;

   setlength(Points,Size+1);
   for i:=0 to high(Points) do Points[i]:=0;
   Points[0]:=OrigValue;

   //this is set so icdf functions works for no dist;
   if DistType=dNone then
     Parm1:=OrigValue;

   if DistType=dCustom then begin
     (*
     CustomPoints:=TVec.Create;
     if CustomList=nil then begin
       CustomPoints.Size(1);
       CustomPoints[0]:=nOrig;
     end else begin
       if CustomList.Count=0 then begin
         CustomPoints.Size(1);
         CustomPoints[0]:=nOrig;
       end else begin
         CustomPoints.Size(CustomList.Count);
         for i:=0 to CustomList.Count-1 do begin
           CustomPoints[0]:=0;
           try
             CustomPoints[0]:=strtofloat(CustomList[i]);
           except
             //TODO add exception.   Values are being set to zero.
           end;
         end;
       end;
```

```
323      end;
324      CustomPoints.SortAscend();
325      *)
326    end;
327 end;
328
329 procedure TDistributed.MakeDist(NPoints : integer; var OutPoints : TVec);
330 var j : integer;
331     v,pctstart,pctend,pct : double;
332 begin
333   setlength(OutPoints,NPoints);
334   pct:=1/NPoints;
335   //find if distribution is truncated
336   v:=cdf(DistType,Lower,Parm1,Parm2,Parm3);
337   if v>0 then PctStart:=v else PctStart:=0;
338   v:=cdf(DistType,Upper,Parm1,Parm2,Parm3);
339   if v>0 then PctEnd:=v else PctEnd:=1;
340
341   for j:=1 to NPoints do begin
342      if (DistType=dNone) then begin
343        OutPoints[j-1]:=OrigValue;
344      end else begin
345        OutPoints[j-1]:=icdf(DistType,PctStart + (((J*Pct)-Pct/2) * (PctEnd-PctStart)),Parm1,Parm2,Parm3);
346      end;
347   end;
348 end;
349
350 procedure TDistributed.Toss;
351 var T : double;
352     N : integer;
353
354     function GotIt : boolean;
355     var bot,top,randnum: double;
356     begin
357        inc(N);
358
359        if DistType=dCustom then begin
360          (*
361          idx:=Random(CustomPoints.Length+1);
362          T:=CustomPoints.Values[idx];
363          GotIt:=True;
364          *)
365          exit;
366        end;
367
368        if T<Censor then begin
369           T:=Censor;
370           GotIt:=True;
371        end else begin
372           if N>MaxAttempts then begin
373              top:=cdf(disttype,upper,parm1,parm2,parm3);
374              bot:=cdf(disttype,lower,parm1,parm2,parm3);
375              randnum:=bot+((top-bot)*random);
376              T:=icdf(disttype,randnum,parm1,parm2,parm3);
377              GotIt:=True;
378           end else
379              Gotit:=(Lower<T) and (T<Upper);
380        end;
381     end;
382
383 begin
384   N:=0;
385   try
386     if DistType=dVariableCustom then begin
387       //TSafeWaterPercUncFile.GetRandom(VCFile,PTiles^);
388     end else
389       repeat
390          T:=RandD(DistType,Parm1,Parm2,Parm3);
391       until GotIt;
392   except on EMathError do begin
393             CountExceptions:=CountExceptions+1;
394             N:=MaxAttempts+1;
395          end;
396   end;
397
```

```
//  if N<MaxAttempts then move(T,Value^,sizeof(Value^));
  if N<MaxAttempts then Value^:=T
end;

procedure TDistributed.StorePoint( const OnTrial : integer);
begin
   Points[OnTrial]:=Value^;
end;
destructor TDistributed.destroy;
begin
   //if DistType=dCustom then
     //FreeAndNil(CustomPoints);
   inherited destroy;
end;

procedure TDistributed.GenerateStats;
(*
var T : TVec;
    i : integer;
    interval : double;
*)
begin
   (*
   createit(T);
   T.Size(Points.Count);
   for i:=0 to Points.Count-1 do begin
     T.Values[i]:=TTrial(Points.Items[i]).R;
   end;
   T.SortAscend;
   Stats.N:=T.Length;
   Stats.Mean:=T.Mean;
   Stats.StdDev:=T.StdDev(Stats.Mean);
   Stats.Skewness:=T.Skewness(Stats.Mean,Stats.StdDev);
   Stats.Kurtosis:=T.Kurtosis(Stats.Mean,Stats.StdDev);
   Stats.P1:=Percentile(T,0.01);
   Stats.P99:=Percentile(T,0.99);
   interval:=(100/(high(Stats.Quantiles)-low(Stats.Quantiles)+1))/100;
   for i:=low(Stats.Quantiles) to high(Stats.Quantiles) do begin
      Stats.Quantiles[i]:=Percentile(T,i*interval+interval/2);
   end;
   freeit(T);
   *)
end;


procedure TDistributed.LoadFromStream(AStream: TStream);
var i,j : integer;
begin
   AStream.ReadBuffer(Name,SizeOf(Name));
   AStream.ReadBuffer(Description,SizeOf(Description));
   AStream.ReadBuffer(DistType,SizeOf(DistType));
   AStream.ReadBuffer(Parm1,SizeOf(Parm1));
   AStream.ReadBuffer(Parm2,SizeOf(Parm2));
   AStream.ReadBuffer(Parm3,SizeOf(Parm3));
   AStream.ReadBuffer(OrigValue,SizeOf(OrigValue));
   AStream.ReadBuffer(Lower,SizeOf(Lower));
   AStream.ReadBuffer(Upper,SizeOf(Upper));
   AStream.ReadBuffer(Censor,SizeOf(Censor));
   AStream.ReadBuffer(i,SizeOf(i));
   setlength(Points,i);
   for j:=0 to high(Points) do begin
     AStream.ReadBuffer(Points[j],SizeOf(Points[j]));
   end;
end;

procedure TDistributed.SaveToStream(AStream: TStream);
var i : integer;
begin
   AStream.Write(Name,SizeOf(Name));
   AStream.Write(Description,SizeOf(Description));
   AStream.Write(DistType,SizeOf(DistType));
   AStream.Write(Parm1,SizeOf(Parm1));
   AStream.Write(Parm2,SizeOf(Parm2));
   AStream.Write(Parm3,SizeOf(Parm3));
```

```
472     AStream.Write(OrigValue,SizeOf(OrigValue));
473     AStream.Write(Lower,SizeOf(Lower));
474     AStream.Write(Upper,SizeOf(Upper));
475     AStream.Write(Censor,SizeOf(Censor));
476     i:=high(Points)+1;
477     AStream.Write(i,SizeOf(i));
478     for i:=0 to high(Points) do begin
479       AStream.Write(Points[i],SizeOf(Points[i]));
480     end;
481 end;
482
483 procedure TDistributed.LoadInputStreamFromList(AList : TObjectList);
484 begin
485
486 end;
487
488 { TVCV }
489
490 constructor TVCV.Create(aName, CSVMembers: string; Filename: string = '');
491 var i : integer;
492 begin
493   Name:=aName;
494   Members:=TstringList.Create;
495   Members.CommaText:=CSVMembers;
496   setlength(VCV,Members.Count);
497   setlength(Vars,Members.Count);
498   for i:=0 to high(VCV) do
499     setlength(VCV[i],Members.Count);
500 end;
501
502 destructor TVCV.destroy;
503 begin
504   Members.Free;
505   inherited;
506 end;
507
508 function TVCV.GetIdx(const aName: string): integer;
509 begin
510   Result:=Members.IndexOf(aName);
511 end;
512
513 function TVCV.GetRowCol(const r, c: integer): double;
514 begin
515   Result:=VCV[r,c];
516 end;
517
518 procedure TVCV.SetVar(varname : string; D : TDistributed);
519 var i : integer;
520 begin
521   i:=GetIdx(varname);
522   Vars[i]:=D;
523 end;
524
525 procedure TVCV.SetRowCol(const r, c: integer; const v: double);
526 begin
527   VCV[r,c]:=v;
528 end;
529
530 procedure TVCV.SetRow(const r: integer; const v: array of double);
531 var i : integer;
532 begin
533   for i:=0 to high(v) do SetRowCol(r,i,v[i]);
534 end;
535
536
537
538 procedure TVCV.LoadFromStream(AStream : TStream);
539 var i,j : integer;
540 begin
541   Name:=ReadStreamStr(AStream);
542   Members.CommaText:=ReadStreamStr(AStream);
543   setlength(VCV,Members.Count);
544   setlength(Vars,Members.Count);
545   for i:=0 to high(VCV) do
546     setlength(VCV[i],Members.Count);
```

```
547   for i:=0 to high(VCV) do begin
548     for j:=0 to high(VCV[i]) do begin
549       AStream.ReadBuffer(VCV[i,j],sizeof(VCV[i,j]));
550     end;
551   end;
552 end;
553
554 procedure TVCV.SaveToStream(AStream : TStream);
555 var i,j : integer;
556 begin
557   WriteStreamStr(AStream,Name);
558   WriteStreamStr(AStream,Members.CommaText);
559   for i:=0 to high(VCV) do begin
560     for j:=0 to high(VCV[i]) do begin
561       AStream.Write(VCV[i,j],sizeof(VCV[i,j]));
562     end;
563   end;
564 end;
565
566
567 { TUncertaintyStudy }
568
569 constructor TUncertaintyStudy.Create;
570 begin
571   inherited;
572   InVars :=TObjectList.Create(true);
573   OutVars:=TObjectList.Create(true);
574   SimpleVars:=TObjectList.Create(true);
575   RandomStream:=TObjectList.Create(true);
576   VCVs:=TObjectList.Create(true);
577   NumTrials:=0;
578   fTotalTrials:=0;
579   StoreInputs:=False;
580   FLock := TObject.Create;
581 end;
582 procedure TUncertaintyStudy.ClearAll;
583 begin
584   FreeAndNil(InVars);
585   FreeAndNil(OutVars);
586   FreeAndNil(SimpleVars);
587   FreeAndNil(RandomStream);
588   FreeAndNil(VCVs);
589 end;
590 destructor TUncertaintyStudy.destroy;
591 begin
592   ClearAll;
593   FLock.Free;
594   inherited destroy;
595 end;
596
597 procedure TUncertaintyStudy.setTotalTrials(aValue : integer);
598 var i,j : integer;
599 begin
600   if (aValue>=0) and (aValue<1000000) then begin
601     ftotalTrials:=aValue;
602     for i:=0 to InVars.Count-1 do begin
603       setlength(TDistributed(InVars[i]).Points,fTotalTrials+1);
604       for j:=0 to fTotalTrials do TDistributed(InVars[i]).Points[j]:=0;
605       TDistributed(InVars[i]).Points[0]:=TDistributed(InVars[i]).OrigValue;
606     end;
607     for i:=0 to Outvars.Count-1 do begin
608       setlength(TDistributed(OutVars[i]).Points,fTotalTrials+1);
609       for j:=0 to fTotalTrials do TDistributed(OutVars[i]).Points[j]:=0;
610     end;
611     ResetValues;
612   end;
613
614 end;
615
616 function TUncertaintyStudy.FindInVar(Name : string; var ix : integer) : boolean;
617 var i : integer;
618 begin
619   ix:=-1;
620   Result:=False;
```

```
621   for i:=0 to InVars.Count-1 do begin
622     if  CompareText(TDistributed(Invars.Items[i]).Name,Name)=0 then begin
623       ix:=i;
624       Result:=True;
625       break;
626     end;
627   end;
628 end;
629
630 procedure TUncertaintyStudy.CloneInputStream(Src: TUncertaintyStudy);
631 begin
632   var ix: integer;
633   for var i:=0 to InVars.Count-1 do begin
634      Src.FindInVar(TDistributed(Invars.Items[i]).Name,ix);
635      if ix>=0 then begin
636         for var j := 0 to TotalTrials do begin
637            TDistributed(Invars.Items[i]).Points[j] := TDistributed(Src.Invars.Items[ix]).Points[j];
638         end;
639      end;
640   end;
641 end;
642
643 function TUncertaintyStudy.FindOutVar(Name : string; var ix : integer) : boolean;
644 var i : integer;
645 begin
646   ix:=-1;
647   Result:=False;
648   for i:=0 to OutVars.Count-1 do begin
649     if  CompareText(TDistributed(Outvars.Items[i]).Name,Name)=0 then begin
650       ix:=i;
651       Result:=True;
652       break;
653     end;
654   end;
655 end;
656
657 function TUncertaintyStudy.AddInVar(nV : pDouble; nType : integer; nOrig,nMin,nMax,nC,nP1,nP2,nP3 : double; aName,aDesc : string; AllowDistChange :
    boolean=true; CustomList : TStringList=nil; VUFile : string=''; pP : PPercentiles=nil) : TDistributed;
658 var P : TDistributed;
659     i : integer;
660     vidx,midx : integer;
661 begin
662   try
663     System.TMonitor.Enter(FLock);
664     if FindInVar(aName,i) then begin
665       P:=TDistributed(InVars.Items[i]);
666     end else begin
667       P:=TDistributed.Create(nV,nType,nOrig,NMin,NMax,nC,nP1,nP2,nP3,aName,aDesc,CustomList, TotalTrials, AllowDistChange,VUFile,pP);
668       FindVCVMatrix(aName,vidx,midx);
669       if VIDX>=0 then begin
670         TVCV(VCVs.Items[vidx]).Vars[midx]:=P;
671         P.VCVMember:=True;
672         P.VCVName:=TVCV(VCVs.Items[vidx]).Name;
673       end;
674       InVars.Add(P);
675     end;
676     Result:=P;
677   finally
678     System.TMonitor.Exit(FLock);
679   end;
680 end;
681 function TUncertaintyStudy.AddOutVar(nV : pDouble; anAME,ADesc : string) : TDistributed;
682 var P : TDistributed;
683     ix : integer;
684 begin
685     NV^:=0;
686     if FindOutVar(aName,ix) then begin
687       //if exists, just change the location of the value...
688       P:=TDistributed(Outvars.Items[ix]);
689       P.Value:=nV;
690     end else begin
691       P:=TDistributed.Create(nV,dNone,0,0,0,0,0,0,0,aName,aDesc,nil,TotalTrials);
692       OutVars.Add(P);
693     end;
694     Result:=P;
```

```
695 end;
696
697 procedure TUncertaintyStudy.FindVCVMatrix(aName : string; var VCIdx,MemberIX : integer);
698 var i : integer;
699     T : TVCV;
700 begin
701   VCIdx:=-1; MemberIX:=-1;
702   for i:=0 to VCVs.Count-1 do begin
703     T:=TVCV(VCVs.Items[i]);
704     MemberIx:=T.Members.IndexOf(aName);
705     if MemberIX>=0 then begin
706       VCIdx:=i;
707       exit;
708     end;
709   end;
710 end;
711
712 function TUncertaintyStudy.AddVCVMatrix(aName : string; fMembers : string; Filename: string='') : TVCV;
713 var T : TVCV;
714     i : integer;
715 begin
716    for i:=0 to VCVs.Count-1 do begin
717      if TVCV(VCVs.Items[i]).Name=aName then begin
718        Result:=TVCV(VCVs.Items[i]);
719        exit;
720      end;
721    end;
722    T:=TVCV.Create(aName,fMembers, FileName);
723    VCVs.Add(T);
724    Result:=T;
725 end;
726
727 procedure TUncertaintyStudy.ResetValues;
728 var i: integer;
729     T: TDistributed;
730 begin
731    for i:=0 to Invars.count-1 do begin
732      T:=TDistributed(InVars.items[i]);
733      if T.DistType=dVariableCustom then begin
734        //T.Value^:=TSafeWaterPercUncFile.GetMean(T.VCFile,T.PTiles^);
735      end else
736        T.Value^:=T.OrigValue;
737      T.Points[0]:=T.OrigValue;
738    end;
739 end;
740 procedure TUncertaintyStudy.NewTrial;
741 var i: integer;
742 begin
743    for i:=0 to Invars.count-1 do TDistributed(InVars.items[i]).Toss;
744    NumTrials:=NumTrials+1;
745 end;
746 procedure TUncertaintyStudy.SetAllToActiveState(state: boolean);
747 var i: integer;
748 begin
749    for i:=0 to Invars.count-1 do TDistributed(InVars.items[i]).Active := state;
750 end;
751 procedure TUncertaintyStudy.StoreInputTrial(const OnTrial : integer);
752 var i: integer;
753 begin
754    for i:=0 to InVars.Count-1 do TDistributed(InVars.items[i]).StorePoint(OnTrial);
755 end;
756 procedure TUncertaintyStudy.StoreOutputTrial(const OnTrial : integer);
757 var i: integer;
758 begin
759    for i:=0 to OutVars.Count-1 do TDistributed(OutVars.items[i]).StorePoint(OnTrial);
760 end;
761 procedure TUncertaintyStudy.StoreTrial(const OnTrial : integer);
762 begin
763    if StoreInputs then StoreInputTrial(OnTrial);
764    StoreOutputTrial(OnTrial);
765 end;
766 procedure TUncertaintyStudy.GenerateInputStream(Latin : boolean);
767 var i,j,n: integer;
768     pct : double;
769     TL   : TTrial;
```

