```
unit LCRBenefits;

interface

uses Windows, SysUtils, Classes,
  LCRConfig, LCRPWSRecords, LCRCosts, LCRGlobals,
  LCRMetricCollector, LCRResultsFile, SafewaterUncertBucket,
  Math, CodeSiteLogging, Generics.Collections;

type
  TBenType = (btVoluntary, btMandatory, btRequested);

  TLCRBenYears = class
    Year: array [1 .. 150] of double;
  end;

  TLCRBenByYear = class
    fTotYears: integer;
    fOutName: string;
    fBenYears: TObjectDictionary<string, TLCRBenYears>;
    Active: boolean;

    constructor create(aConfig: TLCRConfig; aName: string);
    procedure AddBen(aName: string);
    procedure UpdateBen(Yr: integer; const aName: string; aValue: double);
    procedure SaveResults;
    destructor Destroy; override;
  end;

  TBenRec = record
    C,L,POU,TPOU : double;
    Dummy: double;
    function Total() : double;
    procedure Abs();
    procedure AddValue(const v: double; const cat: Integer); // add to one field
base on cat
    procedure DiscountAndAddValue(const v: double; const cat: Integer; const DR:
double; const Yrs: integer); // add to one field base on cat
    procedure Multiply(const v: double);
    procedure Subtract(const r: TBenRec);
    procedure DivideBy(const v: double);
    procedure CalcAnnual(const DR: double; const r: TBenRec; const Yrs: integer);
    procedure CalcAnnualInPlace(const DR: double; const Yrs: integer);
    procedure AddMetrics(Outputs: TMetricList; Name,Option: string; DR: double;
IsDollar: boolean);
  end;

  TLCRBenefits = class
  private
    fConfig: TLCRConfig;
```

```
    fOutputs: TMetricList;
    fDummyProb: double;
    fYearsOfOutput: integer;
    fPWSID: string;
    fLowHigh, fRunVersion: integer;
    fBBYY: TLCRBenByYear;

    dYr, dA, dS: integer;
    bp1, bp2: integer;

    PWSBenefits: double;

    function CorrectBL(const BL: double; const Age, Sex: integer): double;
    function FCVD(const BL1, BL2: double; const Age, Sex: integer): double;
    function FADHD(const BL1, BL2: double): double;
    function FLBW(const BL1, BL2: double): double;
    function FIQLoss(const BL1, BL2: double): double;

    procedure AvgBL(const NewYears,Sex,Age,PBWin,FromBin,ToBin : integer; out
FBL,TBL: double);
    procedure SnapShot;
    procedure SnapDiff;
  public
    DummyProb, BenPop, Ben7, BenA, Ben11, Ben0: double;
    _IQLoss, _IQLossVal: TBenRec;
    _ADHD, _ADHDVal: TBenRec;
    _LBW, _LBWVal: TBenRec;
    _CVD, _CVDVal: TBenRec;

    //used for debugging
    _IQLossB, _IQLossValB: TBenRec;
    _ADHDB, _ADHDValB: TBenRec;
    _LBWB, _LBWValB: TBenRec;
    _CVDB, _CVDValB: TBenRec;

    IQLoss,CVD,ADHD,LBW: double;
    IQLossVal, CVDVal, TotBen, ADHDVal, LBWVal: double;


    // PBBlood by sex,age,bin
    BL: array [1 .. 2, 0 .. 80, 1 .. 32] of double;
    CVDRate: array [1 .. 2, 4 .. 8] of double;
    VSL, IQPointVal, ADHDCaseVal: double;

    FinalBins: array [1 .. 32] of double;
    DoDebugOUt, GotOne: boolean;

    UncertaintyVars: TUncertaintyStudy; // pointer to model level var - must be set

    procedure CalcBinMove(const FromBin, ToBin, Yr: integer; const Pop: double;
```

```
const CCT: boolean;
      BenType: TBenType; Proxy: boolean; AdhocDebug: boolean = false);

    constructor create(aConfig: TLCRConfig; aOutputMetrics: TMetricList;
      Uncertainty: TUncertaintyStudy; aOption: string);
    destructor Destroy; override;

    procedure CalcBenefitsNew(const A: TYearlyMovementMicro; const Wgt: double;
      const DoDebug: boolean; const aCosts: TLCRCosts; const P90CCT, P90LSL: array
of double;
      abp1, abp2: integer; SmallSystem: boolean; ProxySystem: boolean; const
OptionName: string);

  end;

  TBenefitsCollector = class
  private
    fConfig: TLCRConfig;
    fOutputs: TMetricList;
    fUncertainty: TUncertaintyStudy;
    fDummyProb: double;
    fLowHigh: integer;

    BenefitsBaseline: TLCRBenefits;
    BenefitsOption: TLCRBenefits;

    bp1, bp2: integer;
  public
    DummyProb, BenPop, Ben7, BenA, Ben11, Ben0: double;
    _IQLoss, _IQLossVal: TBenRec;
    _ADHD, _ADHDVal: TBenRec;
    _LBW, _LBWVal: TBenRec;
    _CVD, _CVDVal: TBenRec;

    IQLoss,CVD,ADHD,LBW: double;
    IQLossVal, CVDVal, TotBen, ADHDVal, LBWVal: double;

    EndingBins, StartingBins, FinalBins, EndingBinsCheck: array [1 .. 32] of double;
    BinMovements: TMovementMicro;

    DoDebugOUt: boolean;
    NewBenBins: boolean;

    procedure GenerateBenefits(aCosts: TLCRCosts; ProxySystem: boolean);
    constructor create(aConfig: TLCRConfig; aOutputMetrics: TMetricList;
      aUncertainty: TUncertaintyStudy);
    destructor Destroy; override;
  end;

implementation
```

```
var
  PBF, PBM, PBSens1, PBSens2, PBSens3: string;
  // just squeezing in blood leads for women of child bearing age here,...
  // from 6/23/23 sheet from Meghan
  BLCBA: array [1 .. 32] of double = (
    1.728,1.171,0.735,1.262,
    1.262,1.262,0.959,0.959,
    0.959,0.735,0.735,0.735,
    0.998,0.839,0.735,0.735,
    0.735,0.735,0.735,0.735,
    0.735,0.735,0.735,0.735,
    0.735,0.735,0.735,0.735,
    0.735,0.735,0.735,0.735
  );
  // Income change from 1 iq point change in 2016 dollars) - Matt L email 7/29/2020
  IncomeDeltaAge: array [0 .. 80] of double = (
    0,0,0,0,0,0,0,0,0,0,
    -8,-3,-7,-4,-18,-36,75,53,-1891,-2685,874,1704,1405,
    1974,4243,6387,7682,8524,9431,10206,10774,11808,12315,13129,
    13778,14872,15658,16184,16781,17367,17788,18614,18865,19454,
    19958,20525,21185,21407,21926,22241,22273,22865,22810,23029,22782,
    22484,22174,21640,21131,20339,18978,17958,16313,14295,12830,10713,
    8979,7425,6329,5370,4437,3678,3287,2875,2596,2212,1952,1703,1525,1288,1071
  );

function Discount(const Value: double; const Yrs: integer; const Rate: double):
double;
begin
  Result := Value / intpower((1 + Rate), Yrs);
end;

function Annualize(const DiscRate, Value, Years: double): double;
begin
  Result := Value * (DiscRate / (1 - Power((1 + DiscRate), -Years)));
end;

{ TLCRBenefits }

procedure TLCRBenefits.AvgBL(const NewYears, Sex,Age,PBWin,FromBin,ToBin : integer;
out FBL,TBL: double);
begin
  FBL:=0;
  TBL:=0;
  var ti := 0;
  var fi := 0;
  if (NewYears > PBWin) or (NewYears>=Age) then begin
    fi := max(0, Age - PBWin);
    while (fi <= Age) do begin
      TBL := TBL + BL[Sex, fi, ToBin];
```

```
      FBL := FBL + BL[Sex, fi, FromBin];
      inc(ti);
      inc(fi);
    end;
  end else begin
    // find old vs new blood lead
    var Old := Age - NewYears;
    var StartAge := max(0, Age - PBWin);
    var InOld := 0;

    for fi := StartAge to Age  do begin
      if fi > Old then
        TBL := TBL + BL[Sex, fi, ToBin]
      else
        TBL := TBL + BL[Sex, fi, FromBin];
      FBL := FBL + BL[Sex, fi, FromBin];
      inc(ti);
    end;

  end;

   if ti > 0 then begin
     TBL := TBL / ti;
     FBL := FBL / ti;
   end else begin
     Codesite.Send('error bl avg:', age);
   end;

end;

procedure TLCRBenefits.CalcBinMove(const FromBin, ToBin, Yr: integer; const Pop:
double;
  const CCT: boolean; BenType: TBenType; Proxy: boolean; AdhocDebug: boolean =
false);
var
  FBL, TBL, CPop, iFBL, iTBL, tmpPop: double;
  valLBW1, valLBW2: double;
  vIQ, vCVD, vADHD, vLBW: double;
  Counted40: boolean;
  CountPop: double;

begin
  var PBWin := 10; //TODO should be in config
  BenPop := BenPop + Pop;
  FinalBins[ToBin] := FinalBins[ToBin] + Pop;

  var StartYear := Yr;
  var NewYears := 0;

  // Change to delay CCT moves by 2 years 08/25/20
```

```
  if CCT then
     StartYear := StartYear + 2;

  var ok := not GotOne;
  for var y:= StartYear to fConfig.YearsOfOutput do begin
    PWSBenefits := 0;
    for var S := 1 to 2 do begin
      for var A := 0 to 80 do begin
        CPop := Pop * fConfig.DefaultPopPct[S, A];
        NewYears := Y - StartYear;
        //blood lead average over window (and instant values)
        if FromBin > 16 then begin //hack for temporary filters
          iTBL := BL[S, A, FromBin];
          iFBL := BL[S, A, ToBin];
          AvgBL(NewYears,S,A,PBWin,ToBin,FromBin,FBL,TBL);
        end else begin
          iTBL := BL[S, A, ToBin];
          iFBL := BL[S, A, FromBin];
          // get averaged blood leads...
