

```

unit CCTCostEquations;

interface

uses SysUtils, StrUtils, Generics.Collections, Classes, Contnrs,
    DB, DBClient, SafewaterUncertBucket, LCRGlobals, LCRConfig;

type
    TCCTCostEquations = class
    private
        arrBaselineph_wph: array[1..2,1..3] of double;
        arrBaselineph_woph: array[1..2,1..5] of double;
        arrBaselineph_wpo4ph: array[1..2,1..6] of double;

        mnBaselinepo4dose: double;
        mnBaselineph_w: array[1..2] of double;
        mnBaselineph_wo: array[1..2] of double;

        cdsCostEquations: TClientDataset;

        FuncFormOM: integer;
        C1OM, C2OM, C3OM, C4OM, C5OM, C6OM, C7OM, C8OM, C9OM, C10OM: double;
        FuncFormCap: integer;
        C1Cap, C2Cap, C3Cap, C4Cap, C5Cap, C6Cap, C7Cap, C8Cap, C9Cap, C10Cap: double;

        procedure MakeCdsCostEquations;
        function sDFlowSize: string;
        function sAFlowSize: string;
        function sSourceWater: string;
        function GetCCTCostOM: double;
        function GetCCTCostCap: double;

        procedure
ListCds(sAction,sTechnology,sSourceWater,sSizeCategory,sCompLevel,sCostType: string;
        fStartingPH, fEndingPH, fPO4Dose: string);
    public
        UsefulLifeOM: double;
        UsefulLifeCap: double;
        iSourceWater: integer;
        iSystemSize: integer;
        iBaselinepo4dose: integer;
        iBaselineph_wph: integer;
        iBaselineph_woph: integer;
        iBaselineph_wocct: integer;
        iBaselineph_wpo4ph: integer;

        pbaseph: integer;
        pbasepo4: integer;
        pbasephpo4: integer;

```

```
arrBaselinepo4dose: array[1..3] of double;  
arrBaselineP: array[0..3] of double;  
arrBaselineph_wocct: array[1..2,1..4] of double;
```

```
DFlowEP, AFlowEP: double;  
EntryPoints: integer;
```

```
HasExistingEquation, HasAdjustEquation, HasNewEquations,  
  HasFindAndFixEquation: boolean;
```

```
CCTCostEquationLevel: string;
```

```
constructor Create;  
destructor Destroy; override;
```

```
procedure readSpreadSheet(filename: string);  
procedure ExistingCCT;  
procedure AdjustCCT(targetph, targetpo4: double);  
procedure NewCCT(targetph, targetpo4: double);  
procedure FindAndFixCCT(CCTB: integer);  
function ComputeOMCost: double; overload;  
function ComputeOMCost(nEPs: integer): double; overload;  
function ComputeCapitalCost: double; overload;  
function ComputeCapitalCost(nEPs: integer): double; overload;  
end;
```

```
implementation
```

```
uses VCL.FlexCel.Core, FlexCel.XlsAdapter, Math;
```

```
{ TCCTCostEquations }
```

```
procedure TCCTCostEquations.AdjustCCT(targetph, targetpo4: double);
```

```
var
```

```
  sBaselinepo4dose: string;  
  sStartph: string;  
  sEndph: string;  
  sTechnology: string;  
  sCostType: string;  
  sCompLevel: string;  
  sAction: string;
```

```
  found: boolean;  
  sErrLine: string;
```

```
begin
```

```
  found := false;  
  sAction := 'Adjust CCT';  
  sCompLevel := CCTCostEquationLevel;
```

```
  if pbaseph = 1 then
```

```

begin
  sTechnology := 'Modify PH';
  sCostType := 'Annual O&M';

  sStartph := floattostr(arrBaselineph_wph[iSourceWater,iBaselineph_wph] - 0.5);
  sEndph := '9.2';

  cdsCostEquations.IndexName := 'IdxNoDose';
  found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph]);
end
else if pbasepo4 = 1 then
begin
  sTechnology := 'Add P04 with PH Post Treatment';
  sCostType := 'Annual O&M';

  sBaselinepo4dose := '3.2';
  sStartph := arrBaselineph_woph[iSourceWater,iBaselineph_woph].ToString;
  sEndph := sStartph;

  cdsCostEquations.IndexName := 'IdxAll';
  found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
end
else if pbasephpo4 = 1 then
begin
  sTechnology := 'Add P04 and Modify PH';
  sCostType := 'Annual O&M';

  sBaselinepo4dose := targetpo4.ToString;
  sStartph := floattostr(arrBaselineph_wpo4ph[iSourceWater,iBaselineph_wP04Ph] -
0.5);
  sEndph := arrBaselineph_wpo4ph[iSourceWater,iBaselineph_wP04Ph].ToString;
  if 7.2 > arrBaselineph_wpo4ph[iSourceWater,iBaselineph_wP04Ph] then sEndph :=
'7.2';

  cdsCostEquations.IndexName := 'IdxAll';
  found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
end;

  sErrLine := 'AdjustCCT equation: ' + sAction + ', ' + sTechnology + ', ' +
sCostType + ', ' + sSourceWater + ', ' +
              sAFlowSize + ', ' +
              sBaselinepo4dose + ', ' + sStartph + ', ' + sEndph;

  if found then