```
    T   : TDistributed;
    LatinSeries : TObjectLIst;
    v,PctStart,PctEnd : double;

    function GotIt : boolean;
    begin
      inc(n);
      if n=100000 then
        raise exception.Create('cannot generate random value in GenerateInputStream');

      if TL.R<T.Censor then begin
        TL.R:=T.Censor;
        Result:=True;
        exit;
      end;
      Result:=(T.Lower<=TL.R) and (TL.R<=T.Upper);
    end;

begin
   CountExceptions:=0;
   ResetValues;

   if TotalTrials=0 then exit;

   if not Latin then begin
      StoreInputTrial(0);
      for j:=1 to TotalTrials do begin
         for i:=0 to Invars.count-1 do TDistributed(InVars.items[i]).Toss;
         StoreInputTrial(j);
      end
   end else begin
      pct:=1/TotalTrials;
      LatinSeries:=TObjectList.Create(TRUE);
      StoreInputTrial(0);
      for i:=0 to Invars.count-1 do begin
        T:=TDistributed(InVars.items[i]);
        //handled specially...
        if T.DistType=dVariableCustom then continue;
        for j:=1 to TotalTrials do begin
           Tl:=TTrial.Create(0);
           if T.DistType=dCustom then begin
             (*
             TL.R:=Percentile(T.CustomPoints,(J*Pct)-Pct/2);
             *)
           end else
           if (T.DistType=dNone) or (not T.Active) then begin
             TL.R:=T.OrigValue;
           end else begin
             //find if distribution is truncated
             v:=icdf(T.DistType,1e-9,T.Parm1,T.Parm2,T.Parm3);
             if v<T.Lower then begin
               PctStart:=cdf(T.DistType,T.Lower,T.Parm1,T.Parm2,T.Parm3);
             end else
               PctStart:=0;
             v:=icdf(T.DistType,1-1e-9,T.Parm1,T.Parm2,T.Parm3);
             if v>T.Upper then begin
               PctEnd:=cdf(T.DistType,T.Upper,T.Parm1,T.Parm2,T.Parm3);
             end else
               PctEnd:=1;

             n:=0;
             TL.R:=icdf(T.DistType,PctStart + (((J*Pct)-Pct/2) * (PctEnd-PctStart)),T.Parm1,T.Parm2,T.Parm3);
             //TODO MinMax censors?
             if TL.R<T.Censor then TL.R:=T.Censor;
             if not GotIt then
                raise exception.Create('cannot generate random value in GenerateInputStream - Check Upper and Lower bounds for '+T.Name);

           end;
           LatinSeries.Add(TL);
        end;
        LatinSeries.Sort(RandComp);
        for j:=1 to TotalTrials do begin
           TL:=TTrial(LatinSeries.Items[j-1]);
           T.Value^:=TL.R;
```

```
844            T.StorePoint(j);
845          end;
846          LatinSeries.Clear;
847        end;
848        FreeAndNil(LatinSeries);
849     end;
850     NumTrials:=TotalTrials;
851  end;
852
853  procedure TUncertaintyStudy.GenerateRandomStream;
854  var i: integer;
855      T:TTrial;
856      R : double;
857  begin
858     CountExceptions:=0;
859     for i:=0 to TotalTrials do begin
860        R:=Random;
861        T:=TTrial.Create(R);
862        RandomStream.Add(T);
863     end;
864  end;
865
866  procedure TUncertaintyStudy.FetchInputTrial(Trial : integer);
867  var i: integer;
868      T: TDistributed;
869  begin
870     for i:=0 to Invars.count-1 do begin
871       T:=TDistributed(InVars.items[i]);
872       if not T.Active then continue;
873       if T.DistType=dVariableCustom then begin
874         //T.Value^:=TSafeWaterPercUncFile.GetIX(T.VCFile,Trial,T.PTiles^);
875       end else
876         T.Value^:=T.Points[Trial];
877     end;
878  end;
879  function TUncertaintyStudy.GetRandomFromStream(Trial : integer) : double;
880  begin
881     GetRandomFromStream:=TTrial(RandomStream.Items[Trial]).R;
882  end;
883
884  procedure TUncertaintyStudy.SumSingleOutputTrial(T : TDistributed; Trial : integer);
885  begin
886     T.Points[Trial]:=T.Points[Trial]+T.Value^;
887  end;
888
889  procedure TUncertaintyStudy.SumOutputTrial;
890  var i: integer;
891      T: TDistributed;
892  begin
893     for i:=0 to Outvars.count-1 do begin
894       T:=TDistributed(OutVars.items[i]);
895       SumSingleOutputTrial(T,Trial);
896     end;
897  end;
898
899
900  procedure TUncertaintyStudy.ResetRun;
901  begin
902    NumTrials:=0;
903    Invars.destroy;
904    OutVars.destroy;
905  end;
906
907
908  procedure TUncertaintyStudy.DumbPrintToStrings(T : TStrings);
909  var i,j : integer;
910      D : TDistributed;
911  begin
912    for i:=0 to Invars.Count-1 do begin
913      D:=TDistributed(Invars[i]);
914      D.GenerateStats;
915      T.Add(D.Name);
916      T.Add('Mean:'+FloatToStr(D.Stats.Mean));
917      T.Add('StdDev:'+FloatToStr(D.Stats.StdDev));
918      for j:=low(D.Stats.Quantiles) to high(D.Stats.Quantiles) do begin
```

```
      T.Add(inttostr(j)+':'+FloatToStr(D.Stats.Quantiles[j]));
    end;
  end;
end;

procedure TUncertaintyStudy.DumbPrintToCSVStrings(T : TStrings);
var i : integer;
    D : TDistributed;
    s : string;
begin
  T.Add('name,mean,stddev,P1,p5,p50,p95,p99');
  for i:=0 to Invars.Count-1 do begin
    D:=TDistributed(Invars[i]);
    D.GenerateStats;
    S:=D.Name+','+FloatToStr(D.Stats.Mean)+','+FloatToStr(D.Stats.StdDev)+','+
       FloatToStr(D.Stats.Quantiles[1])+','+FloatToStr(D.Stats.Quantiles[5])+','+
       FloatToStr(D.Stats.Quantiles[50])+','+FloatToStr(D.Stats.Quantiles[95])+','+
       FloatToStr(D.Stats.Quantiles[99]);
    T.Add(S);
  end;
end;

procedure TUncertaintyStudy.SetInvarTrialsFromFile(aFilename: string);
begin
  var Data := TStringLIst.Create;
  Data.LoadFromFile(aFIlename);
  var Line := TStringLIst.Create;
  Line.Delimiter:=',';
  Line.StrictDelimiter:=True;
  var Header := TStringLIst.Create;
  Header.Delimiter:=',';
  Header.StrictDelimiter:=True;
  Header.CommaText := Data[0];

  for var j:=0 to TotalTrials do begin
    Line.CommaText := Data[j+1];
    for var i:=0 to Invars.Count-1 do begin
      var D:=TDistributed(Invars[i]);
      var ix := Header.IndexOf(D.Name);
      if ix>=0 then
        D.Points[j] := strtofloat(Line[ix]);
    end;
  end;

  Data.Free;
  Line.Free;
  Header.Free;
end;

procedure TUncertaintyStudy.ReportInvarTrialsToStrings(T : TStrings);
var i,j : integer;
    D : TDistributed;
    s:string;
begin
  S:='Trial';
  for i:=0 to Invars.Count-1 do begin
    D:=TDistributed(Invars[i]);
    S:=S + ',' +D.Name;
  end;
  T.Add(S);

  for j:=0 to TotalTrials do begin
    S:=inttostr(j);
    for i:=0 to Invars.Count-1 do begin
      D:=TDistributed(Invars[i]);
      S:=S+','+FloatToStr(D.Points[j]);
    end;
    T.Add(S);
  end;
end;

procedure TUncertaintyStudy.ReportInvarsToStrings(T : TStrings);
var i: integer;
    D : TDistributed;
```

```
993     s:string;
994 begin
995   S:='Name,PointValue,DistType,Parm1,Parm2,Parm3';
996   T.Add(S);
997   for i:=0 to Invars.Count-1 do begin
998     D:=TDistributed(Invars[i]);
999     S:=D.Name + ',' + FloatToStr(D.OrigValue) + ',' + StrDist(D.DistType) +','+
1000        FloatToStr(D.Parm1) +',' +FloatToStr(D.Parm2)+','+FloatToStr(D.Parm3);
1001     T.Add(S);
1002   end;
1003 end;
1004
1005 procedure TUncertaintyStudy.GenerateStats;
1006 var i: integer;
1007     T: TDistributed;
1008 begin
1009    for i:=0 to Outvars.count-1 do begin
1010      T:=TDistributed(OutVars.items[i]);
1011      T.GenerateStats;
1012    end;
1013    for i:=0 to Invars.count-1 do begin
1014      T:=TDistributed(InVars.items[i]);
1015      T.GenerateStats;
1016    end;
1017 end;
1018
1019 procedure TUncertaintyStudy.SetupVCV;
1020 var j,vidx,midx : integer;
1021     T : TDistributed;
1022 begin
1023   //go through and setup any VCV
1024   for j:=0 to InVars.Count-1 do begin
1025     T:=TDistributed(Invars.Items[j]);
1026     FindVCVMatrix(T.Name,vidx,midx);
1027     if VIDX>=0 then begin
1028       TVCV(VCVs.Items[vidx]).Vars[midx]:=T;
1029       T.VCVMember:=True;
1030       T.VCVName:=TVCV(VCVs.Items[vidx]).Name;
1031     end;
1032   end;
1033
1034 end;
1035
1036
1037 procedure TUncertaintyStudy.LoadFromStream(AStream : TStream);
1038 var i,j : integer;
1039     T : TDistributed;
1040     V : TVCV;
1041 begin
1042   SimpleVars.Clear;
1043   //TODO should we load/save simplevars?
1044   InVars.Clear;
1045   OutVars.Clear;
1046   VCVs.Clear;
1047   NumTrials:=0;
1048   AStream.ReadBuffer(fTotalTrials,SizeOf(fTotalTrials));
1049   AStream.ReadBuffer(I,SizeOf(I));
1050   for j:=1 to i do begin
1051     T:=TDistributed.create(nil,299,0,0,0,0,0,0,0,'','',nil,TotalTrials);
1052     T.LoadFromStream(AStream);
1053     InVars.Add(T);
1054   end;
1055   AStream.ReadBuffer(I,SizeOf(I));
1056   for j:=1 to i do begin
1057     T:=TDistributed.create(nil,299,0,0,0,0,0,0,0,'','',nil,TotalTrials);
1058     T.LoadFromStream(AStream);
1059     OutVars.Add(T);
1060   end;
1061   AStream.ReadBuffer(I,SizeOf(I));
1062   for j:=1 to i do begin
1063     V:=TVCV.Create('','');
1064     V.LoadFromStream(AStream);
1065     VCVs.Add(V);
1066   end;
1067   SetupVCV;
```

```
end;

procedure TUncertaintyStudy.SaveToStream(AStream : TStream);
var i : integer;
begin
  AStream.Write(fTotalTrials,SizeOf(fTotalTrials));
  i:=InVars.Count;
  AStream.Write(I,SizeOf(I));
  for i:=0 to InVars.Count-1 do begin
    TDistributed(InVars[i]).SaveToStream(AStream);
  end;
  i:=OutVars.Count;
  AStream.Write(I,SizeOf(I));
  for i:=0 to OutVars.Count-1 do begin
    TDistributed(OutVars[i]).SaveToStream(AStream);
  end;
  i:=VCVs.Count;
  AStream.Write(I,SizeOf(I));
  for i:=0 to VCVs.Count-1 do begin
    TVCV(VCVs[i]).SaveToStream(AStream);
  end;
end;

function TUncertaintyStudy.FindInVarPointer(Name: string): pDouble;
var i : integer;
begin
  if FindInVar(Name,i) then begin
    Result:=TDistributed(InVars.Items[i]).Value
  end else
    raise exception.Create('Could not find '+Name+' in FindInVarPointer');
end;


begin
end.


```

code\SafeWaterUtility.pas

```
unit SafeWaterUtility;

interface

uses SysUtils, StrUtils, Classes, SafeWaterGlobals, IniFiles, Generics.Collections, Math,
     Windows, psAPI
     ;

function SystemSizeToStr(size:integer): string;
function SourceWaterToStr(source:integer): string;
function OwnershipToStr(owner: integer): string;
function SystemTypeToStr(systype: integer): string;
//function StrToTSystemSize(value:String): integer;
function StrToTSourceWater(value:string): integer;
function StrToTOwnership(value: string): integer;
function StrToTSystemType(value:string): integer;

function StrataToCode(Source,Size,Ownership,SysType: integer) : integer;
procedure CodeToStrata(Code: integer; var Source,Size,Ownership,SysType: integer);
procedure CodeToStrataStr(Code: integer; var Source,Size,Ownership,SysType: string);
function CodeToStrataStrLong(Code: integer) : string;
function PWSEPToCode(const PWSIID: integer; EPNumber: integer): integer;
procedure CodeToPWSEP(const ID: integer; var PWSIID: integer; var EPNumber: integer);
function FIPSLookup(const S: string): integer;


function ReadSubpopData(aFilename: string; var ReadOk: boolean; aColumn: integer = 3; RaiseE: boolean = false): TSubpopArray;
procedure DefaultSubpopData(var RA: TSubpopArray);
function SumSubpopData(const RA: TSubpopArray) : double;
procedure NormalizeSubpopData(var RA: TSubpopArray);
function CopySubpopData(const RA: TSubpopArray) : TSubpopArray;
function ScaleSubpopData(const RA: TSubpopArray; const v: double) : TSubpopArray;
procedure DumpSubPopData(const RA: TSubpopArray; aFilename: string);
```

```
procedure DumpSubPopDataStrings(const RA: TSubpopArray; var SL: TStringList);

procedure SetPaths();
procedure RootToCode(var s: string);
procedure CodeToRoot(var s: string);

procedure WriteIniList(fIni :TIniFile; const Section : string; const Ident: string; const Value: TArray<string>);
function ReadIniList(fIni :TIniFile; const Section : string; const Ident: string) : TArray<string>;

function SigFigs(const Value: double; const SigFigCount: Integer): String;
function SigFigsD(const Value: double; const SigFigCount: Integer): double;

procedure WriteStreamStr(Stream : TStream; Str : string);
function ReadStreamStr(Stream : TStream) : string;
procedure DeleteFiles(Mask: String);

//+=, -= shortcut
procedure _i(var v: double; const a: double); overload;
procedure _i(var v: integer; const a: integer); overload;

function Annualize(const CostCapital : double; const fYears: integer) : double;
procedure InitPreCalcDR2(DiscRate: double; var aPreCalcDiscRate,aPreCalcAnnRate,aPreCalcDiscRateSum : TDRArray);

procedure ProcMsg();
function GetMemoryUsed: UInt64;
function GetProcessorUsage : integer;

//TFile.GetSize replacement for Delhi < 11 - only available in Delphi 11
function TFileGetSize(fn: string): int64;

type
  TSimpleLog=class(Tobject)
  private
    FLock: TObject;
    Lg: TStringLIst;
  public
    constructor Create;
    destructor Destroy; override;

    procedure L(const s: string);
    procedure LD(const s: string; const v: double); overload;
    procedure LD(const s: string; const v: integer); overload;
    procedure LD(const s: string; const v: string); overload;
    procedure Save(aFilename: string);
  end;