          AvgBL(NewYears,S,A,PBWin,FromBin,ToBin,FBL,TBL);
        end;

        //IQ
        if A = 6 then begin
          vIQ := FIQLoss(FBL, TBL) * CPop;
          if vIQ > 0 then begin
            Ben7 := Ben7 + CPop;
          end;
          if FromBin > 16 then vIQ := -vIQ;

          _IQLoss.AddValue(vIQ,trunc(GMoveBinLC[FromBin,ToBin]));
          _IQLossVal.DiscountAndAddValue(vIQ *
IQPointVal,trunc(GMoveBinLC[FromBin,ToBin]),fConfig.DiscountRate,y);

          PWSBenefits := PWSBenefits + vIQ;
        end;

        // low birth weight
        const escalator = 1.2032;  // Go from 2016-2022
        if (A = 0) and (y > StartYear) then begin
          // counting newborns in every year of analysis. After first year
          tmpPop := Pop * fConfig.DefaultPopPct[S, A];

          vLBW := FLBW(BLCBA[FromBin], BLCBA[ToBin]) / 20;
          if FromBin > 16 then vLBW := -vLBW;

          // leaving this like this so it is clear what is going on....
          valLBW1 := 1519.3 * vLBW * 0.3 / 100 * tmpPop + // 2.5
            1139.26 * vLBW * 0.3 / 100 * tmpPop + // 3
            958.54 * vLBW * 0.5 / 100 * tmpPop + // 3.3
```

```
            640.49 * vLBW * 0.9 / 100 * tmpPop + // 4
            480.36 * vLBW * 1.3 / 100 * tmpPop + // 4.5
            360.2 * vLBW * 2.4 / 100 * tmpPop + // 5
            14.86 * vLBW * 4.1 / 100 * tmpPop + // 5.5
            14.61 * vLBW * 13.5 / 100 * tmpPop + // 6
            14.12 * vLBW * 33.2 / 100 * tmpPop + // 7
            13.66 * vLBW * 29.4 / 100 * tmpPop; // 8

          valLBW2 := 0.00 * vLBW * 0.3 / 100 * tmpPop + // 2.5
            593.17 * vLBW * 0.3 / 100 * tmpPop + // 3
            469.72 * vLBW * 0.5 / 100 * tmpPop + // 3.3
            271.85 * vLBW * 0.9 / 100 * tmpPop + // 4
            183.57 * vLBW * 1.3 / 100 * tmpPop + // 4.5
            123.74 * vLBW * 2.4 / 100 * tmpPop + // 5
            16.04 * vLBW * 4.1 / 100 * tmpPop + // 5.5
            14.34 * vLBW * 13.5 / 100 * tmpPop + // 6
            11.45 * vLBW * 33.2 / 100 * tmpPop + // 7
            9.12 * vLBW * 29.4 / 100 * tmpPop; // 8

          if vLBW > 0 then begin
            Ben0 := Ben0 + tmpPop;
          end;

          _LBW.AddValue(vLBW * 20 * tmpPop,trunc(GMoveBinLC[FromBin,ToBin]));
          _LBWVal.DiscountAndAddValue(escalator *(valLBW1 + 2 *
valLBW2),trunc(GMoveBinLC[FromBin,ToBin]),fConfig.DiscountRate,y);

          PWSBenefits := PWSBenefits + vLBW * 20 * tmpPop;
        end;

        // ADHD
        vADHD := 0;
        if (A = 7) and (fLowHigh = rtLow) then
          vADHD := FADHD(FBL, TBL) * CPop;

        if (A = 11) and (fLowHigh = rtHigh) then
          vADHD := FADHD(FBL, TBL) * CPop;

        if FromBin > 16 then vADHD := -vADHD;
        if vADHD<>0 then begin
          Ben11 := Ben11 + CPop;
          _ADHD.AddValue(vADHD,trunc(GMoveBinLC[FromBin,ToBin]));
          _ADHDVal.DiscountAndAddValue(vADHD *
ADHDCaseVal,trunc(GMoveBinLC[FromBin,ToBin]),fConfig.DiscountRate,y);

          PWSBenefits := PWSBenefits + vADHD;
        end;

        //CVD
        if A >= 40 then begin
```

```
          if fConfig.RunNoBLAveraging then
            vCVD := FCVD(iFBL, iTBL, A, S) * CPop
          else
            vCVD := FCVD(FBL, TBL, A, S) * CPop;
          // count this cohort pop once
          CountPop := 0;
          if (not Counted40) and (vCVD > 0) then begin
            BenA := BenA + CPop;
            Counted40 := true;
            CountPop := CPop;
          end;
          if FromBin > 16 then vCVD := -vCVD;

          _CVD.AddValue(vCVD,trunc(GMoveBinLC[FromBin,ToBin]));
          _CVDVal.DiscountAndAddValue(vCVD *
VSL,trunc(GMoveBinLC[FromBin,ToBin]),fConfig.DiscountRate,y);

          PWSBenefits := PWSBenefits + vCVD;
        end;
      end;
    end;

    if not LLL.Exists('PWSBenefit_' + fConfig.RunName) then
      LLL.L('PWSBenefit_' + fConfig.RunName,'PWSId,y,PWSBenefits');
    LLL.L('PWSBenefit_' + fConfig.RunName, fPWSID + ',' + y.ToString + ',' +
PWSBenefits.ToString);

  end; //through analyis years....
end;


function TLCRBenefits.CorrectBL(const BL: double; const Age, Sex: integer): double;
begin
  if BL < 0.76 then begin
    if trunc(Age / 10) = 4 then begin
      if Sex = 1 then
        Result := 0.92
      else
        Result := 1.07;
    end
    else if trunc(Age / 10) = 5 then begin
      if Sex = 1 then
        Result := 0.97
      else
        Result := 1.05;
    end
    else if trunc(Age / 10) = 6 then begin
      if Sex = 1 then
        Result := 0.99
      else
```

```
      Result := 1.05;
  end
  else if trunc(Age / 10) = 7 then begin
    if Sex = 1 then
      Result := 1.03
    else
      Result := 1.04;
  end
  else if Sex = 1 then
    Result := 1.03
  else
    Result := 1.04;
end
else

  if BL < 1.12 then begin
  if trunc(Age / 10) = 4 then begin
    if Sex = 1 then
      Result := 0.93
    else
      Result := 1.04;
  end
  else if trunc(Age / 10) = 5 then begin
    if Sex = 1 then
      Result := 0.96
    else
      Result := 1.04;
  end
  else if trunc(Age / 10) = 6 then begin
    if Sex = 1 then
      Result := 0.94
    else
      Result := 1.02;
  end
  else if trunc(Age / 10) = 7 then begin
    if Sex = 1 then
      Result := 0.98
    else
      Result := 1.05;
  end
  else if Sex = 1 then
    Result := 0.98
  else
    Result := 1.05;
end
else

  if BL < 1.71 then begin
  if trunc(Age / 10) = 4 then begin
    if Sex = 1 then
```

```
      Result := 0.93
    else
      Result := 1.05;
  end
  else if trunc(Age / 10) = 5 then begin
    if Sex = 1 then
      Result := 0.96
    else
      Result := 1.02;
  end
  else if trunc(Age / 10) = 6 then begin
    if Sex = 1 then
      Result := 0.95
    else
      Result := 1.03;
  end
  else if trunc(Age / 10) = 7 then begin
    if Sex = 1 then
      Result := 0.99
    else
      Result := 1.05;
  end
  else if Sex = 1 then
    Result := 0.99
  else
    Result := 1.05;
end else

begin
  if trunc(Age / 10) = 4 then begin
    if Sex = 1 then
      Result := 0.94
    else
      Result := 1.05;
  end
  else if trunc(Age / 10) = 5 then begin
    if Sex = 1 then
      Result := 0.94
    else
      Result := 1.01;
  end
  else if trunc(Age / 10) = 6 then begin
    if Sex = 1 then
      Result := 0.95
    else
      Result := 1.04;
  end
  else if trunc(Age / 10) = 7 then begin
    if Sex = 1 then
      Result := 0.98
```

```
      else
        Result := 1.04;
    end
    else if Sex = 1 then
      Result := 0.98
    else
      Result := 1.04;
  end;

  Result := Result * BL;
end;

procedure TLCRBenefits.SnapShot;
begin
  _IQLossB := _IQLoss;
  _ADHDB := _ADHD;
  _LBWB := _LBW;
  _CVDB := _CVD;
  _IQLossValB := _IQLossVal;
  _ADHDValB := _ADHDVal;
  _LBWValB := _LBWVal;
  _CVDValB := _CVDVal;
end;

procedure TLCRBenefits.SnapDiff;
begin
  _IQLossB.Subtract(_IQLoss);
  _ADHDB.Subtract(_ADHD);
  _LBWB.Subtract(_LBW);
  _CVDB.Subtract(_CVD);
  _IQLossValB.Subtract(_IQLossVal);
  _ADHDValB.Subtract(_ADHDVal);
  _LBWValB.Subtract(_LBWVal);
  _CVDValB.Subtract(_CVDVal);

  _IQLossB.Multiply(-1);
  _ADHDB.Multiply(-1);
  _LBWB.Multiply(-1);
  _CVDB.Multiply(-1);
  _IQLossValB.Multiply(-1);
  _ADHDValB.Multiply(-1);
  _LBWValB.Multiply(-1);
  _CVDValB.Multiply(-1);

end;


procedure TLCRBenefits.CalcBenefitsNew(const A: TYearlyMovementMicro; const Wgt:
double;
  const DoDebug: boolean; const aCosts: TLCRCosts; const P90CCT, P90LSL: array of
```

```
double;
  abp1, abp2: integer; SmallSystem: boolean; ProxySystem: boolean; const OptionName:
string);
var
  y, F, T: integer;
  DoCCT: boolean;
  BenType: TBenType;
begin
  bp1 := abp1;
  bp2 := abp2;
  _IQLoss := Default(TBenRec);
  _IQLossVal := Default(TBenRec);
  _ADHD := Default(TBenRec);
  _ADHDVal := Default(TBenRec);
  _LBW := Default(TBenRec);
  _LBWVal := Default(TBenRec);
  _CVD := Default(TBenRec);
  _CVDVal := Default(TBenRec);

  DoDebugOUt := DoDebug;

  BenPop := 0;
  Ben7 := 0;
  BenA := 0;
  Ben0 := 0;
  Ben11 := 0;
  fillchar(FinalBins, sizeof(FinalBins), 0);
  fPWSID := aCosts.CostingData.PWSid;

  for y := 0 to fConfig.YearsOfOutput do begin
    if y = 0 then begin
      continue;
    end;

    for F := low(A[y]) to high(A[y]) do begin
      for T := low(A[y, F]) to high(A[y, F]) do begin
        if A[y, F, T] = 0 then
          continue;
        DoCCT := true;
        if GMoveBinLC[F, T] > 1 then
          DoCCT := false;

        if OptionName = 'LCRI' then begin
          BenType := btMandatory;

{$IFDEF DEBUG}
SnapShot;
{$ENDIF}
          CalcBinMove(F, T, y, A[y, F, T] * Wgt, DoCCT, BenType, ProxySystem);
```

```
{$IFDEF DEBUG}
SnapDiff;
//SnapShot;
if not LLL.Exists('binben') then LLL.L('binben','pws,allpop,f,t,y,pop,'+
'_iqlossb.C,_iqlossb.L,_iqlossb.POU,_iqlossb.tpou,'+
'_cvdb.C,_cvdb.L,_cvdb.POU,_cvdb.tpou,'+
'_lbwb.C,_lbwb.L,_lbwb.POU,_lbwb.tpou,'+
'_adhdb.C,_adhdb.L,_adhdb.POU,_adhdb.tpou,'+
'_iqlossvalb.C,_iqlossvalb.L,_iqlossvalb.POU,_iqlossvalb.tpou,'+
'_cvdvalb.C,_cvdvalb.L,_cvdvalb.POU,_cvdvalb.tpou,'+
'_lbwvalb.C,_lbwvalb.L,_lbwvalb.POU,_lbwvalb.tpou,'+
'_adhdvalb.C,_adhdvalb.L,_adhdvalb.POU,_adhdvalb.tpou'
);
LLL.L('binben',format('%s,%d,%d,%d,%d,%g,'+
'%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,'+
'%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g,%g'
,
[aCosts.CostingData.PWSid,aCosts.CostingData.Population,
f,t,y,A[y, F, T],
_iqlossb.C,_iqlossb.L,_iqlossb.POU,_iqlossb.tpou,
_cvdb.C,_cvdb.L,_cvdb.POU,_cvdb.tpou,
_lbwb.C,_lbwb.L,_lbwb.POU,_lbwb.tpou,
_adhdb.C,_adhdb.L,_adhdb.POU,_adhdb.tpou,
_iqlossvalb.C,_iqlossvalb.L,_iqlossvalb.POU,_iqlossvalb.tpou,
_cvdvalb.C,_cvdvalb.L,_cvdvalb.POU,_cvdvalb.tpou,
_lbwvalb.C,_lbwvalb.L,_lbwvalb.POU,_lbwvalb.tpou,
_adhdvalb.C,_adhdvalb.L,_adhdvalb.POU,_adhdvalb.tpou


]));


{$ENDIF}

        end else begin
          BenType := btMandatory;
          if DoCCT then begin
            if (P90CCT[y] <= bp2) then
              BenType := btVoluntary;
          end else begin
            if SmallSystem then begin
              if (P90LSL[y] <= bp2) then
                BenType := btRequested;
            end else begin
              if (P90LSL[y] <= bp1) then
                BenType := btRequested
              else if (P90LSL[y] <= bp2) then
                BenType := btVoluntary;
            end;
          end;
```

```
          CalcBinMove(F, T, y, A[y, F, T] * Wgt, DoCCT, BenType, ProxySystem);
        end;
      end;
    end;
  end;

  //total case count
  IQLoss := _IQLoss.Total;
  CVD := _CVD.Total;
  ADHD := _ADHD.Total;
  LBW := _LBW.Total;

  //annualize case count
  _IQLoss.DivideBy(fConfig.YearsOfOutput);
  _LBW.DivideBy(fConfig.YearsOfOutput);
  _ADHD.DivideBy(fConfig.YearsOfOutput);
  _CVD.DivideBy(fConfig.YearsOfOutput);
  //annualize dollars
  _IQLossVal.CalcAnnualInPlace(fConfig.DiscountRate, fConfig.YearsOfOutput);
  _LBWVal.CalcAnnualInPlace(fConfig.DiscountRate, fConfig.YearsOfOutput);
  _ADHDVal.CalcAnnualInPlace(fConfig.DiscountRate, fConfig.YearsOfOutput);
  _CVDVal.CalcAnnualInPlace(fConfig.DiscountRate, fConfig.YearsOfOutput);

  //totals dollars ...