```

```

begin
    UsefulLifeOM := cdsCostEquations.FieldByName('UsefulLife').AsFloat;
    C10M := cdsCostEquations.FieldByName('C1').AsFloat;
    C20M := cdsCostEquations.FieldByName('C2').AsFloat;
    C30M := cdsCostEquations.FieldByName('C3').AsFloat;
    C40M := cdsCostEquations.FieldByName('C4').AsFloat;
    C50M := cdsCostEquations.FieldByName('C5').AsFloat;
    C60M := cdsCostEquations.FieldByName('C6').AsFloat;
    C70M := cdsCostEquations.FieldByName('C7').AsFloat;
    C80M := cdsCostEquations.FieldByName('C8').AsFloat;
    C90M := cdsCostEquations.FieldByName('C9').AsFloat;
    C100M := cdsCostEquations.FieldByName('C10').AsFloat;
    FuncFormOM := cdsCostEquations.FieldByName('FunctionalForm').AsInteger;
end
else
    raise Exception.Create(sErrLine);

if pbaseph = 1 then
begin
    sTechnology := 'Modify PH';
    sCostType := 'Capital Retro';

    sStartph := floattostr(arrBaselineph_wph[iSourceWater,iBaselineph_wph] - 0.5);
    sEndph := '9.2';

    cdsCostEquations.IndexName := 'IdxNoDose';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph]);
end
else if pbasepo4 = 1 then
begin
    sTechnology := 'Add P04 with PH Post Treatment';
    sCostType := 'Capital Retro';

    sBaselinepo4dose := '3.2';
    sStartph := arrBaselineph_woph[iSourceWater,iBaselineph_woph].ToString;
    sEndph := sStartph;

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
end
else if pbasephpo4 = 1 then
begin
    sTechnology := 'Add P04 and Modify PH';
    sCostType := 'Capital Retro';

    sBaselinepo4dose := targetpo4.ToString;

```

```

    sStartph := floattostr(arrBaselineph_wpo4ph[iSourceWater,iBaselineph_wPO4Ph] -
0.5);
    sEndph := arrBaselineph_wpo4ph[iSourceWater,iBaselineph_wPO4Ph].ToString;
    if 7.2 > arrBaselineph_wpo4ph[iSourceWater,iBaselineph_wPO4Ph] then sEndph :=
'7.2';

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
    end;

    sErrLine := 'AdjustCCT equation: ' + sAction + ', ' + sTechnology + ', ' +
sCostType + ', ' + sSourceWater + ', ' +
                sAFlowSize + ', ' +
                sBaselinepo4dose + ', ' + sStartph + ', ' + sEndph;

    if found then
    begin
        UsefulLifeCap := cdsCostEquations.FieldByName('UsefulLife').AsFloat;
        C1Cap := cdsCostEquations.FieldByName('C1').AsFloat;
        C2Cap := cdsCostEquations.FieldByName('C2').AsFloat;
        C3Cap := cdsCostEquations.FieldByName('C3').AsFloat;
        C4Cap := cdsCostEquations.FieldByName('C4').AsFloat;
        C5Cap := cdsCostEquations.FieldByName('C5').AsFloat;
        C6Cap := cdsCostEquations.FieldByName('C6').AsFloat;
        C7Cap := cdsCostEquations.FieldByName('C7').AsFloat;
        C8Cap := cdsCostEquations.FieldByName('C8').AsFloat;
        C9Cap := cdsCostEquations.FieldByName('C9').AsFloat;
        C10Cap := cdsCostEquations.FieldByName('C10').AsFloat;
        FuncFormCap := cdsCostEquations.FieldByName('FunctionalForm').AsInteger;
    end
    else
        raise Exception.Create(sErrLine);

    HasAdjustEquation := true;
end;

{
Dollar year adjustments from Jerry email 7/28/23
From To      Multiplier
2021 2022      1.069
2020 2022      1.095
2021 2020      0.977

For CCT cost equations, use 1.069
}

function TCCTCostEquations.ComputeCapitalCost: double;
begin