  TMultiLog=class(TObject)
  private
     FLock: TObject;
     Lgs: TStringList;
     function GetLog(const LogName: string) : TSimpleLog;
  public
    constructor Create;
    destructor Destroy; override;

    procedure L(const LogName: string; const s: string);
    procedure LD(const LogName: string; const s: string; const v: double); overload;
    procedure LD(const LogName: string; const s: string; const v: integer); overload;
    procedure LD(const LogName: string; const s: string; const v: string); overload;
    procedure Save(aPathName: string);
    function GetLineCount(const LogName: string): integer;
  end;

implementation

function SystemSizeToStr(size:integer): string;
begin
  if (size = ss0_100) then
    Result := 'A) Less than 100'
  else if (size = ss101_500) then
    Result := 'B) 100 to 500'
  else if (size = ss501_1000) then
    Result := 'C) 500 to 1,000'
  else if (size = ss1001_3300) then
```

```
    Result := 'D) 1,000 to 3,300'
  else if (size = ss3301_10k) then
    Result := 'E) 3,300 to 10,000'
  else if (size = ss10k_50k) then
    Result := 'F) 10,000 to 50,000'
  else if (size = ss50k_100k) then
    Result := 'G) 50,000 to 100,000'
  else if (size = ss100k_1m) then
    Result := 'H) 100,000 to 1,000,000'
  else if (size = ss1Mp) then
    Result := 'I) Greater than 1,000,000'
  else if (size = ssAny) then
    Result := 'Any'
  else if (size = ss0_10k) then
    Result := '0) Less than 10,000'
  else if (size = ss10kp) then
    Result := '1) Greater than 10,000'
  else
    raise Exception.Create('Invalid SystemSize of: ' + inttostr(size));
end;

function SourceWaterToStr(source:integer): string;
begin
  if (source = swGroundWater) then
    Result := 'Ground'
  else if (source = swSurfaceWater) then
    Result := 'Surface'
  else if (source = swAny) then
    Result := 'Any'
  else
    raise Exception.Create('Invalid SourceWater of: ' + inttostr(source));
end;

function OwnershipToStr(owner:integer): string;
begin
  if (owner = oPublic) then
    Result := 'Public'
  else if (owner = oPrivate) then
    Result := 'Private'
  else if (owner = oAny) then
    Result := 'Any'
  else
    raise Exception.Create('Invalid Ownership of: ' + inttostr(owner));
end;

function SystemTypeToStr(systype:integer) : string;
begin
  if (systype = sysCWS) then
    Result := 'CWS'
  else if (systype = sysNTNC) then
    Result := 'NTNCWS'
  else if (systype = sysTNC) then
    Result := 'TNC'
  else if (systype = sysAny) then
    Result := 'Any'
  else
   raise Exception.Create('Invalid SystemType of: ' + inttostr(systype));
end;

function StrToTOwnership(value:string): integer;
begin
  if (value = 'Private') then
    Result := oPrivate
  else
  // setting all these to public
  //Local government,State government,Native American,Public/Private,Federal government,
  if (value = 'Public') or (value = 'Local government') or (value = 'State government') or
     (value = 'Native American') or (value = 'Public/Private') or (value = 'Federal government') then
    Result := oPublic
  else
    raise Exception.Create('Invalid Ownership string: ' + value);
end;

function StrToTSystemType(value:string): integer;
begin
```

```
183   if (value = 'Community water system') then
184     Result := sysCWS
185   else if (value = 'Non-Transient non-community system') then
186     Result := sysNTNC
187   else if (value = 'TNCWS') then
188     Result := sysTNC
189   else
190    raise Exception.Create('Invalid System Type string: ' + value);
191 end;
192
193 (*
194 function StrToTSystemSize(value:String): integer;
195 begin
196   if (value = '0_100') then
197     Result := ss0_100
198   else if (value = '101_500') then
199     Result := ss101_500
200   else if (value = '501_1000') then
201     Result := ss501_1000
202   else if (value = '1001_3300') then
203     Result := ss1001_3300
204   else if (value = '3301_10k') then
205     Result := ss3301_10k
206   else if (value = '10k_50k') then
207     Result := ss10k_50k
208   else if (value = '50k_100k') then
209     Result := ss50k_100k
210   else if (value = '100k_1m') then
211     Result := ss100k_1m
212   else
213     raise Exception.Create('Invalid System Size string: ' + value);
214 end;
215 *)
216
217 function StrToTSourceWater(value:string): integer;
218 begin
219   if (value = 'GW') then
220     Result := swGroundWater
221   else if (value = 'SW') then
222     Result := swSurfaceWater
223   else
224     raise Exception.Create('Invalid Source Water string: ' + value);
225 end;
226
227
228 procedure SetPaths();
229 begin
230   AppPath := ExtractFilePath(ParamStr(0));
231   WorkPath := AppPath + 'work\';
232   DataPath := AppPath + 'data\';
233   OutPath := AppPath + 'output\';
234   CfgPath := AppPath + 'configs\';
235   ForceDirectories(WorkPath);
236   ForceDirectories(DataPath);
237   ForceDirectories(OutPath);
238   ForceDirectories(CfgPath);
239 end;
240
241 procedure RootToCode(var s: string);
242 begin
243   if pos(uppercase(AppPath),uppercase(s))=1 then begin
244     s := StringReplace(s,AppPath,'%rp\', [rfIgnoreCase]);
245   end;
246 end;
247
248 procedure CodeToRoot(var s: string);
249 begin
250   if pos(uppercase('%rp\'),uppercase(s))=1 then begin
251     s := StringReplace(s,'%rp\',AppPath,[rfIgnoreCase]);
252   end;
253 end;
254
255 function PWSEPToCode(const PWSIID: integer; EPNumber: integer): integer;
256 begin
```

```
  Result := PWSIID*100 +EPNumber;
end;

procedure CodeToPWSEP(const ID: integer; var PWSIID: integer; var EPNumber: integer);
begin
  PWSIID := ID DIV 100;
  EPNumber := ID MOD 100;
end;

function StrataToCode(Source,Size,Ownership,SysType: integer) : integer;
begin
  Result:= Size*1000000 + Source*10000 + Ownership*100 + SysType;

end;
procedure CodeToStrata(Code: integer; var Source,Size,Ownership,SysType: integer);
begin
  var r := Code;
  Size := r DIV 1000000;
  r := r - (size * 1000000);
  Source := r DIV 10000;
  r := r - (source * 10000);
  Ownership := r DIV 100;
  SysType := r - (Ownership * 100);
end;
procedure CodeToStrataStr(Code: integer; var Source,Size,Ownership,SysType: string);
begin
  var r := Code;
  const iSize = r DIV 1000000;
  r := r - (isize * 1000000);
  const iSource = r DIV 10000;
  r := r - (isource * 10000);
  const iOwnership = r DIV 100;
  const iSysType = r - (iOwnership * 100);
  Source := SourceWaterToStr(isource);
  Size := SystemSizeToStr(iSize);
  Ownership := OwnershipToStr(iOwnership);
  SysType := SystemTypeToStr(iSysType);
end;

function CodeToStrataStrLong(Code: integer) : string;
begin
  var Source,Size,Ownership,SysType: string;
  CodeToStrataStr(Code,Source,Size,Ownership,SysType);
  Result := 'Source: '+ Source+'  Size:'+Size+'  Ownership:'+Ownership+'  Type:'+SysType;
end;

function FIPSLookup(const S: string): integer;
begin
  if S='AL' then Result :=1 else  if S='AK' then Result :=2 else
  if S='AZ' then Result :=4 else  if S='AR' then Result :=5 else
  if S='CA' then Result :=6 else  if S='CO' then Result :=8 else
  if S='CT' then Result :=9 else  if S='DE' then Result :=10 else
  if S='FL' then Result :=12 else if S='GA' then Result :=13 else
  if S='HI' then Result :=15 else if S='ID' then Result :=16 else
  if S='IL' then Result :=17 else if S='IN' then Result :=18 else
  if S='IA' then Result :=19 else if S='KS' then Result :=20 else
  if S='KY' then Result :=21 else if S='LA' then Result :=22 else
  if S='ME' then Result :=23 else if S='MD' then Result :=24 else
  if S='MA' then Result :=25 else if S='MI' then Result :=26 else
  if S='MN' then Result :=27 else if S='MS' then Result :=28 else
  if S='MO' then Result :=29 else if S='MT' then Result :=30 else
  if S='NE' then Result :=31 else if S='NV' then Result :=32 else
  if S='NH' then Result :=33 else if S='NJ' then Result :=34 else
  if S='NM' then Result :=35 else if S='NY' then Result :=36 else
  if S='NC' then Result :=37 else if S='ND' then Result :=38 else
  if S='OH' then Result :=39 else if S='OK' then Result :=40 else
  if S='OR' then Result :=41 else if S='PA' then Result :=42 else
  if S='RI' then Result :=44 else if S='SC' then Result :=45 else
  if S='SD' then Result :=46 else if S='TN' then Result :=47 else
  if S='TX' then Result :=48 else if S='UT' then Result :=49 else
  if S='VT' then Result :=50 else if S='VA' then Result :=51 else
  if S='WA' then Result :=53 else if S='WV' then Result :=54 else
  if S='WI' then Result :=55 else if S='WY' then Result :=56 else
  if S='AS' then Result :=60 else if S='GU' then Result :=66 else
  if S='MP' then Result :=69 else if S='PR' then Result :=72 else
```

```
  if S='VI' then Result :=78 else Result:=0;
end;


procedure WriteIniList(fIni :TIniFile; const Section : string; const Ident: string; const Value: TArray<string>);
begin
  var s: string := '';
  for var f in Value do begin
    if s <>'' then s:=s+',';
    s := s+f;
  end;
  fIni.WriteString(Section,Ident,s);
end;

function ReadIniList(fIni :TIniFile; const Section : string; const Ident: string) : TArray<string>;
begin
  var s := fIni.ReadString(Section, Ident,'');
  Result := SplitString(s,',');
end;

function SigFigs(const Value: double;
                 const SigFigCount: Integer): String;
begin
  Result:=FloatToStrF(Value,ffExponent,SigfigCount,2);
end;

function SigFigsD(const Value: double;
                 const SigFigCount: Integer): double;
begin
  var d := SigFigs(Value,SigFigCount);
  result := strtofloat(D);
end;

procedure WriteStreamStr(Stream : TStream; Str : string);
var
 StrLen : integer;
begin
 StrLen := Length(Str);
 Stream.WriteBuffer(StrLen, SizeOf(Integer));
 Stream.WriteBuffer(Pointer(Str)^, StrLen * SizeOf(Char)); //All Delphi versions compatible
end;

function ReadStreamStr(Stream : TStream) : string;
var
  StrLen : integer;
  TempStr : string;
begin
  TempStr := '';
  Stream.ReadBuffer(StrLen, SizeOf(Integer));
  if StrLen > -1 then begin
    SetLength(TempStr, StrLen);
    Stream.ReadBuffer(Pointer(TempStr)^, StrLen * SizeOf(Char));
    result := TempStr;
  end else Result := '';
end;

procedure DeleteFiles(Mask: String);
var
  SearchRec: SysUtils.TSearchRec;
  X: integer;
  path: String;
  ListToDelete: TStringList;
begin
  ListToDelete := TStringList.create;
  path := ExtractFilePath(Mask);
  X := SysUtils.FindFirst(Mask, faAnyFile - faDirectory - faVolumeID, SearchRec);
  if X = 0 then begin
    while X = 0 do begin
      ListToDelete.Add(path + SearchRec.Name);
      X := SysUtils.FindNext(SearchRec);
    end;
    SysUtils.FindClose(SearchRec);
  end;
  for X := 0 to ListToDelete.Count - 1 do begin
```

```
    try
      FileSetAttr(ListToDelete[X], 0);
      SysUtils.DeleteFile(ListToDelete[X]);
    except
    end;
  end;
  ListToDelete.Free;
end;

procedure _i(var v: double; const a: double); overload;
begin
  v := v + a;
end;
procedure _i(var v: integer; const a: integer); overload;
begin
  v := v + a;
end;

procedure DefaultSubpopData(var RA: TSubpopArray);
begin
  (*
  //unrolled - might help a bit
  var s,r,a : integer;
  for a := 0 to cMaxAge do begin
        RA[gMale,reWhite,a] := 0;
        RA[gFemale,reWhite,a] := 0;
        RA[gMale,reBlack,a] := 0;
        RA[gFemale,reBlack,a] := 0;
        RA[gMale,reHispanic,a] := 0;
        RA[gFemale,reHispanic,a] := 0;
        RA[gMale,reOther,a] := 0;
        RA[gFemale,reOther,a] := 0;
  end;
  *)
  //unrolled - might help a bit
  //TODO warning type specified here....
  const s:integer = (CMaxAge+1) * SizeOf(Single);
  fillchar(RA[gMale,reOther,0],s,0);
  fillchar(RA[gFemale,reOther,0],s,0);
  fillchar(RA[gMale,reHispanic,0],s,0);
  fillchar(RA[gFemale,reHispanic,0],s,0);
  fillchar(RA[gMale,reBlack,0],s,0);
  fillchar(RA[gFemale,reBlack,0],s,0);
  fillchar(RA[gMale,reWhite,0],s,0);
  fillchar(RA[gFemale,reWhite,0],s,0);
end;

function SumSubpopData(const RA: TSubpopArray) : double;
begin
  Result := 0;
  var s,r,a : integer;
  for s := gMale to gFemale do
    for r := reWhite to reOther do
      for a := 0 to cMaxAge do
        Result := Result + RA[s,r,a];
end;

function CopySubpopData(const RA: TSubpopArray) : TSubpopArray;
begin
//TODO - make this like Default suppop data for speedup - CF technique
  var s,r,a : integer;
  for s := gMale to gFemale do
    for r := reWhite to reOther do
      for a := 0 to cMaxAge do
        Result[s,r,a] := RA[s,r,a];
end;

function ScaleSubpopData(const RA: TSubpopArray; const v: double) : TSubpopArray;
begin
  var s,r,a : integer;
  for s := gMale to gFemale do
    for r := reWhite to reOther do
      for a := 0 to cMaxAge do
        Result[s,r,a] := RA[s,r,a] * v;
end;
```

```
481
482 procedure NormalizeSubpopData(var RA: TSubpopArray);
483 begin
484   var sum := SumSubpopData(RA);
485   if sum=0 then exit;
486   var s,r,a : integer;
487   for s := gMale to gFemale do
488     for r := reWhite to reOther do
489       for a := 0 to cMaxAge do
490         RA[s,r,a] := RA[s,r,a] / sum;
491
492 end;
493
494 procedure DumpSubPopData(const RA: TSubpopArray; aFilename: string);
495 begin
496   var TS :=TStreamWriter.Create(aFileName);
497   var s,r,a : integer;
498   for s := gMale to gFemale do
499     for r := reWhite to reOther do
500       for a := 0 to cMaxAge do begin
501         TS.WriteLine(Format('%d,%d,%d,%g',[s,r,a,RA[s,r,a]]));
502       end;
503   TS.Free;
504 end;
505
506 procedure DumpSubPopDataStrings(const RA: TSubpopArray; var SL: TStringList);
507 begin
508   var s,r,a : integer;
509   var ts,tr,ta : string;
510   for s := gMale to gFemale do
511     for r := reWhite to reOther do
512       for a := 0 to cMaxAge do begin
513         if s=gMale then ts:='male' else
514         if s=gFemale then ts:='female' else
515          ts:='bad code';
516         if r=reWhite then tr:='white' else
517         if r=reBlack then tr:='black' else
518         if r=reHispanic then tr:='hispanic' else
519         if r=reOther then tr:='other' else
520            tr:='bad code';
521         ta := inttostr(a);
522         SL.Add(Format('%s,%s,%s,%g',[ts,tr,ta,SigFigsD(RA[s,r,a] ,5)]));
523       end;
524 end;
525
526
527 function ReadSubpopData(aFilename: string; var ReadOk: boolean; aColumn: integer = 3; RaiseE: boolean = false): TSubpopArray;
528 begin
529   ReadOk := true;
530   var s,r,a : integer;
531   DefaultSubpopData(Result);
532
533   var aFS := TBufferedFileStream.Create(aFilename,fmOpenRead + fmShareDenyNone, 32768);
534   var T := TStreamReader.Create(aFS);
535   var hdr := T.ReadLine;
536   var F: TArray<string>;
537   //TODO allow different ordering
538   F := SplitString(hdr, ',');
539   var Offset := 0;
540   //For Liver cancer files from r that have a line number in first column
541   if length(F[0])<3 then
542     Offset := 1;
543   var IL: string;
544
545   while not T.EndOfStream do begin
546     IL := T.ReadLine;
547     IL := StringReplace(IL,'"','',[rfReplaceAll]);
548     F := SplitString(IL, ',');
549     s := -1;
550     if lowercase(F[0+Offset]) = 'male' then s := gMale else
551     if lowercase(F[0+Offset]) = 'female' then s := gFemale;
552     r := -1;
553     var cc := lowercase(F[1+Offset]);
554     if cc = 'white' then r := reWhite else
```