  IQLossVal := _IQLossVal.Total;
  LBWVal := _LBWVal.Total;
  CVDVal := _CVDVal.Total;
  ADHDVal := _ADHDVal.Total;
  TotBen := IQLossVal + CVDVal + ADHDVal + LBWVal;

end;

function RCB(const c: integer): integer;
// this converts the BL table numbering to the Bin Numbering in pops.
begin
  // bin numbering has been corrected
  Result := c;
  exit;

  if c = 4 then
    Result := 1
  else if c = 7 then
    Result := 2
  else if c = 1 then
    Result := 3
  else if c = 5 then
    Result := 4
  else if c = 8 then
    Result := 5
  else if c = 2 then
```

```
    Result := 6
  else if c = 6 then
    Result := 7
  else if c = 9 then
    Result := 8
  else if c = 3 then
    Result := 9
  else
    Result := c;
end;

constructor TLCRBenefits.create;
var
  i, j, cMCL, cDR: integer;
  T, TL: TStringList;
  A: integer;
  v: double;
begin
  fConfig := aConfig;
  fOutputs := aOutputMetrics;
  fDummyProb := 1;
  DoDebugOUt := false;
  GotOne:=False;

  //TODO add messes here for other benefits....
  fBBYY := TLCRBenByYear.create(aConfig, aOption);
  fBBYY.AddBen('IQ CCT BP1');
  fBBYY.AddBen('IQ CCT BP2');
  fBBYY.AddBen('IQ LSLR Vol');
  fBBYY.AddBen('IQ LSLR Mand');
  fBBYY.AddBen('IQ LSLR Req');
  fBBYY.AddBen('IQ POU');

  fLowHigh := fConfig.RunType;
  fRunVersion := fConfig.RunVersion;

  fYearsOfOutput := fConfig.YearsOfOutput;
  UncertaintyVars := Uncertainty;

  fillchar(BL, sizeof(BL), 0);
  T := TStringList.create;
  TL := TStringList.create;

  T.Text := PBF;
  for i := 0 to T.Count - 1 do begin
    TL.CommaText := T[i];
    for j := 1 to 17 do begin
      BL[2, TL[0].ToInteger, RCB(j)] := TL[j].ToDouble;
    end;
    // bin 18-32 same as 17
```

```
  for var jj := 18 to 32 do
    BL[2, TL[0].ToInteger, RCB(jj)] := TL[17].ToDouble;
end;

T.Text := PBM;
for i := 0 to T.Count - 1 do begin
  TL.CommaText := T[i];

  for j := 1 to 17 do begin
    BL[1, TL[0].ToInteger, RCB(j)] := TL[j].ToDouble;
  end;
  // bin 18-32 same as 17
  for var jj := 18 to 32 do
    BL[1, TL[0].ToInteger, RCB(jj)] := TL[17].ToDouble;
end;

if fConfig.ChildBLSens > 0 then begin
  if fConfig.ChildBLSens = 1 then
    T.Text := PBSens1
  else if fConfig.ChildBLSens = 2 then
    T.Text := PBSens2
  else
    T.Text := PBSens3;
  for i := 0 to 6 do begin
    TL.CommaText := T[i];
    for j := 1 to 17 do begin
      BL[1, TL[0].ToInteger, RCB(j)] := TL[j].ToDouble;
      BL[2, TL[0].ToInteger, RCB(j)] := TL[j].ToDouble;
    end;
    // bin 18 same as 17
    BL[1, TL[0].ToInteger, RCB(18)] := TL[17].ToDouble;
    BL[2, TL[0].ToInteger, RCB(18)] := TL[17].ToDouble;

  end;
end;

TL.Free;
T.Free;

// no high-low concept yet for valuation.
// updating to 2021-22
VSL := 12980000;

if fConfig.IQValueSens = 1 then begin
  if Round(fConfig.DiscountRate * 100) / 100 = 0.03 then
    IQPointVal := 14936
  else
    IQPointVal := 3658;
end else begin
  if Round(fConfig.DiscountRate * 100) / 100 = 0.03 then
```

```
      IQPointVal := 27336
    else
      IQPointVal := 6887;
  end;

  if fLowHigh=rtHigh then begin
    if Round(fConfig.DiscountRate * 100) / 100 = 0.03 then
      ADHDCaseVal := 228231
    else
      ADHDCaseVal := 203823;
  end else begin
    if Round(fConfig.DiscountRate * 100) / 100 = 0.03 then
      ADHDCaseVal := 202780
    else
      ADHDCaseVal := 155496;
  end;

  CVDRate[1, 4] := 0.00078592;
  CVDRate[1, 5] := 0.00218595;
  CVDRate[1, 6] := 0.00459819;
  CVDRate[1, 7] := 0.01080168;
  CVDRate[1, 8] := 0.01080168;

  CVDRate[2, 4] := 0.00037709;
  CVDRate[2, 5] := 0.00097204;
  CVDRate[2, 6] := 0.00221088;
  CVDRate[2, 7] := 0.00675097;
  CVDRate[2, 8] := 0.00675097;

  DoDebugOUt := false;

end;


destructor TLCRBenefits.Destroy;
begin
  fBBYY.Free;
  inherited;
end;

function TLCRBenefits.FIQLoss(const BL1, BL2: double): double;
begin
  if fLowHigh = rtHigh then begin
    // change 8/2/2023 meeting
    Result := 3.14 * ln(BL1 / BL2);
  end
  else if fLowHigh = rtLow then begin
    Result := 3.25 * ln((BL1 + 1) / (BL2 + 1));
  end
  else
```

```
    raise Exception.create('Unknown fLowHigh in FIQLoss');

end;

function TLCRBenefits.FLBW(const BL1, BL2: double): double;
begin
  Result := -27.4 * (Power(BL2, 0.5) - Power(BL1, 0.5));
end;

function TLCRBenefits.FADHD(const BL1, BL2: double): double;
var
  Brate, Beta, v: double;
begin
  Brate := 0.096;
  if fLowHigh = rtLow then
    Beta := 0.223
  else
    Beta := 0.588;

  v := Beta * (ln(BL1) - ln(BL2));
  Result := Brate - (Brate / ((1 - Brate) * exp(-v) + Brate));
  Result := Result * -1;
end;

function TLCRBenefits.FCVD(const BL1, BL2: double; const Age, Sex: integer): double;
var
  P1, P2: double;
begin
  P1 := BL1;
  P2 := BL2;

//(*stashed 7/20/23
  if fLowHigh = rtHigh then
    Result := CVDRate[Sex, Age DIV 10] * (1 - exp(0.96 * log10(P2 / P1)))
  else
    Result := CVDRate[Sex, Age DIV 10] * (1 - exp(0.36 * log10(P2 / P1)));
//*)
  (*
  if fLowHigh = rtHigh then begin
    if ((p1 > 1) and (p2 > 1)) or ((p1 < 1) and (p2 < 1)) then begin
      Result := CVDRate[Sex, Age DIV 10] * (1 - exp(0.96 * log10(P2 / P1)))
    end else begin
      p2 := 1;
      Result := CVDRate[Sex, Age DIV 10] * (1-exp(0.416922703 * (P2-P1)));
    end;
  end else begin
    if ((p1 > 1) and (p2 > 1)) or ((p1 < 1) and (p2 < 1)) then begin
      Result := CVDRate[Sex, Age DIV 10] * (1 - exp(0.36 * log10(P2 / P1)))
    end else begin
      p2 := 1;
```

```
      Result := CVDRate[Sex, Age DIV 10] * (1-exp(0.156346013 * (P2-P1)));
    end;
  end;
  *)
end;

{ TBenefitsCollector }

constructor TBenefitsCollector.create(aConfig: TLCRConfig; aOutputMetrics:
TMetricList;
  aUncertainty: TUncertaintyStudy);
var
  i, F, T: integer;
begin
  fConfig := aConfig;
  fOutputs := aOutputMetrics;
  fUncertainty := aUncertainty;
  fDummyProb := 1;

  fLowHigh := fConfig.RunType;

  DoDebugOUt := false;
  NewBenBins := false;

  bp1 := fConfig.PWS90PctBp1;
  bp2 := fConfig.PWS90PctBp2;

  if fConfig.RunDifference then begin
    BenefitsBaseline := TLCRBenefits.create(fConfig, fOutputs, fUncertainty,
'Baseline');
    BenefitsOption := TLCRBenefits.create(fConfig, fOutputs, fUncertainty,
fConfig.OptionName);
  end
  else if fConfig.RunBaselineOnly then
    BenefitsBaseline := TLCRBenefits.create(fConfig, fOutputs, fUncertainty,
'Baseline')
  else
    BenefitsOption := TLCRBenefits.create(fConfig, fOutputs, fUncertainty,
fConfig.OptionName);

  for F := low(BinMovements) to high(BinMovements) do begin
    for T := low(BinMovements[F]) to high(BinMovements[F]) do begin
      if GMoveBinLC[F, T] > 0 then begin
        fOutputs.AddOutputMetric(@BinMovements[F, T], @DummyProb, nil,
          'BinMove' + inttostr(F) + 'To' + inttostr(T), mtBenefitCounts, false,
false, false,
          fConfig.OptionName, 0, true);
      end;
    end;
  end;
```

```
  fOutputs.AddOutputMetric(@BenPop, @DummyProb, nil, 'BenefitsBinPopMove',
mtBenefitCounts, false,
    false, false, fConfig.OptionName, 0, true);

  fOutputs.AddOutputMetric(@BenA, @DummyProb, nil, 'BenefittingAdults',
mtBenefitCounts, false,
    false, false, fConfig.OptionName, 0, true);

  fOutputs.AddOutputMetric(@Ben7, @DummyProb, nil, 'Benefitting7', mtBenefitCounts,
false, false,
    false, fConfig.OptionName, 0, true);
  fOutputs.AddOutputMetric(@Ben0, @DummyProb, nil, 'Benefitting0', mtBenefitCounts,
false, false,
    false, fConfig.OptionName, 0, true);
  fOutputs.AddOutputMetric(@Ben11, @DummyProb, nil, 'Benefitting11',
mtBenefitCounts, false, false,
    false, fConfig.OptionName, 0, true);

  _IQLoss.AddMetrics(fOutputs,'IQ
Point',fConfig.OptionName,fConfig.DiscountRate,false);
  _IQLossVal.AddMetrics(fOutputs,'IQ
Point',fConfig.OptionName,fConfig.DiscountRate,true);
  _CVD.AddMetrics(fOutputs,'CVD',fConfig.OptionName,fConfig.DiscountRate,false);
  _CVDVal.AddMetrics(fOutputs,'CVD',fConfig.OptionName,fConfig.DiscountRate,true);
  _ADHD.AddMetrics(fOutputs,'ADHD',fConfig.OptionName,fConfig.DiscountRate,false);
  _ADHDVal.AddMetrics(fOutputs,'ADHD',fConfig.OptionName,fConfig.DiscountRate,true);
  _LBW.AddMetrics(fOutputs,'LBW',fConfig.OptionName,fConfig.DiscountRate,false);
  _LBWVal.AddMetrics(fOutputs,'LBW',fConfig.OptionName,fConfig.DiscountRate,true);


  fOutputs.AddOutputMetric(@TotBen, @DummyProb, nil, 'Total Annual Benefits',
mtBenefitDollars,
    false, false, false, fConfig.OptionName, fConfig.DiscountRate, true);
  fOutputs.AddOutputMetric(@IQLossVal, @DummyProb, nil, 'IQ Point Annual',
mtBenefitDollars, false,
    false, false, fConfig.OptionName, fConfig.DiscountRate, true);
  fOutputs.AddOutputMetric(@ADHDVal, @DummyProb, nil, 'ADHD Annual',
mtBenefitDollars, false, false,
