```
unit frmTestBin;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Generics.Collections, Math;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

  TTest = class(Tobject)
    id : integer;
    BaseBin,NewBin : integer;
  end;

  TPWS_p_valuesRec = record
    p_lsl: integer;

    p_inventory: integer;
    p_tap_nine: integer;
    p_tap_annual: integer;
    p_tap_triennial: integer;

    p_spec_req: integer;

    p_wqp_annual: integer;
    p_wqp_triennial: integer;
    p_wqp_six_red: integer;

    p_b3: integer;

    pbaseph: integer;
    pbasepo4: integer;
    pbasephpo4: integer;
    baselinepo4dose: integer;
    baselineph_wph: integer;
    baselineph_woph: integer;
    baselineph_wocct: integer;
    baselineph_wpo4ph: integer;
```

```
    perc_lsl: double;

    ppBin1: double;
    ppBin2: double;
    ppBin3: double;
    ppBin4: double;
    ppBin5: double;
    ppAdjBin1: double;
    ppAdjBin2: double;
    ppAdjBin3: double;
    ppAdjBin4: double;
    ppAdjBin5: double;

    pp90aboveal10_1: double;
    pp90aboveal10_2: double;
    pp90aboveal10_3: double;
    pp90aboveal10_4: double;
    pp90aboveal10_5: double;
    pp90aboveal12_1: double;
    pp90aboveal12_2: double;
    pp90aboveal12_3: double;
    pp90aboveal12_4: double;
    pp90aboveal12_5: double;
    pp90aboveal15_1: double;
    pp90aboveal15_2: double;
    pp90aboveal15_3: double;
    pp90aboveal15_4: double;
    pp90aboveal15_5: double;
    pp90aboveal5_1: double;
    pp90aboveal5_2: double;
    pp90aboveal5_3: double;
    pp90aboveal5_4: double;
    pp90aboveal5_5: double;
  end;


  TPWSReplicator = class(TObject)
    function AssignOptionBin2(LSL: integer; p_values: TPWS_p_valuesRec; baseBin:
integer; Option: string): integer;
  end;



var
  Form1: TForm1;

implementation

{$R *.dfm}
```

```
function iiRandom(IsBaseline: boolean): double;
begin
  result := Random();
end;


function TPWSReplicator.AssignOptionBin2(LSL: integer; p_values: TPWS_p_valuesRec;
baseBin: integer; Option: string): integer;
var
  r,v,tot,r2: double;
  i: integer;
  PAdjBin, NewProbs, AdjBin, Bin: array[1..5] of double;
begin
  if LSL = 0 then begin
    result := baseBin;
    exit;
  end;

  r := iiRandom(false);
  Result:=999;
  fillchar(PAdjBin, sizeOf(PAdjBin),0);
  AdjBin[1] := p_values.ppAdjBin1; AdjBin[2] := p_values.ppAdjBin2; AdjBin[3] :=
p_values.ppAdjBin3;
  AdjBin[4] := p_values.ppAdjBin4; AdjBin[5] := p_values.ppAdjBin5;
  Bin[1] := p_values.ppBin1; Bin[2] := p_values.ppBin2; Bin[3] := p_values.ppBin3;
  Bin[4] := p_values.ppBin4; Bin[5] := p_values.ppBin5;

  //WHAT HAPPENS IF AdjBin[1]<Bin[1] - is it possible?
  NewProbs[1] := 0;
  for i := 2 to 5 do
    NewProbs[i] := AdjBin[i] / (1- AdjBin[1]);

  //find movement to Bin 1 and adjust remaining mass---
  tot := 0;
  for i:=1 to BaseBin do begin
    tot := tot + Bin[i];
    if (tot<=AdjBin[1]) then begin
      //everyone moves so short circuit out of here...
      if (BaseBin = i) then begin
        result := 1;
        exit;
      end;
    end else begin
      // calc remainder and folks going to bin 1.
      v := Bin[i] - (tot - AdjBin[1]);
      if (v>0) then begin
        //test to see if the portion should move....