```
555     if cc = 'other' then r := reOther else
556     if cc = 'black' then r := reBlack else
557     if cc = 'hispanic' then r := reHispanic;
558
559     try
560       a := strtoint(F[2+Offset]);
561     except
562       a := -1;
563     end;
564
565     if (s>-1) and (r>-1) and (a>=0) and (a<=85) then
566       try
567         if (CompareText(f[AColumn],'NA')=0) or (CompareText(f[AColumn+Offset],'')=0) then
568           //Result[s,r,a] := vMissing
569           Result[s,r,a] := 0
570         else
571           Result[s,r,a] := strtofloat(F[aColumn+Offset]);
572         if RaiseE then Result[s,r,a] := exp(Result[s,r,a]);
573       except
574         ReadOK := false;
575       end
576     else
577       ReadOK := false;
578   end;
579   T.Free;
580   aFS.Free;
581 end;
582
583 function Annualize(const CostCapital : double; const fYears: integer) : double;
584 begin
585   if fYears<2 then
586     Result :=1
587   else begin
588     if CostCapital <0.000001 then
589       Result := 1/fYears
590     else
591       Result := (CostCapital / (1 - IntPower((1 + CostCapital), -fYears)));
592   end;
593 end;
594
595 procedure InitPreCalcDR2(DiscRate: double; var aPreCalcDiscRate,aPreCalcAnnRate,aPreCalcDiscRateSum : TDRArray);
596 var i : integer;
597 begin
598   var sum : double;
599   sum := 0;
600   for i:=low(aPreCalcDiscRate) to high(aPreCalcDiscRate) do begin
601      aPreCalcDiscRate[i] := intpower((1+DiscRate),i);
602      aPreCalcAnnRate[i] :=  Annualize(DiscRate,i);
603 //      sum := sum + aPreCalcAnnRate[i];
604      sum := sum + 1/aPreCalcDiscRate[i];
605      if DiscRate <0.000001 then
606        aPreCalcDiscRateSum[i] := i
607      else
608        aPreCalcDiscRateSum[i] := sum;
609   end;
610 end;
611
612 procedure ProcMsg();
613 var
614   Msg: TMsg;
615 begin
616   while PeekMessage(Msg, 0, 0, 0, PM_REMOVE) do begin
617     TranslateMessage(Msg);
618     DispatchMessage(Msg);
619   end;
620 end;
621
622
623 //TODO move these to a windows oriented unit for linux/mac command line compilation
624 function GetMemoryUsed: UInt64;
625 var
626   MemCounters: TProcessMemoryCounters;
627 begin
628   Result := 0;
629   MemCounters.cb := SizeOf(MemCounters);
```

```
630   if GetProcessMemoryInfo(GetCurrentProcess,
631       @MemCounters,
632       SizeOf(MemCounters)) then
633     Result := MemCounters.WorkingSetSize
634   else
635     RaiseLastOSError;
636 end;
637
638 type
639   TCPUUsageData=record
640     PID,Handle:cardinal;
641     oldUser,oldKernel:Int64;
642     LastUpdateTime:cardinal;
643     LastUsage:single;
644     //Last result of wsGetCpuUsage is saved here
645     Tag:cardinal;
646     //Use it for anythin you like, not modified by this unit
647   end;
648   PCPUUsageData=^TCPUUsageData;
649
650 var
651   PCPU: PCPUUsageData;
652   PCPUCount: double;
653
654 function GetCpuCount: Word;
655   var SysInfo : _SYSTEM_INFO;
656 begin
657   GetSystemInfo(SysInfo);
658    Result := SysInfo.dwNumberOfProcessors;
659 end;
660
661 function wsCreateUsageCounter(PID:cardinal):PCPUUsageData;
662 var
663     p:PCPUUsageData;
664     mCreationTime,mExitTime,mKernelTime, mUserTime:_FILETIME;
665     h:cardinal;
666 begin
667     result:=nil;
668     //We need a handle with PROCESS_QUERY_INFORMATION privileges
669     h:=OpenProcess(PROCESS_QUERY_INFORMATION,false,PID);
670     if h=0 then exit;
671     new(p);
672     p.PID:=PID;
673     p.Handle:=h;
674     p.LastUpdateTime:=GetTickCount;
675     p.LastUsage:=0;
676     if GetProcessTimes(p.Handle, mCreationTime, mExitTime, mKernelTime, mUserTime) then begin
677         //convert _FILETIME to Int64
678         p.oldKernel:=int64(mKernelTime.dwLowDateTime or (mKernelTime.dwHighDateTime shr 32));
679         p.oldUser:=int64(mUserTime.dwLowDateTime or (mUserTime.dwHighDateTime shr 32));
680         Result:=p;
681     end else begin
682         dispose(p);
683     end;
684 end;
685
686 procedure wsDestroyUsageCounter(aCounter:PCPUUsageData);
687 begin
688     CloseHandle(aCounter.Handle);
689     dispose(aCounter);
690 end;
691
692 function wsGetCpuUsage(aCounter:PCPUUsageData):single;
693 var
694     mCreationTime,mExitTime,mKernelTime, mUserTime:_FILETIME;
695     DeltaMs,ThisTime:cardinal;
696     mKernel,mUser,mDelta:int64;
697 begin
698     result:=aCounter.LastUsage;
699     ThisTime:=GetTickCount; //Get the time elapsed since last query
700
701     DeltaMs:=ThisTime-aCounter.LastUpdateTime;
702     if DeltaMs < 250 then exit;
703     aCounter.LastUpdateTime:=ThisTime;
```

```
704
705     GetProcessTimes(aCounter.Handle,mCreationTime, mExitTime, mKernelTime, mUserTime);
706     //convert _FILETIME to Int64.
707     mKernel:=int64(mKernelTime.dwLowDateTime or (mKernelTime.dwHighDateTime shr 32));
708     mUser:=int64(mUserTime.dwLowDateTime or (mUserTime.dwHighDateTime shr 32));
709     //get the delta
710     mDelta:=mUser+mKernel-aCounter.oldUser-aCounter.oldKernel;
711
712     aCounter.oldUser:=mUser;
713     aCounter.oldKernel:=mKernel;
714
715     Result:=(mDelta/DeltaMs)/100;
716     //mDelta is in units of 100 nanoseconds, so
717
718     aCounter.LastUsage:=Result;
719     //just in case you want to use it later, too
720 end;
721
722
723 function GetProcessorUsage : integer;
724 begin
725   result := round(wsGetCpuUsage(PCPU)/PCPUCount);
726 end;
727
728 { TSimpleLog }
729
730 constructor TSimpleLog.Create;
731 begin
732   FLock := TObject.Create;
733   Lg := TStringLIst.Create;
734 end;
735
736 destructor TSimpleLog.Destroy;
737 begin
738   FLock.Free;
739   Lg.Free;
740   inherited;
741 end;
742
743 procedure TSimpleLog.L(const s: string);
744 begin
745   try
746     System.TMonitor.Enter(FLock);
747     Lg.Add(s);
748   finally
749     System.TMonitor.Exit(FLock);
750   end;
751 end;
752
753 procedure TSimpleLog.LD(const s: string; const v: integer);
754 begin
755   try
756     System.TMonitor.Enter(FLock);
757     Lg.Add(s +': '+inttostr(v));
758   finally
759     System.TMonitor.Exit(FLock);
760   end;
761 end;
762
763 procedure TSimpleLog.LD(const s: string; const v: double);
764 begin
765   try
766     System.TMonitor.Enter(FLock);
767     Lg.Add(s +': '+floattostr(v));
768   finally
769     System.TMonitor.Exit(FLock);
770   end;
771 end;
772
773 procedure TSimpleLog.Save(aFilename: string);
774 begin
775   Lg.SaveToFile(aFilename, TEncoding.ANSI);
776 end;
777
778 procedure TSimpleLog.LD(const s, v: string);
```

```
779 begin
780   try
781     System.TMonitor.Enter(FLock);
782     Lg.Add(s +': '+v);
783   finally
784     System.TMonitor.Exit(FLock);
785   end;
786 end;
787
788 { TMultiLog }
789
790 constructor TMultiLog.Create;
791 begin
792   Lgs:=TStringList.Create;
793   FLock := TObject.Create;
794 end;
795
796 destructor TMultiLog.Destroy;
797 begin
798   for var i := 0 to Lgs.Count-1 do
799     TSimpleLog(Lgs.Objects[i]).Free;
800   FLock.Free;
801   Lgs.Free;
802   inherited;
803 end;
804
805 function TMultiLog.GetLineCount(const LogName: string): integer;
806 begin
807   try
808     System.TMonitor.Enter(FLock);
809     var L := GetLog(LogName);
810     Result := L.Lg.Count;
811   finally
812     System.TMonitor.Exit(FLock);
813   end;
814
815 end;
816
817 function TMultiLog.GetLog(const LogName: string): TSimpleLog;
818 begin
819   var i := Lgs.IndexOf(LogName);
820   if i <0 then begin
821     i := Lgs.AddObject(LogName,TSimpleLog.Create);
822   end;
823   Result := TSimpleLog(Lgs.Objects[i]);
824 end;
825
826 procedure TMultiLog.L(const LogName, s: string);
827 begin
828   try
829     System.TMonitor.Enter(FLock);
830     var L := GetLog(LogName);
831     L.Lg.Add(s);
832   finally
833     System.TMonitor.Exit(FLock);
834   end;
835 end;
836
837 procedure TMultiLog.LD(const LogName, s: string; const v: double);
838 begin
839   try
840     var L := GetLog(LogName);
841     System.TMonitor.Enter(FLock);
842     L.Lg.Add(s +': '+floattostr(v));
843   finally
844     System.TMonitor.Exit(FLock);
845   end;
846
847 end;
848
849 procedure TMultiLog.LD(const LogName, s: string; const v: integer);
850 begin
851   try
852     var L := GetLog(LogName);
```

```
    System.TMonitor.Enter(FLock);
    L.Lg.Add(s +': '+inttostr(v));
  finally
    System.TMonitor.Exit(FLock);
  end;

end;

procedure TMultiLog.LD(const LogName, s, v: string);
begin
  try
    var L := GetLog(LogName);
    System.TMonitor.Enter(FLock);
    L.Lg.Add(s +': '+v);
  finally
    System.TMonitor.Exit(FLock);
  end;
end;

procedure TMultiLog.Save(aPathName: string);
begin
  for var i := 0 to Lgs.Count-1 do begin
     TSimpleLog(Lgs.Objects[i]).Save(aPAthName+'debug_'+Lgs[i]+'.txt');
  end;
end;

function TFileGetSize(fn: string): int64;
begin
  Result := 0;
  if not FileExists(fn) then exit;
  const T = TFileStream.Create(fn, fmOpenRead);
  Result := T.Size;
  T.Free;
end;

initialization
  PCPUCount := GetCPUCount;
  PCPU := wsCreateUsageCounter(GetCurrentProcessId);
finalization
  wsDestroyUsageCounter(PCPU);
end.

```

swcl.dpr

```
program swcl;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  FastMM5,
  System.SysUtils,
  System.IOUtils,
  SafewaterUtility,
  SafewaterGlobals,
  SafeWaterModel in 'code\SafeWaterModel.pas';

begin
  try
    var p1 := ParamStr(1);
    if p1 = '' then begin
      writeln('Safewater command line version: '+swVersion+' date: '+swDate);
      writeln('usage:');
      writeln('swcl MyInput.ini');
      writeln('  swcl MyInput.ini - will run MyInput.ini ');
      //write help
      exit;
    end;

    if not TFile.Exists(P1) then begin
      //write error;
```

```
      writeln('File not found: ' + P1);
      exit;
    end;


    SetPaths();

  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.

```

uicode\wndTests.dfm

```
object wTests: TwTests
  Left = 0
  Top = 0
  Caption = 'SW Tests'
  ClientHeight = 493
  ClientWidth = 574
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -12
  Font.Name = 'Segoe UI'
  Font.Style = []
  TextHeight = 15
  object Memo1: TMemo
    Left = 0
    Top = 328
    Width = 574
    Height = 165
    Align = alBottom
    Lines.Strings = (
      'Memo1')
    ScrollBars = ssVertical
    TabOrder = 0
  end
  object Button2: TButton
    Left = 16
    Top = 11
    Width = 193
    Height = 24
    Caption = 'Test Occ File'
    TabOrder = 1
    OnClick = Button2Click
  end
  object Button3: TButton
    Left = 389
    Top = 8
    Width = 129
    Height = 24
    Caption = 'PWS Records'
    TabOrder = 2
    OnClick = Button3Click
  end
  object Button4: TButton
    Left = 16
    Top = 41
    Width = 193
    Height = 27
    Caption = 'Model Skeleton'
    TabOrder = 3
    OnClick = Button4Click
  end
  object Button5: TButton
    Left = 389
    Top = 38
    Width = 129
    Height = 24
    Caption = 'Config'
```

```
    TabOrder = 4
    OnClick = Button5Click
  end
  object Button1: TButton
    Left = 16
    Top = 86
    Width = 193
    Height = 25
    Caption = 'Make ICF Input File'
    TabOrder = 5
    OnClick = Button1Click
  end
  object edSigFigs: TEdit
    Left = 225
    Top = 87
    Width = 26
    Height = 23
    TabOrder = 6
    Text = '3'
  end
  object edTestLoops: TEdit
    Left = 225
    Top = 44
    Width = 40
    Height = 23
    TabOrder = 7
    Text = '1'
  end
  object Button6: TButton
    Left = 389
    Top = 68
    Width = 129
    Height = 25
    Caption = 'Make Mock BW/Cons'
    TabOrder = 8
    OnClick = Button6Click
  end
  object Button7: TButton
    Left = 389
    Top = 99
    Width = 129
    Height = 25
    Caption = 'Clean ICF CountyPOp'
    TabOrder = 9
    OnClick = Button7Click
  end
  object Button8: TButton
    Left = 389
    Top = 136
    Width = 129
    Height = 25
    Caption = 'Cohort By ID'
    TabOrder = 10
    OnClick = Button8Click
  end
  object Button9: TButton
    Left = 16
    Top = 152
    Width = 193
    Height = 25
    Caption = 'Combine OCC'
    TabOrder = 11
    OnClick = Button9Click
  end
  object Button10: TButton
    Left = 16
    Top = 208
    Width = 193
    Height = 25
    Caption = 'Test Occ 2'
    TabOrder = 12
  end
  object Button11: TButton
    Left = 389
    Top = 176
```

```
    Width = 129
    Height = 25
    Caption = 'PWS Pop Prop'
    TabOrder = 13
    OnClick = Button11Click
  end
  object Button12: TButton
    Left = 389
    Top = 223
    Width = 129
    Height = 25
    Caption = 'VLS Occurrence'
    TabOrder = 14
    OnClick = Button12Click
  end
  object Button13: TButton
    Left = 16
    Top = 264
    Width = 193
    Height = 25
    Caption = 'Adhoc Combine'
    TabOrder = 15
    OnClick = Button13Click
  end
  object Edit1: TEdit
    Left = 225
    Top = 265
    Width = 121
    Height = 23
    TabOrder = 16
    Text = '10000'
  end
  object Button14: TButton
    Left = 16
    Top = 304
    Width = 193
    Height = 25
    Caption = 'Adhob Make Bin'
    TabOrder = 17
    OnClick = Button14Click
  end
  object Button15: TButton
    Left = 389
    Top = 280
    Width = 129
    Height = 25
    Caption = 'EJProportions file'
    TabOrder = 18
    OnClick = Button15Click
  end
  object OD: TOpenDialog
    Left = 43
    Top = 421
  end
  object SD: TSaveDialog
    Left = 136
    Top = 408
  end
end

```

uicode\wndTests.pas