    false, fConfig.OptionName, fConfig.DiscountRate, true);
  fOutputs.AddOutputMetric(@CVDVal, @DummyProb, nil, 'CVD Annual', mtBenefitDollars,
false, false,
    false, fConfig.OptionName, fConfig.DiscountRate, true);
  fOutputs.AddOutputMetric(@LBWVal, @DummyProb, nil, 'LBW Annual', mtBenefitDollars,
false, false,
    false, fConfig.OptionName, fConfig.DiscountRate, true);

  fOutputs.AddOutputMetric(@IQLoss, @DummyProb, nil, 'IQ Point All', mtBenefitCases,
false,
    false, false, fConfig.OptionName, fConfig.DiscountRate, true);
```

```
  fOutputs.AddOutputMetric(@ADHD, @DummyProb, nil, 'ADHD All', mtBenefitCases,
false, false,
    false, fConfig.OptionName, fConfig.DiscountRate, true);
  fOutputs.AddOutputMetric(@CVD, @DummyProb, nil, 'CVD All', mtBenefitCases, false,
false,
    false, fConfig.OptionName, fConfig.DiscountRate, true);
  fOutputs.AddOutputMetric(@LBW, @DummyProb, nil, 'LBW All', mtBenefitCases, false,
false,
    false, fConfig.OptionName, fConfig.DiscountRate, true);

end;

destructor TBenefitsCollector.Destroy;
begin
  if fConfig.BenByYear then begin
    if assigned(BenefitsBaseline) then
      BenefitsBaseline.fBBYY.SaveResults;
    if assigned(BenefitsOption) then
      BenefitsOption.fBBYY.SaveResults;
  end;
  if assigned(BenefitsBaseline) then
    BenefitsBaseline.Free;
  if assigned(BenefitsOption) then
    BenefitsOption.Free;

  inherited;
end;

procedure TBenefitsCollector.GenerateBenefits(aCosts: TLCRCosts; ProxySystem:
boolean);
var
  i, F, T, y: integer;
  SmallSystem: boolean;
begin
  if ProxySystem then exit;

  SmallSystem := false;
  if (aCosts.CostingData.SystemType = 2) or // ntnc = 2
    (aCosts.CostingData.Population <= fConfig.SmallProxyPop) then
    SmallSystem := true;

  if fConfig.RunDifference then begin
    // acosts.CostingData.PWSid
    BenefitsBaseline.CalcBenefitsNew(aCosts.BaseCostSteps.GMoveBinMicro, 1,
DoDebugOUt, aCosts,
      aCosts.BaseCostSteps.pws90pctCCT_yr, aCosts.BaseCostSteps.pws90pctLSL_yr,
-100000, -100000,
      SmallSystem, ProxySystem, aCosts.Config.OptionName);
    BenefitsOption.CalcBenefitsNew(aCosts.ScenCostSteps.GMoveBinMicro, 1,
DoDebugOUt, aCosts,
```

```
      aCosts.ScenCostSteps.pws90pctCCT_yr, aCosts.ScenCostSteps.pws90pctLSL_yr, bp1,
bp2,
      SmallSystem, ProxySystem, aCosts.Config.OptionName);

    for i := low(FinalBins) to high(FinalBins) do begin
      FinalBins[i] := BenefitsOption.FinalBins[i] - BenefitsBaseline.FinalBins[i];
      StartingBins[i] := aCosts.ScenCostSteps.GMoveBin[0, i] -
aCosts.BaseCostSteps.GMoveBin[0, i];
      EndingBins[i] :=
aCosts.ScenCostSteps.GMoveBin[high(aCosts.ScenCostSteps.GMoveBin), i] -
        aCosts.BaseCostSteps.GMoveBin[high(aCosts.BaseCostSteps.GMoveBin), i];
      EndingBinsCheck[i] := aCosts.ScenCostSteps.GMoveBin[0, i];
    end;

    for y := low(aCosts.ScenCostSteps.GMoveBinMicro) to
high(aCosts.ScenCostSteps.GMoveBinMicro) do
    begin
      for F := low(aCosts.ScenCostSteps.GMoveBinMicro[y])
        to high(aCosts.ScenCostSteps.GMoveBinMicro[y]) do begin
        for T := low(aCosts.ScenCostSteps.GMoveBinMicro[y, F])
          to high(aCosts.ScenCostSteps.GMoveBinMicro[y, F]) do begin
          if y = low(aCosts.ScenCostSteps.GMoveBinMicro) then begin
            BinMovements[F, T] := aCosts.ScenCostSteps.GMoveBinTotal[F, T] -
              aCosts.BaseCostSteps.GMoveBinTotal[F, T];
          end;
          if aCosts.ScenCostSteps.GMoveBinMicro[y, F, T] < 0.01 then
            continue;
          EndingBinsCheck[F] := EndingBinsCheck[F] -
aCosts.ScenCostSteps.GMoveBinMicro[y, F, T];
          EndingBinsCheck[T] := EndingBinsCheck[T] +
aCosts.ScenCostSteps.GMoveBinMicro[y, F, T];
        end;
      end;
    end;

    BenPop := BenefitsOption.BenPop - BenefitsBaseline.BenPop;
    Ben7 := BenefitsOption.Ben7 - BenefitsBaseline.Ben7;
    Ben0 := BenefitsOption.Ben0 - BenefitsBaseline.Ben0;
    Ben11 := BenefitsOption.Ben11 - BenefitsBaseline.Ben11;
    BenA := BenefitsOption.BenA - BenefitsBaseline.BenA;

    _IQLoss := BenefitsOption._IQLoss;
    _IQLoss.Subtract(BenefitsBaseline._IQLoss);
    _IQLossVal := BenefitsOption._IQLossVal;
    _IQLossVal.Subtract(BenefitsBaseline._IQLossVal);

    _CVD := BenefitsOption._CVD;
    _CVD.Subtract(BenefitsBaseline._CVD);
    _CVDVal := BenefitsOption._CVDVal;
    _CVDVal.Subtract(BenefitsBaseline._CVDVal);
```

```
    _LBW := BenefitsOption._LBW;
    _LBW.Subtract(BenefitsBaseline._LBW);
    _LBWVal := BenefitsOption._LBWVal;
    _LBWVal.Subtract(BenefitsBaseline._LBWVal);

    _ADHD := BenefitsOption._ADHD;
    _ADHD.Subtract(BenefitsBaseline._ADHD);
    _ADHDVal := BenefitsOption._ADHDVal;
    _ADHDVal.Subtract(BenefitsBaseline._ADHDVal);

    IQLossVal := BenefitsOption.IQLossVal - BenefitsBaseline.IQLossVal;
    ADHDVal := BenefitsOption.ADHDVal - BenefitsBaseline.ADHDVal;
    CVDVal := BenefitsOption.CVDVal - BenefitsBaseline.CVDVal;
    LBWVal := BenefitsOption.LBWVal - BenefitsBaseline.LBWVal;
    TotBen := BenefitsOption.TotBen - BenefitsBaseline.TotBen;

    IQLoss := BenefitsOption.IQLoss - BenefitsBaseline.IQLoss;
    ADHD := BenefitsOption.ADHD - BenefitsBaseline.ADHD;
    CVD := BenefitsOption.CVD - BenefitsBaseline.CVD;
    LBW := BenefitsOption.LBW - BenefitsBaseline.LBW;

  end
  else if fConfig.RunBaselineOnly then begin
    BenefitsBaseline.CalcBenefitsNew(aCosts.BaseCostSteps.GMoveBinMicro, 1,
DoDebugOUt, aCosts,
      aCosts.BaseCostSteps.pws90pctCCT_yr, aCosts.BaseCostSteps.pws90pctLSL_yr,
-10000, -100000,
      SmallSystem, ProxySystem, aCosts.Config.OptionName);

    for i := low(FinalBins) to high(FinalBins) do begin
      FinalBins[i] := BenefitsBaseline.FinalBins[i];
      StartingBins[i] := aCosts.BaseCostSteps.GMoveBin[0, i];
      EndingBins[i] :=
aCosts.BaseCostSteps.GMoveBin[high(aCosts.BaseCostSteps.GMoveBin), i];
      EndingBinsCheck[i] := StartingBins[i];
    end;

    for y := low(aCosts.BaseCostSteps.GMoveBinMicro) to
high(aCosts.BaseCostSteps.GMoveBinMicro) do
    begin
      for F := low(aCosts.BaseCostSteps.GMoveBinMicro[y])
        to high(aCosts.BaseCostSteps.GMoveBinMicro[y]) do begin
        for T := low(aCosts.BaseCostSteps.GMoveBinMicro[y, F])
          to high(aCosts.BaseCostSteps.GMoveBinMicro[y, F]) do begin
          if y = low(aCosts.BaseCostSteps.GMoveBinMicro) then begin
            BinMovements[F, T] := aCosts.BaseCostSteps.GMoveBinTotal[F, T];
          end;
          if aCosts.BaseCostSteps.GMoveBinMicro[y, F, T] < 0.01 then
            continue;
```

```
          EndingBinsCheck[F] := EndingBinsCheck[F] -
aCosts.BaseCostSteps.GMoveBinMicro[y, F, T];
          EndingBinsCheck[T] := EndingBinsCheck[T] +
aCosts.BaseCostSteps.GMoveBinMicro[y, F, T];
        end;
      end;
    end;

    BenPop := BenefitsBaseline.BenPop;
    Ben7 := BenefitsBaseline.Ben7;
    Ben0 := BenefitsBaseline.Ben0;
    Ben11 := BenefitsBaseline.Ben11;
    BenA := BenefitsBaseline.BenA;

  end else begin
    BenefitsOption.CalcBenefitsNew(aCosts.ScenCostSteps.GMoveBinMicro, 1,
DoDebugOUt, aCosts,
      aCosts.ScenCostSteps.pws90pctCCT_yr, aCosts.ScenCostSteps.pws90pctLSL_yr, bp1,
bp2,
      SmallSystem, ProxySystem, aCosts.Config.OptionName);

    for i := low(FinalBins) to high(FinalBins) do begin
      FinalBins[i] := BenefitsOption.FinalBins[i];
      StartingBins[i] := aCosts.ScenCostSteps.GMoveBin[0, i];
      EndingBins[i] :=
aCosts.ScenCostSteps.GMoveBin[high(aCosts.ScenCostSteps.GMoveBin), i];
      EndingBinsCheck[i] := StartingBins[i];
    end;

    for y := low(aCosts.ScenCostSteps.GMoveBinMicro) to
high(aCosts.ScenCostSteps.GMoveBinMicro) do
    begin
      for F := low(aCosts.ScenCostSteps.GMoveBinMicro[y])
        to high(aCosts.ScenCostSteps.GMoveBinMicro[y]) do begin
        for T := low(aCosts.ScenCostSteps.GMoveBinMicro[y, F])
          to high(aCosts.ScenCostSteps.GMoveBinMicro[y, F]) do begin
          if y = low(aCosts.ScenCostSteps.GMoveBinMicro) then begin
            BinMovements[F, T] := aCosts.ScenCostSteps.GMoveBinTotal[F, T];
          end;
          if aCosts.ScenCostSteps.GMoveBinMicro[y, F, T] < 0.01 then
            continue;
          EndingBinsCheck[F] := EndingBinsCheck[F] -
aCosts.ScenCostSteps.GMoveBinMicro[y, F, T];
          EndingBinsCheck[T] := EndingBinsCheck[T] +
aCosts.ScenCostSteps.GMoveBinMicro[y, F, T];
        end;
      end;
    end;

    BenPop := BenefitsOption.BenPop;
```

```
    Ben7 := BenefitsOption.Ben7;
    Ben0 := BenefitsOption.Ben0;
    Ben11 := BenefitsOption.Ben11;
    BenA := BenefitsOption.BenA;

  end;

  //added this to keep from so much copy paste change code...