```

```

    Result := GetCCTCostCap() * EntryPoints * 1.069;
end;

function TCCTCostEquations.ComputeOMCost: double;
begin
    Result := GetCCTCostOM() * EntryPoints * 1.069;
end;

function TCCTCostEquations.ComputeCapitalCost(nEPs: integer): double;
begin
    Result := GetCCTCostCap() * nEPs * 1.069;
end;

function TCCTCostEquations.ComputeOMCost(nEPs: integer): double;
begin
    Result := GetCCTCostOM() * nEPs * 1.069;
end;

constructor TCCTCostEquations.Create;
begin
    arrBaselinepo4dose[1] := 0.525;
    arrBaselinepo4dose[2] := 1.5;
    arrBaselinepo4dose[3] := 2.65;

    {
        P limit based on P04Dose
        P limit = P04Dose * 0.326318
        used in POTW cost calculation
    }
    arrBaselineP[0] := 0;
    arrBaselineP[1] := 0.17;
    arrBaselineP[2] := 0.49;
    arrBaselineP[3] := 0.86;

    // arrays are SourceType, Probability
    // SourceType: 1 = GW, 2 = SW

    arrBaselineph_wph[1,1] := 8.2;
    arrBaselineph_wph[1,2] := 8.3;
    arrBaselineph_wph[1,3] := 8.8;
    arrBaselineph_wph[2,1] := 8.2;
    arrBaselineph_wph[2,2] := 8.3;
    arrBaselineph_wph[2,3] := 8.9;

    arrBaselineph_woph[1,1] := 6.3;
    arrBaselineph_woph[1,2] := 6.6;
    arrBaselineph_woph[1,3] := 7.3;
    arrBaselineph_woph[1,4] := 8.0;
    arrBaselineph_woph[1,5] := 8.6;
    arrBaselineph_woph[2,1] := 6.3;

```

```
arrBaselineph_woph[2,2] := 6.8;  
arrBaselineph_woph[2,3] := 7.4;  
arrBaselineph_woph[2,4] := 7.9;  
arrBaselineph_woph[2,5] := 8.4;
```

```
arrBaselineph_wocct[1,1] := 7.0;  
arrBaselineph_wocct[1,2] := 7.3;  
arrBaselineph_wocct[1,3] := 8.0;  
arrBaselineph_wocct[1,4] := 8.6;  
arrBaselineph_wocct[2,1] := 7.0;  
arrBaselineph_wocct[2,2] := 7.4;  
arrBaselineph_wocct[2,3] := 7.9;  
arrBaselineph_wocct[2,4] := 8.4;
```

```
arrBaselineph_wpo4ph[1,1] := 6.3;  
arrBaselineph_wpo4ph[1,2] := 6.8;  
arrBaselineph_wpo4ph[1,3] := 7.3;  
arrBaselineph_wpo4ph[1,4] := 7.8;  
arrBaselineph_wpo4ph[1,5] := 8.3;  
arrBaselineph_wpo4ph[1,6] := 8.8;  
arrBaselineph_wpo4ph[2,1] := 6.3;  
arrBaselineph_wpo4ph[2,2] := 6.8;  
arrBaselineph_wpo4ph[2,3] := 7.2;  
arrBaselineph_wpo4ph[2,4] := 7.7;  
arrBaselineph_wpo4ph[2,5] := 8.3;  
arrBaselineph_wpo4ph[2,6] := 8.9;
```

```
mnBaselinepo4dose := 1.81;
```

```
mnBaselineph_w[1] := 7.5;  
mnBaselineph_w[2] := 7.5;
```

```
mnBaselineph_wo[1] := 7.0;  
mnBaselineph_wo[2] := 7.1;
```

```
cdsCostEquations := TClientDataset.Create(nil);  
MakeCDSCostEquations;
```

```
HasExistingEquation := false;  
HasAdjustEquation := false;  
HasNewEquations := false;  
HasFindAndFixEquation := false;  
end;
```

```
destructor TCCTCostEquations.Destroy;  
begin  
  cdsCostEquations.Close;  
  cdsCostEquations.Free;
```

```
inherited;
```

```

end;

procedure TCCTCostEquations.ExistingCCT;
var
  sBaselinepo4dose: string;
  sStartph: string;
  sEndph: string;
  sTechnology: string;
  sCostType: string;
  sCompLevel: string;
  sAction: string;

  found: boolean;
  sErrLine: string;

  sw, ss: string;
begin
  found := false;
  sAction := 'Existing CCT';
  sCompLevel := CCTCostEquationLevel;

  if pbaseph = 1 then
  begin
    sTechnology := 'Modify PH';
    sCostType := 'Annual O&M';

    sStartph := floattostr(arrBaselineph_wph[iSourceWater,iBaselineph_wph] - 0.5);
    sEndph := arrBaselineph_wph[iSourceWater,iBaselineph_wph].ToString;

    sw := sSourceWater;
    ss := SAFlowSize;

    cdsCostEquations.IndexName := 'IdxNoDose';
    found := cdsCostEquations.FindKey([sTechnology, sw, ss, sCompLevel, sCostType,
                                         sStartph, sEndph]);
  end
  else if pbasepo4 = 1 then
  begin
    sTechnology := 'Add P04 with PH Post Treatment';
    sCostType := 'Annual O&M';

    sBaselinepo4dose := arrBaselinePo4dose[iBaselinepo4dose].ToString;
    sStartph := arrBaselineph_woph[iSourceWater,iBaselineph_woph].ToString;
    sEndph := sStartph;

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
                                         sCompLevel, sCostType,
                                         sStartph, sEndph, sBaselinepo4dose]);
  end
end