        if BaseBin<>i then continue;
        r2 := iiRandom(false);
        if (r2 < v/Bin[i]) then begin
```

```
          result := 1;
          exit;
        end;
      end;
    end;
  end;

  //if we made it here - just distribute based on altered probs for 2-5
  v:=0;
  for i := 2 to 5 do v := v + NewProbs[i];
  PAdjBin[1] := 0;
  for i := 2 to 5 do PAdjBin[i] := PAdjBin[i-1] + NewProbs[i]/v;
  for i := 2 to BaseBin do begin
    if r<=PAdjBin[i] then begin
      result := i;
      exit;
    end;
    Result:=5; //?????  Fix this nonsense....
  end;
end;


procedure TForm1.Button1Click(Sender: TObject);
var P: TPWS_p_valuesRec;
    T: TPWSReplicator;
    ob,i,j,cnt: integer;
    D: TobjectList<TTest>;
    r: double;
    AdjBin,Bin,pp,bcnt,ocnt: array[1..5] of double;
    TT: TTest;
    TSW: TStreamWriter;
    tto,ttb: double;
begin
  cnt:=20000;
  Randomize;

  T:=TPWSReplicator.Create;
  D:=TobjectList<TTest>.create(true);
  P.ppAdjBin1 := 0.26;
  P.ppAdjBin2 := 0.065;
  P.ppAdjBin3 := 0.084;
  P.ppAdjBin4 := 0.228;
  P.ppAdjBin5 := 0.363;

  P.ppBin1 := 0.085;
  P.ppBin2 := 0.043;
  P.ppBin3 := 0.054;
  P.ppBin4 := 0.241;
  P.ppBin5 := 0.576;
  pp[1]:=P.ppBin1;
```

```
  pp[2] := pp[1] + P.ppBin2;
  pp[3] := pp[2] + P.ppBin3;
  pp[4] := pp[3] + P.ppBin4;
  pp[5] := pp[4] + P.ppBin5;

  AdjBin[1] := p.ppAdjBin1; AdjBin[2] := p.ppAdjBin2; AdjBin[3] := p.ppAdjBin3;
  AdjBin[4] := p.ppAdjBin4; AdjBin[5] := p.ppAdjBin5;
  Bin[1] := p.ppBin1; Bin[2] := p.ppBin2; Bin[3] := p.ppBin3;
  Bin[4] := p.ppBin4; Bin[5] := p.ppBin5;


  for i:=1 to cnt do begin
    r:=iiRandom(false);
    for j := 1 to 5 do begin
      if r<pp[j] then begin
        TT:=TTest.Create;
        TT.id := i;
        TT.BaseBin := j;
        TT.NewBin := -1;
        D.Add(tt);
        break;
      end;
    end;
  end;

  for tt in D do begin
    tt.NewBin:=T.AssignOptionBin2(1,p,tt.BaseBin,'');
  end;

  TSW:=TStreamWriter.Create('out.csv');
  TSW.WriteLine('id,base,option');
  for tt in D do begin
    TSW.WriteLine('%d,%d,%d',[tt.id,tt.BaseBin,tt.NewBin]);

  end;
  TSW.Free;

  fillchar(ocnt,sizeof(ocnt),0);
  fillchar(bcnt,sizeof(bcnt),0);
  for tt in D do begin
    bcnt[tt.BaseBin]:=bcnt[tt.BaseBin]+1;
    if tt.NewBin < 999 then
      ocnt[tt.NewBin]:=ocnt[tt.NewBin]+1;
  end;
  tto:=0; ttb:=0;
  for i := 1 to 5 do begin
    Memo1.Lines.Add(format('%d,%g,%g,%g,%g',[i,
bin[i],bcnt[i]/cnt,adjbin[i],ocnt[i]/cnt]));
    tto:=tto+ocnt[i]/cnt;
    ttb:=ttb+bcnt[i]/cnt;
```

```
  end;
 Memo1.Lines.Add('');
 Memo1.Lines.Add(format('%g,%g',[ttb,tto]));



  D.Free;
  T.Free;
end;

end.
```