  if not fConfig.RunDifference then begin
    var B : TLCRBenefits;
    if fConfig.RunBaselineOnly then
      B := BenefitsBaseline
    else
      B := BenefitsOption;

    _IQLoss := B._IQLoss;
    _IQLossVal := B._IQLossVal;
    _CVD := B._CVD;
    _CVDVal := B._CVDVal;
    _LBW := B._LBW;
    _LBWVal := B._LBWVal;
    _ADHD := B._ADHD;
    _ADHDVal := B._ADHDVal;
    IQLossVal := B.IQLossVal;
    ADHDVal := B.ADHDVal;
    CVDVal := B.CVDVal;
    LBWVal := B.LBWVal;
    TotBen := B.TotBen;

    IQLoss := B.IQLoss;
    ADHD := B.ADHD;
    CVD := B.CVD;
    LBW := B.LBW;
  end;
end;

{ TLCRBenByYear }

procedure TLCRBenByYear.AddBen(aName: string);
begin
  fBenYears.Add(aName, TLCRBenYears.create);
end;

constructor TLCRBenByYear.create(aConfig: TLCRConfig; aName: string);
begin
  inherited create;
  fBenYears := TObjectDictionary<string, TLCRBenYears>.create([doOwnsValues]);
  fTotYears := aConfig.YearsOfOutput;
  Active := aConfig.BenByYear;
  fOutName := UserPath + aConfig.RunName + '_' + aName + '_BenByYear.csv';
```

```
end;

destructor TLCRBenByYear.Destroy;
begin
  fBenYears.Free;
  inherited;
end;

procedure TLCRBenByYear.SaveResults;
var
  M: array of array of double;
  S, ss, HL: string;
  i, j: integer;
  T: TLCRBenYears;
  TOut: TBufferedFileStream;
  DoIt: boolean;
begin
  if not Active then
    exit;
  setlength(M, fBenYears.Count, 150);
  HL := 'Year';
  i := 0;
  for S in fBenYears.Keys do begin
    HL := HL + ',' + S;
    T := fBenYears.Items[S];
    for j := 1 to 149 do begin
      M[i, j - 1] := T.Year[j];
    end;
    inc(i);
  end;
  TOut := TBufferedFileStream.create(fOutName, fmCreate);
  HL := HL + #13#10;
  TOut.WriteBuffer(HL[1], Length(HL) * sizeof(Char));
  for j := 1 to 150 do begin
    ss := j.ToString;
    DoIt := false;
    for i := 0 to fBenYears.Count - 1 do begin
      ss := ss + ',' + M[i, j - 1].ToString;
      if M[i, j - 1] <> 0 then
        DoIt := true;
    end;
    ss := ss + #13#10;
    if DoIt then
      TOut.WriteBuffer(ss[1], Length(ss) * sizeof(Char));
  end;
  TOut.Free;
end;

procedure TLCRBenByYear.UpdateBen(Yr: integer; const aName: string; aValue: double);
var
```

```
  T: TLCRBenYears;
  y: integer;
begin
  if not Active then
    exit;
  // IncomeDeltaAge is a total over 10 years....
  if fBenYears.TryGetValue(aName, T) then begin
    for y := 6 to high(IncomeDeltaAge) do begin
      T.Year[Yr + (y - 6)] := T.Year[Yr + (y - 6)] + aValue * (IncomeDeltaAge[y] /
10);
    end;
  end;
end;

{ TBenRec }

procedure TBenRec.Abs;
begin
  C := System.Abs(C);
  L := System.Abs(L);
  POU := System.Abs(POU);
  TPOU := System.Abs(TPOU);
end;

procedure TBenRec.AddMetrics(Outputs: TMetricList; Name, Option: string; DR: double;
  IsDollar: boolean);
begin
  var s: string;
  var DR2: double;
  var mt: integer;
  if IsDollar then begin
    s := Name + ' Annual';
    DR2 := DR;
    mt := mtBenefitDollars;
  end else begin
    s := Name + ' Annual Cases';
    DR2 := 0;
    mt := mtBenefitCases;
  end;

  Outputs.AddOutputMetric(@C, @Dummy, nil, s+'_CCT', mt, false,
    false, false, Option, DR2, true);
  Outputs.AddOutputMetric(@L, @Dummy, nil, s+'_LSLR', mt, false,
    false, false, Option, DR2, true);
  Outputs.AddOutputMetric(@TPOU, @Dummy, nil, s+'_TPOU', mt, false,
    false, false, Option, DR2, true);
  Outputs.AddOutputMetric(@POU, @Dummy, nil, s+'_POU', mt, false,
    false, false, Option, DR2, true);
end;
```

```
procedure TBenRec.AddValue(const v: double; const cat: Integer);
begin
  case cat of
    1: C := C + v;
    2: L := L + v;
    3: TPOU := TPOU + v;
    4: POU := POU + v;
  else
    CSL('bad cat addvalue:'+cat.ToString);
  end;
end;

procedure TBenRec.CalcAnnual(const DR: double; const r: TBenRec; const Yrs:
integer);
begin
  C := Annualize(DR, r.C, Yrs);
  L := Annualize(DR, r.L, Yrs);;
  POU := Annualize(DR, r.POU, Yrs);
  TPOU := Annualize(DR, r.TPOU, Yrs);
end;

procedure TBenRec.CalcAnnualInPlace(const DR: double; const Yrs: integer);
begin
  C := Annualize(DR, C, Yrs);
  L := Annualize(DR, L, Yrs);;
  POU := Annualize(DR, POU, Yrs);
  TPOU := Annualize(DR, TPOU, Yrs);
end;

procedure TBenRec.DiscountAndAddValue(const v: double; const cat: Integer;
  const DR: double; const Yrs: integer);
begin
  var tmpv := Discount(v, yrs, DR);
  case cat of
    1: C := C + tmpv;
    2: L := L + tmpv;
    3: TPOU := TPOU + tmpv;
    4: POU := POU + tmpv;
  else
    //? raise? - this shouldn't happen
    CSL('bad cat discountandaddvalue:'+cat.ToString);
  end;
end;

procedure TBenRec.DivideBy(const v: double);
begin
  C := C/v;
  L := L/v;
  POU := POU/v;
  TPOU := TPOU/v;
```

```
end;

procedure TBenRec.Multiply(const v: double);
begin
  C := C*v;
  L := L*v;
  POU := POU*v;
  TPOU := TPOU*v;
end;

procedure TBenRec.Subtract(const r: TBenRec);
begin
  C := C - R.C;
  L := L - R.L;
  POU := POU - R.POU;
  TPOU := TPOU - R.TPOU;
end;

function TBenRec.Total(): double;
begin
  Result := C + L + POU + TPOU;
end;

initialization

//blood leads from 6/8/23 and 6/21/23 spreadsheets from meghan

PBF:=
'0,3.476,2.269,0.962,2.487,2.487,2.487,1.720,1.720,1.720,0.962,0.962,0.962,1.798,1.3
33,0.962,0.962,0.962,0.962'+#13#10+
'1,2.435,1.830,1.130,1.886,1.886,1.886,1.513,1.513,1.513,1.130,1.130,1.130,1.562,1.3
14,1.130,1.130,1.130,1.130'+#13#10+
'2,2.612,1.905,1.158,1.996,1.996,1.996,1.559,1.559,1.559,1.158,1.158,1.158,1.641,1.3
44,1.158,1.158,1.158,1.158'+#13#10+
'3,2.456,1.775,1.153,1.917,1.917,1.917,1.503,1.503,1.503,1.153,1.153,1.153,1.570,1.3
28,1.153,1.153,1.153,1.153'+#13#10+
'4,2.437,1.792,1.134,1.917,1.917,1.917,1.501,1.501,1.501,1.134,1.134,1.134,1.571,1.3
15,1.134,1.134,1.134,1.134'+#13#10+
'5,2.573,1.864,1.186,1.991,1.991,1.991,1.531,1.531,1.531,1.186,1.186,1.186,1.618,1.3
64,1.186,1.186,1.186,1.186'+#13#10+
'6,2.287,1.630,0.977,1.758,1.758,1.758,1.350,1.350,1.350,0.977,0.977,0.977,1.410,1.1
71,0.977,0.977,0.977,0.977'+#13#10+
'7,2.528,1.825,1.066,1.973,1.973,1.973,1.476,1.476,1.476,1.066,1.066,1.066,1.556,1.2
82,1.066,1.066,1.066,1.066'+#13#10+
'8,1.464,1.072,0.765,1.136,1.136,1.136,0.922,0.922,0.922,0.765,0.765,0.765,0.95,0.83
8,0.765,0.765,0.765'+#13#10+
'9,1.399,1.025,0.733,1.086,1.086,1.086,0.883,0.883,0.883,0.733,0.733,0.733,0.909,0.8
03,0.733,0.733,0.733'+#13#10+
'10,1.281,0.939,0.671,0.995,0.995,0.995,0.809,0.809,0.809,0.671,0.671,0.671,0.832,0.