```



```

else if pbasephpo4 = 1 then
begin
    sTechnology := 'Add P04 and Modify PH';
    sCostType := 'Annual O&M';

    sBaselinepo4dose := arrBaselinePo4dose[iBaselinepo4dose].ToString;
    sStartph := floattostr(arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph] -
0.5);
    sEndph := arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph].ToString;

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
    end;

    sErrLine := 'ExistingCCT equation: ' + sAction + ', ' + sTechnology + ', ' +
sCostType + ', ' + sSourceWater + ', ' +
                sAFlowSize + ', ' +
                sBaselinepo4dose + ', ' + sStartph + ', ' + sEndph;

    if found then
    begin
        UsefulLifeOM := cdsCostEquations.FieldByName('UsefulLife').AsFloat;
        C10M := cdsCostEquations.FieldByName('C1').AsFloat;
        C20M := cdsCostEquations.FieldByName('C2').AsFloat;
        C30M := cdsCostEquations.FieldByName('C3').AsFloat;
        C40M := cdsCostEquations.FieldByName('C4').AsFloat;
        C50M := cdsCostEquations.FieldByName('C5').AsFloat;
        C60M := cdsCostEquations.FieldByName('C6').AsFloat;
        C70M := cdsCostEquations.FieldByName('C7').AsFloat;
        C80M := cdsCostEquations.FieldByName('C8').AsFloat;
        C90M := cdsCostEquations.FieldByName('C9').AsFloat;
        C100M := cdsCostEquations.FieldByName('C10').AsFloat;
        FuncFormOM := cdsCostEquations.FieldByName('FunctionalForm').AsInteger;
    end
    else
        raise Exception.Create(sErrLine);

    HasExistingEquation := true;
end;

procedure TCCTCostEquations.FindAndFixCCT(CCTB: integer);
var
    sBaselinepo4dose: string;
    sStartph: string;
    sEndph: string;
    sTechnology: string;
    sCostType: string;
    sCompLevel: string;