```
unit wndTests;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, StrUtils;

type
  TwTests = class(TForm)
```

```
11      Memo1: TMemo;
12      Button2: TButton;
13      Button3: TButton;
14      Button4: TButton;
15      Button5: TButton;
16      Button1: TButton;
17      OD: TOpenDialog;
18      SD: TSaveDialog;
19      edSigFigs: TEdit;
20      edTestLoops: TEdit;
21      Button6: TButton;
22      Button7: TButton;
23      Button8: TButton;
24      Button9: TButton;
25      Button10: TButton;
26      Button11: TButton;
27      Button12: TButton;
28      Button13: TButton;
29      Edit1: TEdit;
30      Button14: TButton;
31      Button15: TButton;
32      procedure Button2Click(Sender: TObject);
33      procedure Button3Click(Sender: TObject);
34      procedure Button4Click(Sender: TObject);
35      procedure Button5Click(Sender: TObject);
36      procedure Button1Click(Sender: TObject);
37      procedure Button6Click(Sender: TObject);
38      procedure Button7Click(Sender: TObject);
39      procedure Button8Click(Sender: TObject);
40      procedure Button9Click(Sender: TObject);
41      procedure Button11Click(Sender: TObject);
42      procedure Button12Click(Sender: TObject);
43      procedure Button13Click(Sender: TObject);
44      procedure Button14Click(Sender: TObject);
45      procedure Button15Click(Sender: TObject);
46    private
47      { Private declarations }
48    public
49      { Public declarations }
50    end;
51
52  var
53    wTests: TwTests;
54
55  implementation
56
57  uses SafewaterGlobals, SafeWaterOccurrence, SafewaterPWSRecords, SafewaterModel, SafewaterConfig,
58       SafewaterCohortArray, SafewaterUtility;
59
60  {$R *.dfm}
61
62  procedure TwTests.Button11Click(Sender: TObject);
63  begin
64    var CFIPS := TCohortDataByIDs.create();
65    CFIPS.LoadFromFile('K:\Safewater2022\data\Cohortdata\FIPS_PopPropAvg.csv', True);
66    var T :=  TStreamReader.Create('K:\Safewater2022\Population_Proportion_Table.csv');
67    var Ot :=  TStreamWriter.Create('K:\Safewater2022\data\Cohortdata\PWS_PopPropAvg.csv');
68    OT.WriteLine('pwsid,sex,race_eth,age,value');
69    var Ot2 :=  TStreamWriter.Create('K:\Safewater2022\data\Cohortdata\PWS_PopPropAvgList.csv');
70    OT2.WriteLine('pwsid');
71
72    var s,f,lastpws,pws: string;
73    var fipscnt: integer;
74    var P: PSubPopArray;
75    var R: TSubPopArray;
76    var Inp:=TSTringLIst.Create;
77    var OSL:=TSTringLIst.Create;
78    var totcnt:=0;
79    var v: double;
80    s:=T.ReadLine;
81    lastpws:='';
82    while not T.EndOfStream do begin
83      s:=T.ReadLine;
84      inp.CommaText := S;
```

```
85      pws:=inp[0];
86
87      if pws<>lastpws then begin
88        //output
89        if lastpws<>'' then begin
90          OSL.Clear;
91          DumpSubPopDataStrings(R,OSL);
92          for s in OSL do
93            OT.WriteLine(lastpws+','+s);
94          OT2.WriteLine(lastpws);
95          inc(totcnt);
96        end;
97
98        fipscnt:=0;
99        DefaultSubpopData(R);
100       lastpws:=pws;
101     end;
102
103     v:=strtofloat(inp[2]);
104     f:=inp[1];
105     if f[1]='0' then delete(f,1,1);
106     P:=CFips.GetCohortData(f);
107     if p=nil then begin
108       //debug breakpoint
109       s:='f';
110     end;
111
112     for var sx:=gMale to gFemale do begin
113       for var rc:=reWhite to reOther do begin
114         for var a:=0 to CMaxAge do begin
115           R[sx,rc,a] := R[sx,rc,a] + P[sx,rc,a] * v;
116         end;
117       end;
118     end;
119
120     inc(fipscnt);
121   end;
122
123   //output last...
124   OSL.Clear;
125   DumpSubPopDataStrings(R,OSL);
126   for s in OSL do
127     OT.WriteLine(Pws+','+s);
128   OT2.WriteLine(pws);
129   inc(totcnt);
130
131   OSL.Free;
132   INp.Free;
133   CFIPS.Free;
134   T.Free;
135   Ot.Free;
136   Ot2.Free;
137 end;
138
139 procedure TwTests.Button12Click(Sender: TObject);
140 begin
141   var T := TStreamWriter.Create('K:\safewater2022\data\PFAS_VLS_occurrence_4000.csv');
142   T.WriteLine('PWSID,iteration,entry_point,data,PFHpA,PFHxS,PFOS,PFOA');
143   const l = 'FL4130871_1,iii,1,99,1,1,13,4.7'+#13#10+
144     'FL4130871_2,iii,1,99,1,1,13,4.7'+#13#10+
145     'FL4130871_3,iii,1,99,1,1,22,5.6'+#13#10+
146     'FL4130871_4,iii,1,99,1,1,34,11'+#13#10+
147     'NY5110526_1,iii,1,99,1,1,8,5.28'+#13#10;
148   for var i  := 1 to 4000 do begin
149     var nl := StringReplace(l,'iii',i.ToString,[rfReplaceAll]);
150     T.Write(nl);
151   end;
152   T.Free;
153   showmessage('done');
154 end;
155
156 procedure DoIt(Scen,Path: string; MaxIter:integer);
157 begin
158   var hdr := 'PWSID,iteration,entry_point,PFOS,PFOA,PFHXS,PFHPA,PFBS,PFNA,HFPODA';
159   var C :TArray<string> := ['PFOS', 'PFOA', 'PFHXS', 'PFHPA',
```

```
    'PFBS', 'PFNA', 'HFPODA' ];

  var Files: TArray<TStreamReader>;
  var OFName: string;
  if MaxIter<4000 then
    OFName := path+'PFAS_Occurrence_40007C_'+Scen+'_Iter'+inttostr(MaxIter)+'.csv'
  else
    ofName := path+'PFAS_Occurrence_40007C_'+Scen+'.csv';
  var oFile:= TStreamWriter.Create(OFName,False,TEncoding.ASCII,4096*8);
  oFile.WriteLine(hdr);
  setlength(Files,7);

  for var i:=0 to length(c)-1 do begin
    if i<4 then begin
      Files[i] := TStreamReader.Create(path+'PFAS_occurrence_iter_'+C[i]+'_full_samps.csv');
    end else begin
      Files[i] := TStreamReader.Create(path+'PFAS_occurrence_iter_'+Scen+'_'+C[i]+'.csv');
    end;
    Files[i].ReadLine;
  end;

  var ILine := TStringList.Create;
  ILine.StrictDelimiter:=True;
  var ILine2 := TStringList.Create;
  ILine2.StrictDelimiter:=True;
  var S: String;
  var D: double;
  var OOO := 0;
  while not Files[0].EndOfStream do begin
    ILine.CommaText := Files[0].ReadLine;
    S := ILine[0]+','+ILine[1]+','+ILine[2];
    D:=SigFigsD(strtofloat(ILine[3]),4);
    S := S + format(',%g',[D]);
    for var i:= 1 to length(C) -1 do begin
      ILine2.CommaText := Files[i].ReadLine;
      //if (CompareText(ILine[0],ILine2[0]) <> 0) or
      //   (CompareText(ILine[1],ILine2[1]) <> 0) or
      //   (CompareText(ILine[2],ILine2[2]) <> 0) then
      //     inc(ooo);
       D:=SigFigsD(strtofloat(ILine2[3]),4);
       S := S + format(',%g',[D]);
    end;
    if strtoint(ILine[1])<MaxIter then begin
      ofile.Writeline(s);
    end;
  end;

  //ShowMessage('ooo:'+ooo.ToString+'   done');
  ILine.Free;
  ILine2.Free;
  for var i:=0 to length(c)-1 do Files[i].Free;
  oFile.Free;
//PFAS_occurrence_iter_1EP_EqDis_HFPODA
end;

procedure ProcessIt(Scen,Path: string);
begin
  var T := TOccurrenceData.create();
  T.LoadFromFile(path+scen+'.csv');
  T.Free;
end;

procedure TwTests.Button13Click(Sender: TObject);
begin
  DoIt('1EP_EqDis','c:\scratch\',strtoint(Edit1.Text));
  DoIt('AllEP_EqDis','c:\scratch\',strtoint(Edit1.Text));
  DoIt('1EP_Redund','c:\scratch\',strtoint(Edit1.Text));
  DoIt('AllEP_Redund','c:\scratch\',strtoint(Edit1.Text));
  Memo1.Lines.Add('Done All');
end;

procedure TwTests.Button14Click(Sender: TObject);
begin

```

```
234   ProcessIt('PFAS_Occurrence_40007C_1EP_EqDis','c:\scratch\');
235   ProcessIt('PFAS_Occurrence_40007C_AllEP_EqDis','c:\scratch\');
236   ProcessIt('PFAS_Occurrence_40007C_1EP_Redund','c:\scratch\');
237   ProcessIt('PFAS_Occurrence_40007C_AllEP_Redund','c:\scratch\');
238   Memo1.Lines.Add('Done All');
239
240
241 end;
242
243 procedure TwTests.Button15Click(Sender: TObject);
244 begin
245   var t := TEJProportionsByIDs.create;
246   t.LoadFromFile('K:\Safewater2022\data\CohortData\PWS_EJProportions.csv');
247
248   var a := T.GetAllProportionData('90400020');
249   a := T.GetAllProportionData('904');
250   a := T.GetAllProportionData('AL0000213');
251   t.Free;
252 end;
253
254 procedure TwTests.Button1Click(Sender: TObject);
255 var T: TOccurrenceData;
256 begin
257   if not SD.Execute() then exit;
258
259   T := TOccurrenceData.create();
260   T.LoadFromFile('K:\safewater11\data\2022-02-23_PFAS_occurrence.csv');
261 //  T.MakeICFInputFile(SD.FileName, strtoint(edSigFigs.Text));
262   T.Free;
263 end;
264
265 procedure TwTests.Button2Click(Sender: TObject);
266 var T: TOccurrenceData;
267 begin
268   T := TOccurrenceData.create();
269
270 //T.MakeSubset('K:\safewater2022\data\2022-04-01_PFAS_occurrence.csv',5);
271 //T.Free;
272 //exit;
273
274   T.LoadFromFile('K:\safewater2022\data\PFAS_092023_small.csv');
275   var TS := T.OccurrenceReport;
276   TS.SaveToFile('K:\safewater2022\data\PFAS_092023_occrep_small.txt');
277   TS.Free;
278 //  var ok: boolean;
279 //  TS := T.VerifyIterations(ok);
280 //  TS.SaveToFile('K:\safewater2022\data\PFAS_occurrence_4000.csv_verifylog.txt');
281 //  TS.Free;
282   T.Free;
283   memo1.lines.add('done');
284 end;
285
286 procedure TwTests.Button3Click(Sender: TObject);
287 var T: TPWSRecords;
288 begin
289   var d : TCohortdata;
290   T := TPWSRecords.Create;
291   T.LoadFromFile('K:\safewater2022\data\2022-02-08_CWS_Entry_Points.csv',nil,nil,nil,d,0,false,true);
292   T.LoadFromFile('K:\safewater2022\data\2022-02-08_NTNCWS_Entry_Points.csv',nil,nil,nil,d,0,false,false);
293
294   T.SaveCategoryCounts('PWSCatCountsCode.csv',False);
295   T.SaveCategoryCounts('PWSCatCounts.csv');
296
297   T.Free;
298
299 end;
300
301 procedure TwTests.Button4Click(Sender: TObject);
302 var T: TSafeWaterModel;
303 begin
304   for var i :=1 to strtoint(edTestLoops.Text) do begin
305     T := TSafeWaterModel.create;
306     T.Initialize('test',AppPath+'test.ini');
307     T.Run;
308     //T.RunOne;
```

```
    T.Free;
    CSL('Done loop:'+i.ToString);
  end;
  Memo1.Lines.Add('done');
end;

procedure TwTests.Button5Click(Sender: TObject);
var T: TSafeWaterConfig;
begin
  T := TSafeWaterConfig.create;
  T.SaveToFile(AppPath+'config.ini');
  T.Free;
end;

procedure TwTests.Button6Click(Sender: TObject);
begin
  var s,r,a : integer;
  var ss,rr:string;

  var t := TStreamWriter.Create('k:\safewater2022\data\Cohortdata\BodyWeightsMock.csv',false,TEncoding.ASCII,4096);
  T.WriteLine('sex,race_eth,age,value');
  for s := gMale to gFemale do
    for r := reWhite to reOther do
      for a := 0 to cMaxAge do begin
        if s=gMale then ss:='male' else ss:='female';
        case r of
          reWhite: rr:='white';
          reBlack: rr:='black';
          reHispanic: rr:='hispanic';
          reOther: rr:='other';
        end;
        T.WriteLine(Format('%s,%s,%d,%d',[ss,rr,a,a+1]));
      end;
  T.Free;

  t := TStreamWriter.Create('k:\safewater2022\data\Cohortdata\cvd\test.csv',false,TEncoding.ASCII,4096);
  T.WriteLine('sex,race_eth,age,value');
  for s := gMale to gFemale do
    for r := reWhite to reOther do
      for a := 0 to cMaxAge do begin
        if s=gMale then ss:='male' else ss:='female';
        case r of
          reWhite: rr:='white';
          reBlack: rr:='black';
          reHispanic: rr:='hispanic';
          reOther: rr:='other';
        end;
        T.WriteLine(Format('%s,%s,%d,%g',[ss,rr,a,(a+1)/100]));
      end;
  T.Free;


  t := TStreamWriter.Create('k:\safewater2022\data\Cohortdata\ConsumptionMock.csv',false,TEncoding.ASCII,4096);
  T.WriteLine('sex,race_eth,age,value');
  for s := gMale to gFemale do
    for r := reWhite to reOther do
      for a := 0 to cMaxAge do begin
        if s=gMale then ss:='male' else ss:='female';
        case r of
          reWhite: rr:='white';
          reBlack: rr:='black';
          reHispanic: rr:='hispanic';
          reOther: rr:='other';
        end;
        var g := (a+1)/100;
        T.WriteLine(Format('%s,%s,%d,%g',[ss,rr,a,g]));
      end;
  T.Free;

end;

procedure TwTests.Button7Click(Sender: TObject);
begin
  var t := TStreamWriter.Create('k:\safewater2022\data\CohortData\FIPS_PopProp.csv',false,TEncoding.ASCII,4096);
```

```
383   T.WriteLine('fips,sex,race_eth,age,value');
384   var inf := TStreamReader.Create('k:\safewater2022\data\CohortData\COUNTY_2020_PropPop.csv');
385   var iline: TArray<string>;
386   iline:=SplitString(inf.ReadLine,',');
387   var fips,sex,race_eth,age: string;
388   var value:double;
389   while not inf.EndOfStream do begin
390     var s:=inf.ReadLine;
391     s:=StringReplace(s,'"','',[rfReplaceAll]);
392     iline:=SplitString(s,',');
393     fips := iline[0];
394     race_eth:=lowercase(iline[1]);
395     if race_eth[1]='n' then Delete(race_eth,1,2);
396     sex:=iline[2];
397     if sex ='F' then sex:='female' else sex:='male';
398     age:=iline[3];
399     if age[1]='0' then delete(age,1,1);
400     if age[1]='0' then delete(age,1,1);
401     if age[1]='0' then delete(age,1,1);
402
403
404     if uppercase(age)='85UP'  then
405       age:='85';
406     value:=strtofloat(iline[7]);
407     value := round(value*10000)/10000;
408     T.WriteLine(Format('%s,%s,%s,%s,%g',[fips,sex,race_eth,age, value]));
409   end;
410   inf.Free;
411   t.Free;
412
413
414 end;
415
416 procedure TwTests.Button8Click(Sender: TObject);
417 begin
418   const T = TCohortDataByIds.create;
419   T.LoadFromFile('K:\Safewater2022\data\CohortData\FIPS_PopProp.csv');
420   T.Free;
421 end;
422
423 procedure TwTests.Button9Click(Sender: TObject);
424 begin
425   var T:=TStringLIst.Create;
426   T.Add('K:\Safewater2022\data\tempocc\PFAS_occurrence_iter_1_99.csv');
427   for var i := 1 to 38 do begin
428     T.Add('K:\Safewater2022\data\tempocc\PFAS_occurrence_iter_'+inttostr(i*100)+'_'+inttostr(i*1)+'99.csv');
429   end;
430   T.Add('K:\Safewater2022\data\tempocc\PFAS_occurrence_iter_3900_4000.csv');
431
432   var s: string;
433   var OutF := TStreamWriter.Create('K:\Safewater2022\data\PFAS_occurrence_4000.csv',false,TEncoding.ANSI,4096);
434   for var i := 0 to T.Count-1 do begin
435     var InF := TStreamReader.Create(T[i]);
436     if i>0 then s:=Inf.ReadLine;
437     while not InF.EndOfStream do begin
438       s:=Inf.ReadLine;
439       OutF.WriteLine(s);
440     end;
441     InF.Free;
442   end;
443   memo1.Lines.Add('done');
444   OutF.Free;
445   T.Free;
446 end;
447
448 end.
449
```

wndMain.dfm