735,0.671,0.671,0.671'+#13#10+
```

```
'11,1.111,0.814,0.582,0.863,0.863,0.863,0.701,0.701,0.701,0.582,0.582,0.582,0.722,0.
637,0.582,0.582,0.582'+#13#10+
'12,1.001,0.733,0.523,0.777,0.777,0.777,0.631,0.631,0.631,0.523,0.523,0.523,0.649,0.
573,0.523,0.523,0.523'+#13#10+
'13,0.936,0.684,0.487,0.725,0.725,0.725,0.588,0.588,0.588,0.487,0.487,0.487,0.606,0.
534,0.487,0.487,0.487'+#13#10+
'14,0.902,0.658,0.468,0.698,0.698,0.698,0.565,0.565,0.565,0.468,0.468,0.468,0.582,0.
513,0.468,0.468,0.468'+#13#10+
'15,0.94,0.684,0.484,0.726,0.726,0.726,0.587,0.587,0.587,0.484,0.484,0.484,0.604,0.5
32,0.484,0.484,0.484'+#13#10+
'16,1.048,0.76,0.534,0.807,0.807,0.807,0.65,0.65,0.65,0.534,0.534,0.534,0.67,0.588,0
.534,0.534,0.534'+#13#10+
'17,1.158,0.836,0.585,0.889,0.889,0.889,0.714,0.714,0.714,0.585,0.585,0.585,0.736,0.
645,0.585,0.585,0.585'+#13#10+
'18,1.265,0.91,0.631,0.968,0.968,0.968,0.774,0.774,0.774,0.631,0.631,0.631,0.799,0.6
98,0.631,0.631,0.631'+#13#10+
'19,1.365,0.977,0.673,1.04,1.04,1.04,0.829,0.829,0.829,0.673,0.673,0.673,0.856,0.746
,0.673,0.673,0.673'+#13#10+
'20,1.456,1.037,0.709,1.106,1.106,1.106,0.878,0.878,0.878,0.709,0.709,0.709,0.907,0.
787,0.709,0.709,0.709'+#13#10+
'21,1.537,1.089,0.739,1.162,1.162,1.162,0.919,0.919,0.919,0.739,0.739,0.739,0.95,0.8
22,0.739,0.739,0.739'+#13#10+
'22,1.603,1.131,0.761,1.208,1.208,1.208,0.951,0.951,0.951,0.761,0.761,0.761,0.983,0.
849,0.761,0.761,0.761'+#13#10+
'23,1.654,1.16,0.774,1.241,1.241,1.241,0.972,0.972,0.972,0.774,0.774,0.774,1.007,0.8
66,0.774,0.774,0.774'+#13#10+
'24,1.687,1.177,0.778,1.26,1.26,1.26,0.983,0.983,0.983,0.778,0.778,0.778,1.018,0.873
,0.778,0.778,0.778'+#13#10+
'25,1.699,1.178,0.772,1.264,1.264,1.264,0.981,0.981,0.981,0.772,0.772,0.772,1.017,0.
869,0.772,0.772,0.772'+#13#10+
'26,1.714,1.184,0.77,1.271,1.271,1.271,0.983,0.983,0.983,0.77,0.77,0.77,1.02,0.869,0
.77,0.77,0.77'+#13#10+
'27,1.731,1.191,0.769,1.279,1.279,1.279,0.986,0.986,0.986,0.769,0.769,0.769,1.023,0.
87,0.769,0.769,0.769'+#13#10+
'28,1.745,1.196,0.767,1.286,1.286,1.286,0.987,0.987,0.987,0.767,0.767,0.767,1.025,0.
869,0.767,0.767,0.767'+#13#10+
'29,1.755,1.198,0.763,1.289,1.289,1.289,0.986,0.986,0.986,0.763,0.763,0.763,1.025,0.
866,0.763,0.763,0.763'+#13#10+
'30,1.762,1.198,0.757,1.29,1.29,1.29,0.984,0.984,0.984,0.757,0.757,0.757,1.023,0.862
,0.757,0.757,0.757'+#13#10+
'31,1.768,1.197,0.751,1.291,1.291,1.291,0.98,0.98,0.98,0.751,0.751,0.751,1.02,0.858,
0.751,0.751,0.751'+#13#10+
'32,1.774,1.197,0.746,1.292,1.292,1.292,0.978,0.978,0.978,0.746,0.746,0.746,1.018,0.
854,0.746,0.746,0.746'+#13#10+
'33,1.782,1.198,0.741,1.293,1.293,1.293,0.976,0.976,0.976,0.741,0.741,0.741,1.016,0.
85,0.741,0.741,0.741'+#13#10+
'34,1.79,1.199,0.737,1.296,1.296,1.296,0.974,0.974,0.974,0.737,0.737,0.737,1.015,0.8
47,0.737,0.737,0.737'+#13#10+
'35,1.8,1.201,0.733,1.299,1.299,1.299,0.974,0.974,0.974,0.733,0.733,0.733,1.015,0.84
5,0.733,0.733,0.733'+#13#10+
```

```
'36,1.811,1.204,0.73,1.304,1.304,1.304,0.974,0.974,0.974,0.73,0.73,0.73,1.016,0.843,
0.73,0.73,0.73'+#13#10+
'37,1.823,1.208,0.727,1.309,1.309,1.309,0.974,0.974,0.974,0.727,0.727,0.727,1.017,0.
842,0.727,0.727,0.727'+#13#10+
'38,1.836,1.212,0.725,1.315,1.315,1.315,0.975,0.975,0.975,0.725,0.725,0.725,1.019,0.
841,0.725,0.725,0.725'+#13#10+
'39,1.849,1.217,0.723,1.321,1.321,1.321,0.977,0.977,0.977,0.723,0.723,0.723,1.021,0.
841,0.723,0.723,0.723'+#13#10+
'40,1.862,1.221,0.721,1.326,1.326,1.326,0.978,0.978,0.978,0.721,0.721,0.721,1.022,0.
84,0.721,0.721,0.721'+#13#10+
'41,1.873,1.225,0.718,1.331,1.331,1.331,0.978,0.978,0.978,0.718,0.718,0.718,1.023,0.
839,0.718,0.718,0.718'+#13#10+
'42,1.887,1.229,0.716,1.337,1.337,1.337,0.979,0.979,0.979,0.716,0.716,0.716,1.025,0.
838,0.716,0.716,0.716'+#13#10+
'43,1.901,1.234,0.714,1.343,1.343,1.343,0.981,0.981,0.981,0.714,0.714,0.714,1.027,0.
838,0.714,0.714,0.714'+#13#10+
'44,1.915,1.24,0.712,1.35,1.35,1.35,0.983,0.983,0.983,0.712,0.712,0.712,1.03,0.838,0
.712,0.712,0.712'+#13#10+
'45,1.93,1.245,0.71,1.357,1.357,1.357,0.985,0.985,0.985,0.71,0.71,0.71,1.032,0.838,0
.71,0.71,0.71'+#13#10+
'46,1.944,1.251,0.708,1.364,1.364,1.364,0.987,0.987,0.987,0.708,0.708,0.708,1.035,0.
838,0.708,0.708,0.708'+#13#10+
'47,1.959,1.256,0.707,1.371,1.371,1.371,0.989,0.989,0.989,0.707,0.707,0.707,1.037,0.
838,0.707,0.707,0.707'+#13#10+
'48,1.974,1.262,0.705,1.378,1.378,1.378,0.991,0.991,0.991,0.705,0.705,0.705,1.04,0.8
38,0.705,0.705,0.705'+#13#10+
'49,1.989,1.267,0.703,1.386,1.386,1.386,0.993,0.993,0.993,0.703,0.703,0.703,1.043,0.
838,0.703,0.703,0.703'+#13#10+
'50,2,1.271,0.701,1.39,1.39,1.39,0.994,0.994,0.994,0.701,0.701,0.701,1.044,0.837,0.7
01,0.701,0.701'+#13#10+
'51,2.003,1.271,0.699,1.391,1.391,1.391,0.993,0.993,0.993,0.699,0.699,0.699,1.044,0.
836,0.699,0.699,0.699'+#13#10+
'52,2.005,1.271,0.697,1.391,1.391,1.391,0.992,0.992,0.992,0.697,0.697,0.697,1.042,0.
834,0.697,0.697,0.697'+#13#10+
'53,2.006,1.27,0.695,1.391,1.391,1.391,0.99,0.99,0.99,0.695,0.695,0.695,1.041,0.832,
0.695,0.695,0.695'+#13#10+
'54,2.007,1.269,0.692,1.39,1.39,1.39,0.989,0.989,0.989,0.692,0.692,0.692,1.04,0.83,0
.692,0.692,0.692'+#13#10+
'55,2.008,1.268,0.69,1.389,1.389,1.389,0.987,0.987,0.987,0.69,0.69,0.69,1.038,0.828,
0.69,0.69,0.69'+#13#10+
'56,2.009,1.267,0.688,1.389,1.389,1.389,0.986,0.986,0.986,0.688,0.688,0.688,1.037,0.
826,0.688,0.688,0.688'+#13#10+
'57,2.009,1.266,0.686,1.388,1.388,1.388,0.984,0.984,0.984,0.686,0.686,0.686,1.035,0.
824,0.686,0.686,0.686'+#13#10+
'58,2.009,1.265,0.683,1.387,1.387,1.387,0.982,0.982,0.982,0.683,0.683,0.683,1.033,0.
822,0.683,0.683,0.683'+#13#10+
'59,2.008,1.263,0.681,1.385,1.385,1.385,0.98,0.98,0.98,0.681,0.681,0.681,1.032,0.82,
0.681,0.681,0.681'+#13#10+
'60,2.008,1.262,0.679,1.384,1.384,1.384,0.978,0.978,0.978,0.679,0.679,0.679,1.03,0.8
18,0.679,0.679,0.679'+#13#10+
```

```
'61,2.008,1.261,0.677,1.383,1.383,1.383,0.977,0.977,0.977,0.677,0.677,0.677,1.029,0.
816,0.677,0.677,0.677'+#13#10+
'62,2.008,1.26,0.675,1.382,1.382,1.382,0.976,0.976,0.976,0.675,0.675,0.675,1.027,0.8
15,0.675,0.675,0.675'+#13#10+
'63,2.007,1.259,0.674,1.381,1.381,1.381,0.974,0.974,0.974,0.674,0.674,0.674,1.026,0.
813,0.674,0.674,0.674'+#13#10+
'64,2.007,1.258,0.672,1.38,1.38,1.38,0.973,0.973,0.973,0.672,0.672,0.672,1.025,0.812
,0.672,0.672,0.672'+#13#10+
'65,2.007,1.257,0.67,1.379,1.379,1.379,0.971,0.971,0.971,0.67,0.67,0.67,1.023,0.81,0
.67,0.67,0.67'+#13#10+
'66,2.006,1.255,0.669,1.378,1.378,1.378,0.97,0.97,0.97,0.669,0.669,0.669,1.022,0.808
,0.669,0.669,0.669'+#13#10+
'67,2.006,1.254,0.667,1.377,1.377,1.377,0.969,0.969,0.969,0.667,0.667,0.667,1.021,0.