```

```

sAction: string;

found: boolean;
sErrLine: string;
begin
    found := false;
    sAction := 'Find & Fix';
    sCompLevel := CCTCostEquationLevel;

    if (pbaseph = 1) and (CCTB = 1) then
    begin
        sTechnology := 'Modify PH';
        sCostType := 'Annual O&M';

        sStartph := floattostr(arrBaselineph_wph[iSourceWater,iBaselineph_wph] - 0.5);
        sEndph := '9.4';

        cdsCostEquations.IndexName := 'IdxNoDose';
        found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph]));
    end
    else
    if (pbaseph = 1) and (CCTB = 0) then
    begin
        sTechnology := 'Modify PH';
        sCostType := 'Annual O&M';

        sStartph := floattostr(arrBaselineph_wocct[iSourceWater,iBaselineph_wocct]);
        sEndph := '9.4';

        cdsCostEquations.IndexName := 'IdxNoDose';
        found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph]));
    end
    else if pbasepo4 = 1 then
    begin
        if arrBaselineph_woPh[iSourceWater,iBaselineph_woph] < 7.5 then
        begin
            sTechnology := 'Add P04 and Modify PH';
            sCostType := 'Annual O&M';

            sBaselinepo4dose := '3.2';
            sStartph := arrBaselineph_woph[iSourceWater,iBaselineph_woph].ToString;
            sEndph := '7.5';
        end
        else
        begin
            sTechnology := 'Add P04 with PH Post Treatment';

```

```

sCostType := 'Annual O&M';

sBaselinepo4dose := '3.2';
sStartph := arrBaselineph_woph[iSourceWater,iBaselineph_woph].ToString;
sEndph := sStartph;
end;

cdsCostEquations.IndexName := 'IdxAll';
found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
end
else if pbasephpo4 = 1 then
begin
sTechnology := 'Add P04 and Modify PH';
sCostType := 'Annual O&M';

sBaselinepo4dose := '3.2';
sStartph := floattostr(arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph] -
0.5);
if arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph] > 7.5 then
sEndph := arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph].ToString
else
sEndph := '7.5';

cdsCostEquations.IndexName := 'IdxAll';
found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
end;

sErrLine := 'FindAndFixCCT equation: ' + sAction + ', ' + sTechnology + ', ' +
sCostType + ', ' + sSourceWater + ', ' +
sAFlowSize + ', ' +
sBaselinepo4dose + ', ' + sStartph + ', ' + sEndph;

if found then
begin
UsefulLifeOM := cdsCostEquations.FieldName('UsefulLife').AsFloat;
C1OM := cdsCostEquations.FieldName('C1').AsFloat;
C2OM := cdsCostEquations.FieldName('C2').AsFloat;
C3OM := cdsCostEquations.FieldName('C3').AsFloat;
C4OM := cdsCostEquations.FieldName('C4').AsFloat;
C5OM := cdsCostEquations.FieldName('C5').AsFloat;
C6OM := cdsCostEquations.FieldName('C6').AsFloat;
C7OM := cdsCostEquations.FieldName('C7').AsFloat;
C8OM := cdsCostEquations.FieldName('C8').AsFloat;
C9OM := cdsCostEquations.FieldName('C9').AsFloat;
C10OM := cdsCostEquations.FieldName('C10').AsFloat;
FuncFormOM := cdsCostEquations.FieldName('FunctionalForm').AsInteger;

```

```

end
else
    raise Exception.Create(sErrLine);

if (pbaseph = 1) and (CCTB = 1) then
begin
    sTechnology := 'Modify PH';
    sCostType := 'Capital Retro';

    sStartph := floattostr(arrBaselineph_wph[iSourceWater,iBaselineph_wph] - 0.5);
    sEndph := '9.4';

    cdsCostEquations.IndexName := 'IdxNoDose';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph]);
end
else
if (pbaseph = 1) and (CCTB = 0) then
begin
    sTechnology := 'Modify PH';
    sCostType := 'Capital Retro';

    sStartph := floattostr(arrBaselineph_wocct[iSourceWater,iBaselineph_wocct]);
    sEndph := '9.4';

    cdsCostEquations.IndexName := 'IdxNoDose';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph]);
end
else if pbasepo4 = 1 then
begin
    if arrBaselineph_woph[iSourceWater,iBaselineph_woph] < 7.5 then
begin
        sTechnology := 'Add P04 and Modify PH';
        sCostType := 'Capital Retro';

        sBaselinepo4dose := '3.2';
        sStartph := arrBaselineph_woph[iSourceWater,iBaselineph_woph].ToString;
        sEndph := '7.5';
end
    else
begin
        sTechnology := 'Add P04 with PH Post Treatment';
        sCostType := 'Capital Retro';

        sBaselinepo4dose := '3.2';
        sStartph := arrBaselineph_woph[iSourceWater,iBaselineph_woph].ToString;
        sEndph := sStartph;
    end
end