```
1 object wMain: TwMain
2   Left = 0
3   Top = 0
```

```
  BorderStyle = bsSingle
  Caption = 'Safewater Run Editor'
  ClientHeight = 841
  ClientWidth = 1070
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -12
  Font.Name = 'Segoe UI'
  Font.Style = []
  Position = poScreenCenter
  ShowHint = True
  OnCreate = FormCreate
  OnDestroy = FormDestroy
  OnShow = FormShow
  TextHeight = 15
  object ScrollBox1: TScrollBox
    AlignWithMargins = True
    Left = 3
    Top = 111
    Width = 566
    Height = 727
    VertScrollBar.Position = 443
    Align = alLeft
    Ctl3D = False
    ParentCtl3D = False
    TabOrder = 0
    ExplicitHeight = 718
    object CfgPanel: TStackPanel
      Left = 21
      Top = -437
      Width = 508
      Height = 1250
      BevelOuter = bvNone
      ControlCollection = <
        item
          Control = Panel7
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel1
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel13
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel14
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel15
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel16
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel3
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel4
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel5
          HorizontalPositioning = sphpFill
        end
        item
          Control = Panel19
          HorizontalPositioning = sphpFill
        end
```

```
79          item
80            Control = Panel20
81            HorizontalPositioning = sphpFill
82          end
83          item
84            Control = Panel6
85            HorizontalPositioning = sphpFill
86          end
87          item
88            Control = Panel2
89            HorizontalPositioning = sphpFill
90          end
91          item
92            Control = Panel8
93            HorizontalPositioning = sphpFill
94          end
95          item
96            Control = Panel9
97            HorizontalPositioning = sphpFill
98          end
99          item
100           Control = Panel23
101           HorizontalPositioning = sphpFill
102         end
103         item
104           Control = Panel22
105           HorizontalPositioning = sphpFill
106         end>
107       Ctl3D = False
108       ParentCtl3D = False
109       TabOrder = 0
110       object Panel7: TPanel
111         Left = 0
112         Top = 0
113         Width = 508
114         Height = 35
115         BevelOuter = bvNone
116         BorderStyle = bsSingle
117         TabOrder = 6
118         object Label8: TLabel
119           Left = 8
120           Top = 8
121           Width = 52
122           Height = 19
123           Caption = 'General'
124           Font.Charset = DEFAULT_CHARSET
125           Font.Color = clWindowText
126           Font.Height = -14
127           Font.Name = 'Segoe UI'
128           Font.Style = [fsBold]
129           ParentFont = False
130         end
131       end
132       object Panel1: TPanel
133         Left = 0
134         Top = 37
135         Width = 508
136         Height = 49
137         BevelOuter = bvNone
138         BorderStyle = bsSingle
139         TabOrder = 0
140         object Label3: TLabel
141           Left = 24
142           Top = 16
143           Width = 82
144           Height = 15
145           Caption = 'Occrurence File'
146         end
147         object edOccFile: TEdit
148           AlignWithMargins = True
149           Left = 136
150           Top = 11
151           Width = 353
152           Height = 21
```

```
153          Hint = 'Double click to edit'
154          TabOrder = 0
155          Text = '***'
156          OnDblClick = edOccFileDblClick
157        end
158      end
159      object Panel13: TPanel
160        Left = 0
161        Top = 88
162        Width = 508
163        Height = 49
164        BevelOuter = bvNone
165        BorderStyle = bsSingle
166        TabOrder = 9
167        object Label13: TLabel
168          Left = 24
169          Top = 16
170          Width = 89
171          Height = 15
172          Caption = 'Body Weight File'
173        end
174        object edBWFile: TEdit
175          AlignWithMargins = True
176          Left = 136
177          Top = 11
178          Width = 353
179          Height = 21
180          Hint = 'Double click to edit'
181          TabOrder = 0
182          Text = '***'
183          OnDblClick = edOccFileDblClick
184        end
185      end
186      object Panel14: TPanel
187        Left = 0
188        Top = 139
189        Width = 508
190        Height = 49
191        BevelOuter = bvNone
192        BorderStyle = bsSingle
193        TabOrder = 10
194        object Label14: TLabel
195          Left = 24
196          Top = 16
197          Width = 94
198          Height = 15
199          Caption = 'Consumption File'
200        end
201        object edConsFile: TEdit
202          AlignWithMargins = True
203          Left = 136
204          Top = 11
205          Width = 353
206          Height = 21
207          Hint = 'Double click to edit'
208          TabOrder = 0
209          Text = '***'
210          OnDblClick = edOccFileDblClick
211        end
212      end
213      object Panel15: TPanel
214        Left = 0
215        Top = 190
216        Width = 508
217        Height = 49
218        BevelOuter = bvNone
219        BorderStyle = bsSingle
220        TabOrder = 11
221        object Label18: TLabel
222          Left = 24
223          Top = 16
224          Width = 87
225          Height = 15
226          Caption = 'PWS Subpop file'
227        end
```

```
228         object edPWSSPFile: TEdit
229           AlignWithMargins = True
230           Left = 136
231           Top = 10
232           Width = 353
233           Height = 21
234           Hint = 'Double click to edit'
235           TabOrder = 0
236           Text = '***'
237           OnDblClick = edOccFileDblClick
238         end
239       end
240       object Panel16: TPanel
241         Left = 0
242         Top = 241
243         Width = 508
244         Height = 49
245         BevelOuter = bvNone
246         BorderStyle = bsSingle
247         TabOrder = 12
248         object Label19: TLabel
249           Left = 24
250           Top = 16
251           Width = 102
252           Height = 15
253           Caption = 'County Subpop file'
254         end
255         object edCountySPFile: TEdit
256           AlignWithMargins = True
257           Left = 136
258           Top = 11
259           Width = 353
260           Height = 21
261           Hint = 'Double click to edit'
262           TabOrder = 0
263           Text = '***'
264           OnDblClick = edOccFileDblClick
265         end
266       end
267       object Panel3: TPanel
268         Left = 0
269         Top = 292
270         Width = 508
271         Height = 65
272         BevelOuter = bvNone
273         BorderStyle = bsSingle
274         TabOrder = 2
275         object Label5: TLabel
276           Left = 24
277           Top = 16
278           Width = 60
279           Height = 15
280           Caption = 'SDWIS Files'
281         end
282         object edSdwisFiles: TMemo
283           AlignWithMargins = True
284           Left = 136
285           Top = 3
286           Width = 361
287           Height = 57
288           Hint = 'Double click to edit'
289           HideSelection = False
290           ScrollBars = ssVertical
291           TabOrder = 0
292           WordWrap = False
293           OnDblClick = edSdwisFilesDblClick
294         end
295       end
296       object Panel4: TPanel
297         Left = 0
298         Top = 359
299         Width = 508
300         Height = 128
301         BevelOuter = bvNone
```

```
302         BorderStyle = bsSingle
303         TabOrder = 3
304         object Label1: TLabel
305           Left = 24
306           Top = 16
307           Width = 69
308           Height = 15
309           Caption = 'Cost Options'
310         end
311         object edCostOptions: TCheckListBox
312           Left = 136
313           Top = 3
314           Width = 353
315           Height = 120
316           BevelOuter = bvNone
317           Columns = 2
318           Ctl3D = False
319           ItemHeight = 15
320           Items.Strings = (
321             'PFASOption1a'
322             'PFASOption1b'
323             'PFASOption1c'
324             'PFASOption1d'
325             'PFASOption2'
326             'PFASOption2s'
327             'PFASOption3'
328             'PFASOption4'
329             'PFASOption2x'
330             'PFASOption2x-PFNA'
331             'PFASOption2x-HFPODA'
332             'PFASOption2x-PFHXS'
333             'PFASCOOC'
334             'PFASMCL-PFNA'
335             'PFASMCL-HFPODA'
336             'PFASMCL-PFHXS'
337             'PFASOccRun')
338           ParentCtl3D = False
339           TabOrder = 0
340         end
341       end
342       object Panel5: TPanel
343         Left = 0
344         Top = 489
345         Width = 508
346         Height = 65
347         BevelOuter = bvNone
348         BorderStyle = bsSingle
349         TabOrder = 4
350         object Label6: TLabel
351           Left = 24
352           Top = 16
353           Width = 82
354           Height = 15
355           Caption = 'Benefit Options'
356         end
357         object edBenefitOptions: TCheckListBox
358           Left = 136
359           Top = 3
360           Width = 353
361           Height = 57
362           BevelOuter = bvNone
363           Columns = 2
364           Ctl3D = False
365           ItemHeight = 15
366           Items.Strings = (
367             'PFASCVD'
368             'PFASKidneyCanc'
369             'PFASLiverCanc'
370             'PFASLBW'
371             'PFASDBP'
372             'PFASSTATS')
373           ParentCtl3D = False
374           TabOrder = 0
375         end
376       end
```

```
377       object Panel19: TPanel
378         Left = 0
379         Top = 556
380         Width = 508
381         Height = 282
382         BevelOuter = bvNone
383         BorderStyle = bsSingle
384         TabOrder = 13
385         object Label22: TLabel
386           Left = 312
387           Top = 17
388           Width = 73
389           Height = 15
390           Caption = 'Discount Rate'
391           Visible = False
392         end
393         object Label26: TLabel
394           Left = 22
395           Top = 17
396           Width = 210
397           Height = 15
398           Caption = 'Discount Rates (comma seperated in %)'
399         end
400         object Label39: TLabel
401           Left = 182
402           Top = 142
403           Width = 72
404           Height = 15
405           Caption = 'Yearly Output'
406         end
407         object Label35: TLabel
408           Left = 49
409           Top = 142
410           Width = 78
411           Height = 15
412           Caption = 'Yearly 0% Only'
413         end
414         object Label24: TLabel
415           Left = 46
416           Top = 41
417           Width = 61
418           Height = 15
419           Caption = 'Uncertainty'
420         end
421         object Label34: TLabel
422           Left = 316
423           Top = 229
424           Width = 76
425           Height = 15
426           Caption = 'Single Size Cat'
427         end
428         object Label25: TLabel
429           Left = 45
430           Top = 252
431           Width = 75
432           Height = 15
433           Caption = 'Cohort Follow'
434         end
435         object Label9: TLabel
436           Left = 351
437           Top = 254
438           Width = 58
439           Height = 15
440           Caption = 'Debug Out'
441           Visible = False
442         end
443         object Label36: TLabel
444           Left = 25
445           Top = 175
446           Width = 71
447           Height = 15
448           Caption = 'Period Adjust'
449         end
450         object Label32: TLabel
```

```
451           Left = 295
452           Top = 142
453           Width = 84
454           Height = 15
455           Caption = 'Single Cost Year'
456           Visible = False
457         end
458         object Label21: TLabel
459           Left = 25
460           Top = 101
461           Width = 71
462           Height = 15
463           Caption = 'Tech Forecast'
464         end
465         object Label37: TLabel
466           Left = 45
467           Top = 229
468           Width = 218
469           Height = 15
470           Caption = 'Add additional Small/Large Size Category'
471         end
472         object Label20: TLabel
473           Left = 45
474           Top = 208
475           Width = 238
476           Height = 15
477           Caption = 'Use All Categories (false for all category only)'
478         end
479         object edDiscRate: TEdit
480           AlignWithMargins = True
481           Left = 424
482           Top = 15
483           Width = 41
484           Height = 21
485           Hint = '0 for max'
486           NumbersOnly = True
487           TabOrder = 0
488           Text = '0'
489           Visible = False
490           OnDblClick = edOccFileDblClick
491         end
492         object edDiscRates: TEdit
493           AlignWithMargins = True
494           Left = 239
495           Top = 15
496           Width = 67
497           Height = 21
498           Hint = 'Comma seperated'
499           TabOrder = 1
500           Text = '0'
501           OnDblClick = edOccFileDblClick
502         end
503         object cbYearlyOut: TCheckBox
504           Left = 157
505           Top = 141
506           Width = 25
507           Height = 17
508           TabOrder = 2
509         end
510         object cbZeroOnly: TCheckBox
511           Left = 22
512           Top = 142
513           Width = 25
514           Height = 17
515           TabOrder = 3
516         end
517         object cbUnc: TCheckBox
518           Left = 22
519           Top = 41
520           Width = 25
521           Height = 17
522           TabOrder = 4
523         end
524         object edSingleSize: TEdit
525           Left = 419
```

```
526           Top = 227
527           Width = 25
528           Height = 21
529           TabOrder = 5
530           Text = '0'
531         end
532         object cbCohortFollow: TCheckBox
533           Left = 21
534           Top = 252
535           Width = 25
536           Height = 17
537           TabOrder = 6
538         end
539         object cbDebug: TCheckBox
540           Left = 424
541           Top = 254
542           Width = 25
543           Height = 17
544           TabOrder = 7
545           Visible = False
546         end
547         object edPeriodAdjust: TEdit
548           AlignWithMargins = True
549           Left = 137
550           Top = 173
551           Width = 35
552           Height = 21
553           Hint = 'Double click to edit'
554           TabOrder = 8
555           Text = '0'
556         end
557         object edSingleCostYear: TEdit
558           AlignWithMargins = True
559           Left = 407
560           Top = 142
561           Width = 56
562           Height = 21
563           Hint = 'Double click to edit'
564           TabOrder = 9
565           Text = '0'
566           Visible = False
567         end
568         object edTechForecast: TEdit
569           AlignWithMargins = True
570           Left = 120
571           Top = 99
572           Width = 362
573           Height = 21
574           Hint = 'Double click to edit'
575           TabOrder = 10
576           Text = '***'
577           OnDblClick = edOccFileDblClick
578         end
579         object cbAltSize: TCheckBox
580           Left = 20
581           Top = 229
582           Width = 25
583           Height = 17
584           TabOrder = 11
585         end
586         object lbOUncFile: TLabeledEdit
587           Left = 120
588           Top = 64
589           Width = 360
590           Height = 21
591           EditLabel.Width = 64
592           EditLabel.Height = 21
593           EditLabel.Caption = 'Use Unc File'
594           LabelPosition = lpLeft
595           TabOrder = 12
596           Text = ''
597           OnDblClick = edOccFileDblClick
598         end
599         object cbAllCategories: TCheckBox
```

```
600           Left = 20
601           Top = 208
602           Width = 25
603           Height = 17
604           TabOrder = 13
605         end
606       end
607       object Panel20: TPanel
608         Left = 0
609         Top = 840
610         Width = 508
611         Height = 65
612         BevelOuter = bvNone
613         BorderStyle = bsSingle
614         TabOrder = 14
615         object Label23: TLabel
616           Left = 24
617           Top = 16
618           Width = 83
619           Height = 15
620           Caption = 'PWS Debug List'
621         end
622         object edDebugLIst: TMemo
623           AlignWithMargins = True
624           Left = 136
625           Top = 3
626           Width = 361
627           Height = 54
628           HideSelection = False
629           ScrollBars = ssVertical
630           TabOrder = 0
631           WordWrap = False
632         end
633       end
634       object Panel6: TPanel
635         Left = 0
636         Top = 907
637         Width = 508
638         Height = 50
639         BevelOuter = bvNone
640         BorderStyle = bsSingle
641         TabOrder = 5
642         object Label2: TLabel
643           Left = 24
644           Top = 13
645           Width = 43
646           Height = 15
647           Caption = 'Workers'
648         end
649         object Label7: TLabel
650           Left = 393
651           Top = 13
652           Width = 55
653           Height = 15
654           Caption = 'Quick Run'
655         end
656         object Label28: TLabel
657           Left = 137
658           Top = 13
659           Width = 51
660           Height = 15
661           Caption = 'CPU Mult'
662         end
663         object Label29: TLabel
664           Left = 267
665           Top = 13
666           Width = 40
667           Height = 15
668           Caption = 'Censor '
669         end
670         object edWorkers: TEdit
671           AlignWithMargins = True
672           Left = 85
673           Top = 11
674           Width = 41
```