807,0.667,0.667,0.667'+#13#10+
'68,2.005,1.253,0.665,1.376,1.376,1.376,0.967,0.967,0.967,0.665,0.665,0.665,1.019,0.
805,0.665,0.665,0.665'+#13#10+
'69,2.005,1.252,0.664,1.375,1.375,1.375,0.966,0.966,0.966,0.664,0.664,0.664,1.018,0.
804,0.664,0.664,0.664'+#13#10+
'70,2.004,1.251,0.662,1.374,1.374,1.374,0.964,0.964,0.964,0.662,0.662,0.662,1.016,0.
802,0.662,0.662,0.662'+#13#10+
'71,2.003,1.249,0.66,1.372,1.372,1.372,0.962,0.962,0.962,0.66,0.66,0.66,1.014,0.8,0.
66,0.66,0.66'+#13#10+
'72,2.001,1.247,0.657,1.371,1.371,1.371,0.96,0.96,0.96,0.657,0.657,0.657,1.012,0.798
,0.657,0.657,0.657'+#13#10+
'73,2,1.245,0.655,1.369,1.369,1.369,0.958,0.958,0.958,0.655,0.655,0.655,1.01,0.796,0
.655,0.655,0.655'+#13#10+
'74,1.999,1.243,0.653,1.367,1.367,1.367,0.956,0.956,0.956,0.653,0.653,0.653,1.008,0.
794,0.653,0.653,0.653'+#13#10+
'75,1.997,1.241,0.651,1.365,1.365,1.365,0.954,0.954,0.954,0.651,0.651,0.651,1.006,0.
792,0.651,0.651,0.651'+#13#10+
'76,1.996,1.24,0.648,1.363,1.363,1.363,0.952,0.952,0.952,0.648,0.648,0.648,1.004,0.7
89,0.648,0.648,0.648'+#13#10+
'77,1.994,1.238,0.646,1.361,1.361,1.361,0.95,0.95,0.95,0.646,0.646,0.646,1.002,0.787
,0.646,0.646,0.646'+#13#10+
'78,1.992,1.236,0.644,1.359,1.359,1.359,0.948,0.948,0.948,0.644,0.644,0.644,1,0.785,
0.644,0.644,0.644'+#13#10+
'79,1.992,1.235,0.643,1.359,1.359,1.359,0.947,0.947,0.947,0.643,0.643,0.643,0.999,0.
784,0.643,0.643,0.643'+#13#10+
'80,1.992,1.235,0.643,1.359,1.359,1.359,0.947,0.947,0.947,0.643,0.643,0.643,0.999,0.
784,0.643,0.643,0.643'
 ;

PBM:=
'0,3.476,2.269,0.962,2.487,2.487,2.487,1.720,1.720,1.720,0.962,0.962,0.962,1.798,1.3
33,0.962,0.962,0.962,0.962'+#13#10+
'1,2.435,1.830,1.130,1.886,1.886,1.886,1.513,1.513,1.513,1.130,1.130,1.130,1.562,1.3
14,1.130,1.130,1.130,1.130'+#13#10+
'2,2.612,1.905,1.158,1.996,1.996,1.996,1.559,1.559,1.559,1.158,1.158,1.158,1.641,1.3
44,1.158,1.158,1.158,1.158'+#13#10+
'3,2.456,1.775,1.153,1.917,1.917,1.917,1.503,1.503,1.503,1.153,1.153,1.153,1.570,1.3
```

```
28,1.153,1.153,1.153,1.153'+#13#10+
'4,2.437,1.792,1.134,1.917,1.917,1.917,1.501,1.501,1.501,1.134,1.134,1.134,1.571,1.3
15,1.134,1.134,1.134,1.134'+#13#10+
'5,2.573,1.864,1.186,1.991,1.991,1.991,1.531,1.531,1.531,1.186,1.186,1.186,1.618,1.3
64,1.186,1.186,1.186,1.186'+#13#10+
'6,2.287,1.630,0.977,1.758,1.758,1.758,1.350,1.350,1.350,0.977,0.977,0.977,1.410,1.1
71,0.977,0.977,0.977,0.977'+#13#10+
'7,2.528,1.825,1.066,1.973,1.973,1.973,1.476,1.476,1.476,1.066,1.066,1.066,1.556,1.2
82,1.066,1.066,1.066,1.066'+#13#10+
'8,1.516,1.107,0.788,1.174,1.174,0.952,0.952,0.952,0.952,0.788,0.788,0.788,0.98,0.86
4,0.788,0.788,0.788'+#13#10+
'9,1.507,1.103,0.788,1.17,1.17,0.95,0.95,0.95,0.95,0.788,0.788,0.788,0.978,0.863,0.7
88,0.788,0.788'+#13#10+
'10,1.424,1.045,0.748,1.107,1.107,0.9,0.9,0.9,0.9,0.748,0.748,0.748,0.927,0.819,0.74
8,0.748,0.748'+#13#10+
'11,1.249,0.919,0.661,0.973,0.973,0.793,0.793,0.793,0.793,0.661,0.661,0.661,0.816,0.
722,0.661,0.661,0.661'+#13#10+
'12,1.104,0.814,0.588,0.862,0.862,0.704,0.704,0.704,0.704,0.588,0.588,0.588,0.724,0.
642,0.588,0.588,0.588'+#13#10+
'13,0.989,0.731,0.53,0.773,0.773,0.633,0.633,0.633,0.633,0.53,0.53,0.53,0.651,0.578,
0.53,0.53,0.53'+#13#10+
'14,0.904,0.67,0.487,0.708,0.708,0.581,0.581,0.581,0.581,0.487,0.487,0.487,0.597,0.5
31,0.487,0.487,0.487'+#13#10+
'15,0.895,0.664,0.483,0.702,0.702,0.576,0.576,0.576,0.576,0.483,0.483,0.483,0.592,0.
526,0.483,0.483,0.483'+#13#10+
'16,0.957,0.709,0.515,0.749,0.749,0.614,0.614,0.614,0.614,0.515,0.515,0.515,0.632,0.
561,0.515,0.515,0.515'+#13#10+
'17,1.026,0.759,0.55,0.802,0.802,0.657,0.657,0.657,0.657,0.55,0.55,0.55,0.676,0.6,0.
55,0.55,0.55'+#13#10+
'18,1.098,0.811,0.587,0.858,0.858,0.702,0.702,0.702,0.702,0.587,0.587,0.587,0.722,0.
64,0.587,0.587,0.587'+#13#10+
'19,1.172,0.864,0.623,0.914,0.914,0.746,0.746,0.746,0.746,0.623,0.623,0.623,0.768,0.
68,0.623,0.623,0.623'+#13#10+
'20,1.242,0.913,0.656,0.967,0.967,0.788,0.788,0.788,0.788,0.656,0.656,0.656,0.811,0.
717,0.656,0.656,0.656'+#13#10+
'21,1.305,0.956,0.684,1.013,1.013,0.824,0.824,0.824,0.824,0.684,0.684,0.684,0.848,0.
749,0.684,0.684,0.684'+#13#10+
'22,1.358,0.992,0.706,1.052,1.052,0.853,0.853,0.853,0.853,0.706,0.706,0.706,0.878,0.
774,0.706,0.706,0.706'+#13#10+
'23,1.4,1.019,0.72,1.081,1.081,0.873,0.873,0.873,0.873,0.72,0.72,0.72,0.9,0.791,0.72
,0.72,0.72'+#13#10+
'24,1.428,1.034,0.726,1.099,1.099,0.885,0.885,0.885,0.885,0.726,0.726,0.726,0.912,0.
8,0.726,0.726,0.726'+#13#10+
'25,1.438,1.037,0.723,1.102,1.102,0.884,0.884,0.884,0.884,0.723,0.723,0.723,0.912,0.
798,0.723,0.723,0.723'+#13#10+
'26,1.45,1.042,0.723,1.109,1.109,0.887,0.887,0.887,0.887,0.723,0.723,0.723,0.915,0.7
99,0.723,0.723,0.723'+#13#10+
'27,1.464,1.048,0.723,1.116,1.116,0.89,0.89,0.89,0.89,0.723,0.723,0.723,0.919,0.801,
0.723,0.723,0.723'+#13#10+
'28,1.475,1.053,0.722,1.122,1.122,0.892,0.892,0.892,0.892,0.722,0.722,0.722,0.921,0.
```

```
801,0.722,0.722,0.722'+#13#10+
'29,1.484,1.055,0.719,1.125,1.125,0.892,0.892,0.892,0.892,0.719,0.719,0.719,0.921,0.
799,0.719,0.719,0.719'+#13#10+
'30,1.489,1.055,0.715,1.126,1.126,0.89,0.89,0.89,0.89,0.715,0.715,0.715,0.92,0.796,0
.715,0.715,0.715'+#13#10+
'31,1.494,1.054,0.711,1.126,1.126,0.887,0.887,0.887,0.887,0.711,0.711,0.711,0.917,0.
792,0.711,0.711,0.711'+#13#10+
'32,1.498,1.054,0.706,1.126,1.126,0.884,0.884,0.884,0.884,0.706,0.706,0.706,0.915,0.
789,0.706,0.706,0.706'+#13#10+
'33,1.504,1.054,0.702,1.127,1.127,0.882,0.882,0.882,0.882,0.702,0.702,0.702,0.914,0.
786,0.702,0.702,0.702'+#13#10+
'34,1.51,1.054,0.698,1.129,1.129,0.881,0.881,0.881,0.881,0.698,0.698,0.698,0.912,0.7
83,0.698,0.698,0.698'+#13#10+
'35,1.517,1.056,0.695,1.131,1.131,0.88,0.88,0.88,0.88,0.695,0.695,0.695,0.912,0.781,
0.695,0.695,0.695'+#13#10+
'36,1.525,1.058,0.692,1.134,1.134,0.88,0.88,0.88,0.88,0.692,0.692,0.692,0.912,0.779,
0.692,0.692,0.692'+#13#10+
'37,1.534,1.06,0.69,1.138,1.138,0.88,0.88,0.88,0.88,0.69,0.69,0.69,0.913,0.778,0.69,
0.69,0.69'+#13#10+
'38,1.544,1.063,0.687,1.142,1.142,0.88,0.88,0.88,0.88,0.687,0.687,0.687,0.914,0.777,
0.687,0.687,0.687'+#13#10+
'39,1.554,1.067,0.686,1.146,1.146,0.881,0.881,0.881,0.881,0.686,0.686,0.686,0.915,0.
776,0.686,0.686,0.686'+#13#10+
'40,1.563,1.069,0.683,1.15,1.15,0.881,0.881,0.881,0.881,0.683,0.683,0.683,0.916,0.77
5,0.683,0.683,0.683'+#13#10+
'41,1.572,1.072,0.681,1.154,1.154,0.881,0.881,0.881,0.881,0.681,0.681,0.681,0.916,0.
774,0.681,0.681,0.681'+#13#10+
'42,1.582,1.075,0.678,1.158,1.158,0.882,0.882,0.882,0.882,0.678,0.678,0.678,0.917,0.
773,0.678,0.678,0.678'+#13#10+
'43,1.592,1.078,0.676,1.163,1.163,0.883,0.883,0.883,0.883,0.676,0.676,0.676,0.918,0.
772,0.676,0.676,0.676'+#13#10+
'44,1.603,1.082,0.675,1.167,1.167,0.884,0.884,0.884,0.884,0.675,0.675,0.675,0.92,0.7
72,0.675,0.675,0.675'+#13#10+
'45,1.614,1.086,0.673,1.172,1.172,0.885,0.885,0.885,0.885,0.673,0.673,0.673,0.922,0.
771,0.673,0.673,0.673'+#13#10+
'46,1.625,1.09,0.671,1.178,1.178,0.886,0.886,0.886,0.886,0.671,0.671,0.671,0.923,0.7
71,0.671,0.671,0.671'+#13#10+
'47,1.637,1.094,0.67,1.183,1.183,0.887,0.887,0.887,0.887,0.67,0.67,0.67,0.925,0.771,
0.67,0.67,0.67'+#13#10+
'48,1.648,1.098,0.668,1.188,1.188,0.889,0.889,0.889,0.889,0.668,0.668,0.668,0.927,0.
77,0.668,0.668,0.668'+#13#10+
'49,1.659,1.102,0.666,1.193,1.193,0.89,0.89,0.89,0.89,0.666,0.666,0.666,0.929,0.77,0
.666,0.666,0.666'+#13#10+
'50,1.667,1.104,0.665,1.196,1.196,0.89,0.89,0.89,0.89,0.665,0.665,0.665,0.929,0.769,
0.665,0.665,0.665'+#13#10+
'51,1.669,1.104,0.663,1.197,1.197,0.89,0.89,0.89,0.89,0.663,0.663,0.663,0.929,0.768,
0.663,0.663,0.663'+#13#10+
'52,1.671,1.104,0.661,1.197,1.197,0.889,0.889,0.889,0.889,0.661,0.661,0.661,0.928,0.
767,0.661,0.661,0.661'+#13#10+
'53,1.672,1.104,0.659,1.197,1.197,0.888,0.888,0.888,0.888,0.659,0.659,0.659,0.927,0.
```

```
765,0.659,0.659,0.659'+#13#10+
'54,1.673,1.103,0.658,1.197,1.197,0.887,0.887,0.887,0.887,0.658,0.658,0.658,0.926,0.
764,0.658,0.658,0.658'+#13#10+
'55,1.674,1.103,0.656,1.196,1.196,0.885,0.885,0.885,0.885,0.656,0.656,0.656,0.925,0.
762,0.656,0.656,0.656'+#13#10+
'56,1.674,1.102,0.654,1.195,1.195,0.884,0.884,0.884,0.884,0.654,0.654,0.654,0.924,0.
761,0.654,0.654,0.654'+#13#10+
'57,1.674,1.101,0.653,1.195,1.195,0.883,0.883,0.883,0.883,0.653,0.653,0.653,0.923,0.
759,0.653,0.653,0.653'+#13#10+
'58,1.674,1.1,0.651,1.194,1.194,0.881,0.881,0.881,0.881,0.651,0.651,0.651,0.921,0.75
8,0.651,0.651,0.651'+#13#10+
'59,1.674,1.099,0.649,1.193,1.193,0.88,0.88,0.88,0.88,0.649,0.649,0.649,0.92,0.756,0
.649,0.649,0.649'+#13#10+
'60,1.674,1.098,0.647,1.192,1.192,0.879,0.879,0.879,0.879,0.647,0.647,0.647,0.919,0.
755,0.647,0.647,0.647'+#13#10+
'61,1.674,1.097,0.646,1.191,1.191,0.877,0.877,0.877,0.877,0.646,0.646,0.646,0.917,0.
753,0.646,0.646,0.646'+#13#10+
'62,1.673,1.095,0.644,1.19,1.19,0.876,0.876,0.876,0.876,0.644,0.644,0.644,0.916,0.75
2,0.644,0.644,0.644'+#13#10+
'63,1.673,1.094,0.642,1.189,1.189,0.874,0.874,0.874,0.874,0.642,0.642,0.642,0.914,0.
75,0.642,0.642,0.642'+#13#10+
'64,1.672,1.093,0.64,1.188,1.188,0.873,0.873,0.873,0.873,0.64,0.64,0.64,0.913,0.748,
0.64,0.64,0.64'+#13#10+
'65,1.671,1.092,0.639,1.187,1.187,0.871,0.871,0.871,0.871,0.639,0.639,0.639,0.911,0.
747,0.639,0.639,0.639'+#13#10+
'66,1.671,1.09,0.637,1.185,1.185,0.87,0.87,0.87,0.87,0.637,0.637,0.637,0.91,0.745,0.
637,0.637,0.637'+#13#10+
'67,1.67,1.089,0.635,1.184,1.184,0.868,0.868,0.868,0.868,0.635,0.635,0.635,0.908,0.7
43,0.635,0.635,0.635'+#13#10+
'68,1.669,1.088,0.633,1.183,1.183,0.867,0.867,0.867,0.867,0.633,0.633,0.633,0.907,0.
742,0.633,0.633,0.633'+#13#10+
'69,1.668,1.086,0.632,1.182,1.182,0.865,0.865,0.865,0.865,0.632,0.632,0.632,0.905,0.
74,0.632,0.632,0.632'+#13#10+
'70,1.667,1.085,0.63,1.18,1.18,0.863,0.863,0.863,0.863,0.63,0.63,0.63,0.904,0.738,0.
63,0.63,0.63'+#13#10+
'71,1.666,1.083,0.628,1.178,1.178,0.861,0.861,0.861,0.861,0.628,0.628,0.628,0.902,0.
736,0.628,0.628,0.628'+#13#10+
'72,1.664,1.081,0.625,1.177,1.177,0.859,0.859,0.859,0.859,0.625,0.625,0.625,0.9,0.73
4,0.625,0.625,0.625'+#13#10+
'73,1.663,1.079,0.623,1.175,1.175,0.857,0.857,0.857,0.857,0.623,0.623,0.623,0.898,0.
732,0.623,0.623,0.623'+#13#10+
'74,1.661,1.077,0.621,1.173,1.173,0.855,0.855,0.855,0.855,0.621,0.621,0.621,0.896,0.
73,0.621,0.621,0.621'+#13#10+
'75,1.66,1.075,0.619,1.171,1.171,0.853,0.853,0.853,0.853,0.619,0.619,0.619,0.894,0.7
28,0.619,0.619,0.619'+#13#10+
'76,1.658,1.073,0.616,1.169,1.169,0.851,0.851,0.851,0.851,0.616,0.616,0.616,0.892,0.
725,0.616,0.616,0.616'+#13#10+
'77,1.656,1.071,0.614,1.167,1.167,0.849,0.849,0.849,0.849,0.614,0.614,0.614,0.89,0.7
23,0.614,0.614,0.614'+#13#10+
'78,1.655,1.069,0.612,1.165,1.165,0.847,0.847,0.847,0.847,0.612,0.612,0.612,0.887,0.
```

```
721,0.612,0.612,0.612'+#13#10+
'79,1.654,1.068,0.611,1.164,1.164,0.846,0.846,0.846,0.846,0.611,0.611,0.611,0.886,0.
72,0.611,0.611,0.611'+#13#10+
'80,1.654,1.068,0.611,1.164,1.164,0.846,0.846,0.846,0.846,0.611,0.611,0.611,0.886,0.
72,0.611,0.611,0.611'
;


PBSens1 :=
'0,3.612,2.349,1.152,2.566,2.566,2.566,1.724,1.724,1.724,1.152,1.152,1.152,1.853,1.3
59,0.968,0.968,0.968'+#13#10+
'1,2.468,1.826,1.228,1.926,1.926,1.926,1.517,1.517,1.517,1.228,1.228,1.228,1.573,1.3
34,1.141,1.141,1.141'+#13#10+
'2,2.645,1.881,1.250,2.047,2.047,2.047,1.570,1.570,1.570,1.250,1.250,1.250,1.644,1.3
56,1.177,1.177,1.177'+#13#10+
'3,2.473,1.812,1.242,1.952,1.952,1.952,1.537,1.537,1.537,1.242,1.242,1.242,1.604,1.3
37,1.155,1.155,1.155'+#13#10+
'4,2.480,1.813,1.219,1.937,1.937,1.937,1.514,1.514,1.514,1.219,1.219,1.219,1.574,1.3
19,1.138,1.138,1.138'+#13#10+
'5,2.656,1.876,1.254,2.032,2.032,2.032,1.577,1.577,1.577,1.254,1.254,1.254,1.632,1.3
71,1.188,1.188,1.188'+#13#10+
'6,2.340,1.650,1.071,1.756,1.756,1.756,1.369,1.369,1.369,1.071,1.071,1.071,1.432,1.1
94,0.982,0.982,0.982'
;

PBSens2 :=
'0,3.612,2.349,1.259,2.566,2.566,2.566,1.724,1.724,1.724,1.259,1.259,1.259,1.853,1.3
59,0.968,0.968,0.968'+#13#10+
'1,2.468,1.826,1.281,1.926,1.926,1.926,1.517,1.517,1.517,1.281,1.281,1.281,1.573,1.3
34,1.141,1.141,1.141'+#13#10+
'2,2.645,1.881,1.311,2.047,2.047,2.047,1.570,1.570,1.570,1.311,1.311,1.311,1.644,1.3
56,1.177,1.177,1.177'+#13#10+
'3,2.473,1.812,1.296,1.952,1.952,1.952,1.537,1.537,1.537,1.296,1.296,1.296,1.604,1.3
37,1.155,1.155,1.155'+#13#10+
'4,2.480,1.813,1.274,1.937,1.937,1.937,1.514,1.514,1.514,1.274,1.274,1.274,1.574,1.3
19,1.138,1.138,1.138'+#13#10+
'5,2.656,1.876,1.314,2.032,2.032,2.032,1.577,1.577,1.577,1.314,1.314,1.314,1.632,1.3
71,1.188,1.188,1.188'+#13#10+
'6,2.340,1.650,1.126,1.756,1.756,1.756,1.369,1.369,1.369,1.126,1.126,1.126,1.432,1.1
94,0.982,0.982,0.982';
;

PBSens3 :=
'0,3.156,2.119,0.980,2.234,2.234,2.234,1.608,1.608,1.608,0.980,0.980,0.980,1.675,1.2
61,0.980,0.980,0.980'+#13#10+
'1,2.228,1.679,1.141,1.778,1.778,1.778,1.441,1.441,1.441,1.141,1.141,1.141,1.485,1.2
68,1.141,1.141,1.141'+#13#10+
'2,2.350,1.749,1.160,1.849,1.849,1.849,1.479,1.479,1.479,1.160,1.160,1.160,1.523,1.3
26,1.160,1.160,1.160'+#13#10+
'3,2.254,1.690,1.142,1.777,1.777,1.777,1.438,1.438,1.438,1.142,1.142,1.142,1.494,1.2
```

```
91,1.142,1.142,1.142'+#13#10+
'4,2.270,1.711,1.156,1.772,1.772,1.772,1.446,1.446,1.446,1.156,1.156,1.156,1.485,1.2
83,1.156,1.156,1.156'+#13#10+
'5,2.341,1.774,1.195,1.870,1.870,1.870,1.512,1.512,1.512,1.195,1.195,1.195,1.541,1.3
15,1.195,1.195,1.195'+#13#10+
'6,2.097,1.527,1.001,1.602,1.602,1.602,1.276,1.276,1.276,1.001,1.001,1.001,1.317,1.1
25,1.001,1.001,1.001';
;


end.
```