```
675          Height = 21
676          Hint = '0 for max'
677          Enabled = False
678          NumbersOnly = True
679          TabOrder = 0
680          Text = '0'
681          OnDblClick = edOccFileDblClick
682        end
683        object cbQuickRun: TCheckBox
684          Left = 481
685          Top = 13
686          Width = 25
687          Height = 17
688          Enabled = False
689          TabOrder = 1
690        end
691        object edCPUMult: TEdit
692          AlignWithMargins = True
693          Left = 198
694          Top = 11
695          Width = 41
696          Height = 21
697          Hint = '0 for max'
698          Enabled = False
699          NumbersOnly = True
700          TabOrder = 2
701          Text = '0'
702          OnDblClick = edOccFileDblClick
703        end
704        object cbCensor: TCheckBox
705          Left = 355
706          Top = 13
707          Width = 25
708          Height = 17
709          Enabled = False
710          TabOrder = 3
711        end
712      end
713      object Panel2: TPanel
714        Left = 0
715        Top = 959
716        Width = 508
717        Height = 35
718        BevelOuter = bvNone
719        BorderStyle = bsSingle
720        TabOrder = 1
721        object Label4: TLabel
722          Left = 8
723          Top = 8
724          Width = 33
725          Height = 19
726          Caption = 'PFAS'
727          Font.Charset = DEFAULT_CHARSET
728          Font.Color = clWindowText
729          Font.Height = -14
730          Font.Name = 'Segoe UI'
731          Font.Style = [fsBold]
732          ParentFont = False
733        end
734      end
735      object Panel8: TPanel
736        Left = 0
737        Top = 996
738        Width = 508
739        Height = 80
740        BevelOuter = bvNone
741        BorderStyle = bsSingle
742        TabOrder = 7
743        object Label10: TLabel
744          Left = 134
745          Top = 14
746          Width = 83
747          Height = 15
748          Caption = 'PBPK Time Step'
```

```
749         end
750         object Label30: TLabel
751           Left = 8
752           Top = 14
753           Width = 80
754           Height = 15
755           Caption = 'VLS force Conc'
756         end
757         object Label33: TLabel
758           Left = 295
759           Top = 14
760           Width = 79
761           Height = 15
762           Caption = 'HazWaste Cost'
763         end
764         object Label38: TLabel
765           Left = 8
766           Top = 46
767           Width = 51
768           Height = 15
769           Caption = 'EJ Output'
770         end
771         object edPBPKTimeSTep: TEdit
772           AlignWithMargins = True
773           Left = 223
774           Top = 11
775           Width = 41
776           Height = 21
777           TabOrder = 0
778           Text = '30'
779           OnDblClick = edOccFileDblClick
780         end
781         object cbForcePFASConc: TCheckBox
782           Left = 101
783           Top = 13
784           Width = 25
785           Height = 17
786           TabOrder = 1
787         end
788         object cbHazWaste: TCheckBox
789           Left = 381
790           Top = 13
791           Width = 25
792           Height = 17
793           TabOrder = 2
794         end
795         object cbEJOutput: TCheckBox
796           Left = 101
797           Top = 45
798           Width = 25
799           Height = 17
800           TabOrder = 3
801         end
802       end
803       object Panel9: TPanel
804         Left = 0
805         Top = 1078
806         Width = 508
807         Height = 47
808         BevelOuter = bvNone
809         BorderStyle = bsSingle
810         TabOrder = 8
811         object Label11: TLabel
812           Left = 24
813           Top = 13
814           Width = 122
815           Height = 15
816           Caption = 'Start, End, Change Year'
817         end
818         object edStartYear: TEdit
819           AlignWithMargins = True
820           Left = 176
821           Top = 11
822           Width = 50
823           Height = 21
```

```
824          NumbersOnly = True
825          TabOrder = 0
826          Text = '2020'
827          OnDblClick = edOccFileDblClick
828        end
829        object edEndYear: TEdit
830          AlignWithMargins = True
831          Left = 257
832          Top = 11
833          Width = 50
834          Height = 21
835          NumbersOnly = True
836          TabOrder = 1
837          Text = '2050'
838          OnDblClick = edOccFileDblClick
839        end
840        object edChangeYear: TEdit
841          AlignWithMargins = True
842          Left = 336
843          Top = 11
844          Width = 50
845          Height = 21
846          NumbersOnly = True
847          TabOrder = 2
848          Text = '2023'
849          OnDblClick = edOccFileDblClick
850        end
851      end
852      object Panel23: TPanel
853        Left = 0
854        Top = 1127
855        Width = 508
856        Height = 35
857        BevelOuter = bvNone
858        BorderStyle = bsSingle
859        TabOrder = 16
860        object Label27: TLabel
861          Left = 8
862          Top = 8
863          Width = 102
864          Height = 19
865          Caption = 'Output Options'
866          Font.Charset = DEFAULT_CHARSET
867          Font.Color = clWindowText
868          Font.Height = -14
869          Font.Name = 'Segoe UI'
870          Font.Style = [fsBold]
871          ParentFont = False
872        end
873      end
874      object Panel22: TPanel
875        Left = 0
876        Top = 1164
877        Width = 508
878        Height = 65
879        BevelOuter = bvNone
880        BorderStyle = bsSingle
881        TabOrder = 15
882        object edOutputOptions: TCheckListBox
883          Left = 24
884          Top = 3
885          Width = 465
886          Height = 57
887          BevelOuter = bvNone
888          Ctl3D = False
889          ItemHeight = 15
890          Items.Strings = (
891            'GACTOC'
892            'GACTOCPOP')
893          ParentCtl3D = False
894          TabOrder = 0
895        end
896      end
897    end
```

```
898     end
899     object Panel10: TPanel
900       AlignWithMargins = True
901       Left = 3
902       Top = 3
903       Width = 1064
904       Height = 102
905       Align = alTop
906       BevelOuter = bvNone
907       BorderWidth = 1
908       BorderStyle = bsSingle
909       Ctl3D = False
910       ParentCtl3D = False
911       TabOrder = 1
912       ExplicitWidth = 1058
913       DesignSize = (
914         1062
915         100)
916       object gCPU: TGauge
917         Left = 971
918         Top = 3
919         Width = 80
920         Height = 40
921         Anchors = [akTop, akRight]
922         BackColor = clBtnFace
923         ForeColor = clMoneyGreen
924         Font.Charset = DEFAULT_CHARSET
925         Font.Color = clWindowText
926         Font.Height = -14
927         Font.Name = 'Segoe UI'
928         Font.Style = [fsBold]
929         ParentFont = False
930         Progress = 0
931         ExplicitLeft = 982
932       end
933       object Label12: TLabel
934         Left = 998
935         Top = 45
936         Width = 23
937         Height = 15
938         Anchors = [akTop, akRight]
939         Caption = 'CPU'
940         ExplicitLeft = 1009
941       end
942       object lbMemory: TLabel
943         Left = 739
944         Top = 14
945         Width = 45
946         Height = 15
947         Alignment = taRightJustify
948         Caption = '*********'
949       end
950       object lbTotTime: TLabel
951         Left = 739
952         Top = 38
953         Width = 45
954         Height = 15
955         Alignment = taRightJustify
956         Caption = '*********'
957       end
958       object Label31: TLabel
959         Left = 14
960         Top = 68
961         Width = 40
962         Height = 15
963         Caption = 'Label31'
964       end
965       object Button2: TButton
966         Left = 258
967         Top = 22
968         Width = 75
969         Height = 25
970         Anchors = []
971         Caption = 'Tests'
972         TabOrder = 0
```

```
973       Visible = False
974       OnClick = Button2Click
975       ExplicitLeft = 256
976     end
977     object Button4: TButton
978       Left = 176
979       Top = 22
980       Width = 75
981       Height = 25
982       Caption = 'Run'
983       TabOrder = 1
984       OnClick = Button4Click
985     end
986     object Button3: TButton
987       Left = 95
988       Top = 22
989       Width = 75
990       Height = 25
991       Caption = 'Save'
992       TabOrder = 2
993       OnClick = Button3Click
994     end
995     object Button1: TButton
996       Left = 14
997       Top = 22
998       Width = 75
999       Height = 25
1000      Caption = 'Load'
1001      TabOrder = 3
1002      OnClick = Button1Click
1003    end
1004    object lbRestart: TLabeledEdit
1005      Left = 517
1006      Top = 26
1007      Width = 122
1008      Height = 21
1009      EditLabel.Width = 36
1010      EditLabel.Height = 21
1011      EditLabel.Caption = 'Restart'
1012      LabelPosition = lpLeft
1013      TabOrder = 4
1014      Text = ''
1015      Visible = False
1016    end
1017    object lbStartAt: TLabeledEdit
1018      Left = 415
1019      Top = 26
1020      Width = 42
1021      Height = 21
1022      EditLabel.Width = 39
1023      EditLabel.Height = 21
1024      EditLabel.Caption = 'Start At'
1025      LabelPosition = lpLeft
1026      TabOrder = 5
1027      Text = ''
1028      Visible = False
1029    end
1030  end
1031  object pnlMain: TPanel
1032    AlignWithMargins = True
1033    Left = 575
1034    Top = 111
1035    Width = 492
1036    Height = 727
1037    Align = alClient
1038    BevelOuter = bvNone
1039    BorderStyle = bsSingle
1040    Ctl3D = False
1041    ParentCtl3D = False
1042    TabOrder = 2
1043    ExplicitWidth = 486
1044    ExplicitHeight = 718
1045    object pnlGauge: TPanel
1046      AlignWithMargins = True
```

```
1047        Left = 3
1048        Top = 3
1049        Width = 484
1050        Height = 97
1051        Align = alTop
1052        BevelOuter = bvNone
1053        TabOrder = 0
1054        ExplicitWidth = 478
1055        object Label15: TLabel
1056          Left = 8
1057          Top = 16
1058          Width = 34
1059          Height = 19
1060          Caption = 'Step:'
1061          Font.Charset = DEFAULT_CHARSET
1062          Font.Color = clWindowText
1063          Font.Height = -14
1064          Font.Name = 'Segoe UI'
1065          Font.Style = [fsBold]
1066          ParentFont = False
1067        end
1068        object Label16: TLabel
1069          Left = 8
1070          Top = 44
1071          Width = 32
1072          Height = 19
1073          Caption = 'Msg:'
1074          Font.Charset = DEFAULT_CHARSET
1075          Font.Color = clWindowText
1076          Font.Height = -14
1077          Font.Name = 'Segoe UI'
1078          Font.Style = [fsBold]
1079          ParentFont = False
1080        end
1081        object Label17: TLabel
1082          Left = 8
1083          Top = 72
1084          Width = 37
1085          Height = 19
1086          Caption = 'Time:'
1087          Font.Charset = DEFAULT_CHARSET
1088          Font.Color = clWindowText
1089          Font.Height = -14
1090          Font.Name = 'Segoe UI'
1091          Font.Style = [fsBold]
1092          ParentFont = False
1093        end
1094        object lbTime: TLabel
1095          Left = 48
1096          Top = 72
1097          Width = 18
1098          Height = 19
1099          Caption = '***'
1100          Font.Charset = DEFAULT_CHARSET
1101          Font.Color = clWindowText
1102          Font.Height = -14
1103          Font.Name = 'Segoe UI'
1104          Font.Style = []
1105          ParentFont = False
1106        end
1107        object lbMsg: TLabel
1108          Left = 48
1109          Top = 44
1110          Width = 18
1111          Height = 19
1112          Caption = '***'
1113          Font.Charset = DEFAULT_CHARSET
1114          Font.Color = clWindowText
1115          Font.Height = -14
1116          Font.Name = 'Segoe UI'
1117          Font.Style = []
1118          ParentFont = False
1119        end
1120        object lbStep: TLabel
1121          Left = 48
```

```
1122        Top = 16
1123        Width = 18
1124        Height = 19
1125        Caption = '***'
1126        Font.Charset = DEFAULT_CHARSET
1127        Font.Color = clWindowText
1128        Font.Height = -14
1129        Font.Name = 'Segoe UI'
1130        Font.Style = []
1131        ParentFont = False
1132      end
1133      object lbProgress: TLabel
1134        Left = 358
1135        Top = 36
1136        Width = 29
1137        Height = 28
1138        Caption = '0%'
1139        Font.Charset = DEFAULT_CHARSET
1140        Font.Color = clWindowText
1141        Font.Height = -20
1142        Font.Name = 'Segoe UI'
1143        Font.Style = [fsBold]
1144        ParentFont = False
1145      end
1146    end
1147    object Panel11: TPanel
1148      Left = 0
1149      Top = 103
1150      Width = 490
1151      Height = 622
1152      Align = alClient
1153      BevelOuter = bvNone
1154      TabOrder = 1
1155      ExplicitWidth = 484
1156      ExplicitHeight = 613
1157      object Panel12: TPanel
1158        AlignWithMargins = True
1159        Left = 3
1160        Top = 3
1161        Width = 484
1162        Height = 30
1163        Align = alTop
1164        Alignment = taLeftJustify
1165        BevelOuter = bvNone
1166        Caption = 'Run Info'
1167        Font.Charset = DEFAULT_CHARSET
1168        Font.Color = clWindowText
1169        Font.Height = -14
1170        Font.Name = 'Segoe UI'
1171        Font.Style = [fsBold]
1172        ParentFont = False
1173        TabOrder = 0
1174        ExplicitWidth = 478
1175      end
1176      object memLog: TMemo
1177        AlignWithMargins = True
1178        Left = 3
1179        Top = 39
1180        Width = 484
1181        Height = 580
1182        Align = alClient
1183        BevelOuter = bvNone
1184        Font.Charset = DEFAULT_CHARSET
1185        Font.Color = clWindowText
1186        Font.Height = -10
1187        Font.Name = 'Cascadia Code'
1188        Font.Style = []
1189        Lines.Strings = (
1190          '')
1191        ParentFont = False
1192        ScrollBars = ssVertical
1193        TabOrder = 1
1194        WordWrap = False
1195        ExplicitWidth = 478
```

```
1196         ExplicitHeight = 571
1197       end
1198     end
1199   end
1200   object OD: TOpenDialog
1201     Filter = 'Config|*.ini|CSV Files|*.csv'
1202     FilterIndex = 2
1203     Left = 760
1204     Top = 133
1205   end
1206   object SD: TSaveDialog
1207     Filter = 'Config|*.ini'
1208     Left = 712
1209     Top = 136
1210   end
1211   object ch: TTimer
1212     OnTimer = chTimer
1213     Left = 827
1214     Top = 139
1215   end
1216 end
1217
```

wndMain.pas

```
1  unit wndMain;
2
3  interface
4
5  uses
6    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
7    Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls, SafewaterConfig,
8    Vcl.WinXPanels, Vcl.CheckLst, StrUtils, Vcl.Samples.Gauges, System.Threading,
9    Vcl.ComCtrls, Vcl.Mask, CompiledCost;
10
11 type
12
13   TControlProc = reference to procedure (const AControl: TControl);
14
15   TwMain = class(TForm)
16     OD: TOpenDialog;
17     ScrollBox1: TScrollBox;
18     CfgPanel: TStackPanel;
19     Panel1: TPanel;
20     Label3: TLabel;
21     Panel2: TPanel;
22     Label4: TLabel;
23     edOccFile: TEdit;
24     Panel3: TPanel;
25     Label5: TLabel;
26     edSdwisFiles: TMemo;
27     Panel4: TPanel;
28     Label1: TLabel;
29     edCostOptions: TCheckListBox;
30     Panel5: TPanel;
31     Label6: TLabel;
32     edBenefitOptions: TCheckListBox;
33     Panel6: TPanel;
34     Label2: TLabel;
35     edWorkers: TEdit;
36     Panel7: TPanel;
37     Label8: TLabel;
38     Panel8: TPanel;
39     Label10: TLabel;
40     edPBPKTimeSTep: TEdit;
41     Panel9: TPanel;
42     Label11: TLabel;
43     edStartYear: TEdit;
44     edEndYear: TEdit;
45     edChangeYear: TEdit;
46     SD: TSaveDialog;
47     Panel10: TPanel;
48     Button2: TButton;
```

```
    Button4: TButton;
    Button3: TButton;
    Button1: TButton;
    gCPU: TGauge;
    Label12: TLabel;
    pnlMain: TPanel;
    ch: TTimer;
    lbMemory: TLabel;
    pnlGauge: TPanel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    lbTotTime: TLabel;
    lbTime: TLabel;
    lbMsg: TLabel;
    lbStep: TLabel;
    Panel11: TPanel;
    Panel12: TPanel;
    memLog: TMemo;
    lbProgress: TLabel;
    Panel13: TPanel;
    Label13: TLabel;
    edBWFile: TEdit;
    Panel14: TPanel;
    Label14: TLabel;
    edConsFile: TEdit;
    Panel15: TPanel;
    Label18: TLabel;
    edPWSSPFile: TEdit;
    Panel16: TPanel;
    Label19: TLabel;
    edCountySPFile: TEdit;
    Panel19: TPanel;
    Label22: TLabel;
    edDiscRate: TEdit;
    Panel20: TPanel;
    Label23: TLabel;
    edDebugLIst: TMemo;
    cbQuickRun: TCheckBox;
    Label7: TLabel;
    Panel22: TPanel;
    edOutputOptions: TCheckListBox;
    Panel23: TPanel;
    Label27: TLabel;
    edCPUMult: TEdit;
    Label28: TLabel;
    Label29: TLabel;
    cbCensor: TCheckBox;
    Label30: TLabel;
    cbForcePFASConc: TCheckBox;
    lbRestart: TLabeledEdit;
    lbStartAt: TLabeledEdit;
    Label31: TLabel;
    cbHazWaste: TCheckBox;
    Label33: TLabel;
    edDiscRates: TEdit;
    Label26: TLabel;
    cbEJOutput: TCheckBox;
    Label38: TLabel;
    cbYearlyOut: TCheckBox;
    cbZeroOnly: TCheckBox;
    cbUnc: TCheckBox;
    Label39: TLabel;
    Label35: TLabel;
    Label24: TLabel;
    edSingleSize: TEdit;
    Label34: TLabel;
    cbCohortFollow: TCheckBox;
    Label25: TLabel;
    cbDebug: TCheckBox;
    Label9: TLabel;
    edPeriodAdjust: TEdit;
    Label36: TLabel;
    edSingleCostYear: TEdit;
    Label32: TLabel;
```

```
    edTechForecast: TEdit;
    Label21: TLabel;
    Label37: TLabel;
    cbAltSize: TCheckBox;
    lbOUncFile: TLabeledEdit;
    Label20: TLabel;
    cbAllCategories: TCheckBox;
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure edOccFileDblClick(Sender: TObject);
    procedure edSdwisFilesDblClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure chTimer(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure lbOUncFileSubLabelDblClick(Sender: TObject);
  private
    { Private declarations }
    LockUpdate: TObject;
    PeakMemory: double;
    StartTime, StepStartTime, StepTime: int64;
    HoldHeight: integer;
    LastStep: string;
    Running: boolean;
    procedure ModifyControl(const AControl: TControl;
          const ARef: TControlProc);
    procedure GUIProgress(const kind,progress: integer; const step,msg: string);
  public
    { Public declarations }
    _Cfg: TSafeWaterConfig;
    function FTime(const c: integer): string;
    procedure UIToCfg(cfg: TSafeWaterConfig);
    procedure CfgToUI(cfg: TSafeWaterConfig);
  end;

var
  wMain: TwMain;

implementation

{$R *.dfm}

uses wndTests, SafewaterUtility, SafewaterGlobals, SafewaterModel;

procedure TwMain.Button1Click(Sender: TObject);
begin
  OD.InitialDir := CfgPath;
  OD.Options := OD.Options - [ofAllowMultiSelect];
  if not OD.Execute() then exit;
  _cfg.LoadFromFile(OD.FileName);
  //always make config.ini last loaded
  _cfg.SaveToFile(apppath+'config.ini');
  CfgToUI(_cfg);
end;

procedure TwMain.Button2Click(Sender: TObject);
begin
  wTests.ShowModal;
end;

procedure TwMain.Button3Click(Sender: TObject);
begin
  SD.InitialDir := CfgPath;
  SD.FilterIndex := 1;
  if not SD.Execute() then exit;
  UIToCfg(_cfg);
  _cfg.SaveToFile(apppath+'config.ini');
  _cfg.SaveToFile(changefileext(SD.FileName,'.ini'));
end;

procedure TwMain.Button4Click(Sender: TObject);
begin
```

```
198   var s: string :='';
199   UiToCfg(_Cfg);
200   if not InputQuery('Name/Path to results..', 'Enter name (no ext, no spaces)', s) then exit;
201   //TODO check S for legal filename
202
203   _Cfg.SaveToFile(apppath+'config.ini');
204   var fn := s+'.ini';
205   _Cfg.SaveToFile(cfgpath+fn);
206
207   PeakMemory := 0;
208   pnlGauge.Height := HoldHeight;
209   StartTime := GetTickCount;
210   Running := True;
211   TTask.Run(procedure
212     begin
213       var T := TSafeWaterModel.create;
214       T.GUIProc := GUIProgress;
215       T.Initialize(s,CfgPath+fn);
216       T.Run;
217       T.Free;
218       pnlGauge.Height := 1;
219       Running:=False;
220       ProcMsg();
221     end);
222
223 end;
224
225 procedure TwMain.CfgToUI(cfg: TSafeWaterConfig);
226 begin
227   edOccFile.Text := cfg.OccFile;
228   edBWFile.Text := cfg.BWFile;
229   edConsFile.Text := cfg.ConsumptionFile;
230   edPWSSPFile.Text := cfg.PWSSubpopFIle;
231   edCountySPFile.Text := cfg.CountySubpopfile;
232   edTechForecast.Text := cfg.TechForecastfile;
233   lbOUncFile.Text := cfg.OverrideUnc;
234   edSingleSize.Text := cfg.SingleSystemCat.ToString;
235
236   edSDWISFIles.Lines.Clear;
237   for var s in cfg.PWSFiles do
238     edSDWISFIles.Lines.Add(s);
239
240   edCostOptions.CheckAll(cbUnchecked);
241   for var s in cfg.CostOptions do begin
242     var i := edCostOptions.Items.IndexOf(s);
243     if i>-1 then edCostOptions.Checked[i] := true;
244   end;
245   edBenefitOptions.CheckAll(cbUnchecked);
246   for var s in cfg.BenefitOptions do begin
247     var i := edBenefitOptions.Items.IndexOf(s);
248     if i>-1 then edBenefitOptions.Checked[i] := true;
249   end;
250   edOutputOptions.CheckAll(cbUnchecked);
251   for var s in cfg.OutputOptions do begin
252     var i := edOutputOptions.Items.IndexOf(s);
253     if i>-1 then edOutputOptions.Checked[i] := true;
254   end;
255
256   cbZeroOnly.Checked := cfg.ZeroDiscYearly;
257   cbAltSize.Checked := cfg.SmallLargeCat;
258   cbUnc.Checked := cfg.Uncertainty;
259   cbDebug.Checked := cfg.DebugOut;
260   edDebugList.Lines.CommaText := cfg.DebugList;
261   cbQuickRun.Checked := cfg.QuickRun;
262   cbCohortFollow.Checked := cfg.ICFCF;
263   cbCensor.Checked := cfg.Censor;
264   cbForcePFASCOnc.Checked := cfg.ForcePFASCOnc;
265   cbHazWaste.Checked := cfg.PFASCostHW;
266   cbEJOutput.Checked := cfg.PFASEJOutput;
267   cbYearlyOut.Checked := cfg.YearlyOut;
268   cbAllCategories.Checked := cfg.AllCategories;
269
270   edWorkers.Text := cfg.Workers.ToString;
271   edCPUMult.Text := cfg.CPUMult.ToString;
272   edDiscRate.Text := cfg.DiscountRate.ToString;
```

```
273   edDiscRates.Text := cfg.DiscountRates;
274   edPBPKTimeSTep.Text := cfg.PBPKTimeStep.ToString;
275   edStartYear.Text := cfg.StartYear.ToString;
276   edEndYear.Text := cfg.EndYear.ToString;
277   edChangeYear.Text := cfg.PFASChangeYear.ToString;
278   edSingleCostYear.Text := cfg.SingleCostYear.ToString;
279   edPeriodAdjust.Text := cfg.PeriodAdjust.ToString;
280 end;
281
282 procedure TwMain.edOccFileDblClick(Sender: TObject);
283 begin
284   OD.FilterIndex := 2;
285   OD.Options := OD.Options - [ofAllowMultiSelect];
286   OD.InitialDir := DataPath;
287   if not OD.Execute then exit;
288   var s: string := OD.FileName;
289   RootToCode(s);
290   (Sender as TEdit).Text := s;
291 end;
292
293 procedure TwMain.edSdwisFilesDblClick(Sender: TObject);
294 begin
295   OD.Options := OD.Options + [ofAllowMultiSelect];
296   OD.InitialDir := DataPath;
297   OD.FilterIndex := 2;
298   if not OD.Execute then exit;
299   edSdwisFiles.Clear;
300   for var s in OD.Files do begin
301     var ss:string := s;
302     RootToCode(ss);
303     edSdwisFiles.Lines.add(ss);
304   end;
305 end;
306
307 procedure TwMain.FormCreate(Sender: TObject);
308 begin
309   Label31.Caption:=_CWorkBookOPTION;
310   SetPaths();
311   ImInGUI := True;
312   LockUpdate := TObject.Create;
313   _Cfg:=TSafeWaterConfig.create;
314   _Cfg.LoadFromFile(AppPath+'config.ini');
315   CfgToUI(_Cfg);
316   Caption := 'Safewater Version: '+swVersion+' Date: '+swDate;
317   {$IFDEF DEBUG}
318     Button2.Visible := true;
319   {$ENDIF}
320 end;
321
322 procedure TwMain.FormDestroy(Sender: TObject);
323 begin
324   _Cfg.Free;
325   LockUpdate.Free;
326 end;
327
328 procedure TwMain.FormShow(Sender: TObject);
329 begin
330   HoldHeight := pnlGauge.Height;
331   pnlGauge.Height := 1;
332 end;
333
334 function TwMain.FTime(const c: integer): string;
335 begin
336   var H:=c div 3600000;
337   var M := c mod 3600000 div 60000;
338   var S := c mod 3600000 mod 60000 div 1000;
339   Result := Format('%.2d:%.2d:%.2d',[h,m,s]);
340 end;
341
342 procedure TwMain.GUIProgress(const kind,progress: integer; const step,msg: string);
343 begin
344   System.TMonitor.Enter(LockUpdate);
345   try
346     if Kind = 0 then begin
```

```
347       memLog.Lines.Add(msg);
348     end else
349     if Kind = 1 then begin
350       lbStep.Caption := step;
351       lbMsg.Caption := msg;
352       lbProgress.Caption := inttostr(progress)+'%';
353       if LastStep<>step then begin
354         StepStartTime := gettickcount;
355         LastStep:=Step;
356       end;
357       lbTime.Caption := FTime(gettickcount - StepStartTime);
358     end;
359     Application.ProcessMessages;
360   finally
361     System.TMonitor.Exit(LockUpdate);
362   end;
363 end;
364
365
366 procedure TwMain.lbOUncFileSubLabelDblClick(Sender: TObject);
367 begin
368   if not OD.Execute then exit;
369   lbOUncFile.Text:=OD.FileName;
370 end;
371
372 procedure TwMain.ModifyControl(const AControl: TControl;
373   const ARef: TControlProc);
374 var
375   i : Integer;
376 begin
377   if AControl=nil then
378     Exit;
379   if AControl is TWinControl then begin
380     for i := 0 to TWinControl(AControl).ControlCount-1 do
381       ModifyControl(TWinControl(AControl).Controls[i], ARef);
382   end;
383   ARef(AControl);
384 end;
385
386 procedure TwMain.chTimer(Sender: TObject);
387 begin
388   var m := GetMemoryUsed;
389   var mu := m / (1024*1024*1024);
390   if (mu>PeakMemory) then PeakMemory:=mu;
391   lbMemory.Caption := Format('Memory (Current,Max): %.f3GB, %.f3GB',[mu,PeakMemory]);
392   gCPU.Progress := GetProcessorUsage;
393   lbTotTime.Caption := '';
394   if Running then
395     lbTotTime.Caption := 'Time: ' + FTIme(GetTickCount - StartTime);
396   Application.ProcessMessages;
397 end;
398
399 procedure TwMain.UIToCfg(cfg: TSafeWaterConfig);
400 begin
401   cfg.OccFile := edOccFile.Text;
402   cfg.BWFile := edBWFile.Text;
403   cfg.ConsumptionFile := edConsFile.Text;
404   cfg.PWSSubpopFIle := edPWSSPFile.Text;
405   cfg.CountySubpopfile := edCountySPFile.Text;
406   cfg.TechForecastfile := edTechForecast.Text;
407   cfg.PWSFiles := SplitString(edSDWISFIles.Lines.CommaText,',');
408   cfg.SingleSystemCat := strtoint(edSingleSize.Text);
409   cfg.PeriodAdjust:=strtoint(edPeriodAdjust.Text);
410
411   var l: string := '';
412   for var i := 0 to edCostOptions.Items.Count-1 do begin
413     if edCostOptions.Checked[i] then begin
414       if l<>'' then l:=l+',';
415       l := l + edCostOptions.Items[i];
416     end;
417   end;
418   cfg.CostOptions := SplitString(l,',');
419
420   l:='';
421   for var i := 0 to edBenefitOptions.Items.Count-1 do begin
```

```
    if edBenefitOptions.Checked[i] then begin
      if l<>'' then l:=l+',';
      l := l + edBenefitOptions.Items[i];
    end;
  end;
  cfg.BenefitOptions := SplitString(l,',');

  l:='';
  for var i := 0 to edOutputOptions.Items.Count-1 do begin
    if edOutputOptions.Checked[i] then begin
      if l<>'' then l:=l+',';
      l := l + edOutputOptions.Items[i];
    end;
  end;
  cfg.OutputOptions := SplitString(l,',');

  cfg.ZeroDiscYearly := cbZeroOnly.Checked;
  cfg.SmallLargeCat := cbAltSize.Checked;
  cfg.Uncertainty := cbUnc.Checked;
  cfg.DebugOut := cbDebug.Checked;
  cfg.DebugList := edDebugList.Lines.CommaText;
  cfg.QuickRun := cbQuickRun.Checked;
  cfg.Censor := cbCensor.Checked;
  cfg.ForcePFASCOnc := cbForcePFASCOnc.Checked;
  cfg.PFASCostHW := cbHazWaste.Checked;
  cfg.PFASEJOutput := cbEJOutput.Checked;
  cfg.YearlyOut := cbYearlyOut.Checked;
  cfg.AllCategories := cbAllCategories.Checked;

  cfg.Workers := strtoint(edWorkers.Text);
  cfg.CPUMult := strtoint(edCPUMult.Text);
  cfg.DiscountRate := strtofloat(edDiscRate.Text);
  cfg.DiscountRates := edDiscRates.Text;
  cfg.PBPKTimeStep := strtofloat(edPBPKTimeSTep.Text);
  cfg.StartYear := strtoint(edStartYear.Text);
  cfg.EndYear := strtoint(edEndYear.Text);

  cfg.PFASChangeYear := strtoint(edChangeYear.Text);
  cfg.ICFCF := cbCohortFollow.Checked;
  cfg.SingleCostYear := strtoint(edSingleCostYear.Text);

  cfg.StartAt:=0;
  cfg.OverrideUnc:='';
  cfg.OverrideGUID:=lbRestart.Text;
  cfg.OverrideUnc:=lbOUncFile.Text;
  if cfg.OverrideGUID>'' then begin
    cfg.StartAt:=1;

  end;
end;

end.
```