```

```

end;

cdsCostEquations.IndexName := 'IdxAll';
found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
end
else if pbasephpo4 = 1 then
begin
    sTechnology := 'Add P04 and Modify PH';
    sCostType := 'Capital Retro';

    sBaselinepo4dose := '3.2';
    sStartph := floattostr(arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph] -
0.5);
    if arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph] > 7.5 then
        sEndph := arrBaselineph_wP04Ph[iSourceWater,iBaselineph_wpo4ph].ToString
    else
        sEndph := '7.5';

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
end;

sErrLine := 'FindAndFixCCT equation: ' + sAction + ', ' + sTechnology + ', ' +
sCostType + ', ' + sSourceWater + ', ' +
            sFlowSize + ', ' +
            sBaselinepo4dose + ', ' + sStartph + ', ' + sEndph;

if found then
begin
    UsefulLifeCap := cdsCostEquations.FieldByName('UsefulLife').AsFloat;
    C1Cap := cdsCostEquations.FieldByName('C1').AsFloat;
    C2Cap := cdsCostEquations.FieldByName('C2').AsFloat;
    C3Cap := cdsCostEquations.FieldByName('C3').AsFloat;
    C4Cap := cdsCostEquations.FieldByName('C4').AsFloat;
    C5Cap := cdsCostEquations.FieldByName('C5').AsFloat;
    C6Cap := cdsCostEquations.FieldByName('C6').AsFloat;
    C7Cap := cdsCostEquations.FieldByName('C7').AsFloat;
    C8Cap := cdsCostEquations.FieldByName('C8').AsFloat;
    C9Cap := cdsCostEquations.FieldByName('C9').AsFloat;
    C10Cap := cdsCostEquations.FieldByName('C10').AsFloat;
    FuncFormCap := cdsCostEquations.FieldByName('FunctionalForm').AsInteger;
end
else
    raise Exception.Create(sErrLine);

HasFindAndFixEquation := true;

```

```

end;

function TCCTCostEquations.GetCCTCostCap: double;
var
    EPDF: double;
    adj: boolean;
begin
    EPDF := DFlowEP;
    adj := false;

    if (DFlowEP > 162) then
        begin
            EPDF := 162;
            adj := true;
        end;

    case FuncFormCap of
        integer(ffPOW) : Result := C1Cap * power(EPDF,C2Cap);
        integer(ffLN)   : Result := C3Cap * ln(EPDF) + C4Cap;
        integer(ffExp)  : Result := C5Cap * exp(EPDF * C6Cap);
        integer(ffPoly): Result := C7Cap * intpower(EPDF,3) + C8Cap * intpower(EPDF,2) +
C9Cap * EPDF + C10Cap;
    else
        Result := 0;
    end;

    if adj then
        Result := Result * (DFlowEP / 162);
    end;

function TCCTCostEquations.GetCCTCostOM: double;
var
    EPADF: double;
    swAdj, gwAdj, adj: boolean;
begin
    EPADF := AFlowEP;
    swAdj := false;
    gwAdj := false;
    adj := false;

    if AFlowEP > 76 then
        begin
            EPADF := 76;
            adj := true;
        end;

    case FuncFormOM of
        integer(ffPOW) : Result := C1OM * power(EPADF,C2OM);
        integer(ffLN)   : Result := C3OM * ln(EPADF) + C4OM;
        integer(ffExp)  : Result := C5OM * exp(EPADF * C6OM);
    end;

```

```

integer(ffPoly): Result := C70M * intpower(EPADF,3) + C80M * intpower(EPADF,2) +
C90M * EPADF + C100M;
else
    Result := 0;
end;

if adj then
    Result := Result * (AFlowEP / 76);

end;

procedure
TCCTCostEquations.ListCds(sAction,sTechnology,sSourceWater,sSizeCategory,sCompLevel,
sCostType: string;
                        fStartingPH, fEndingPH, fPO4Dose: string);
var
    i: integer;
    Action, Technology, SourceWater, SizeCategory, CompLevel, CostType: string;
    StartingPH, EndingPH, PO4Dose: string;
    fnd: boolean;
begin
    cdsCostEquations.First;
    while not cdsCostEquations.Eof do
        begin
            Action := cdsCostEquations.FieldName('Action').AsString;
            Technology := cdsCostEquations.FieldName('Technology').AsString;
            SourceWater := cdsCostEquations.FieldName('SourceWater').AsString;
            SizeCategory := cdsCostEquations.FieldName('SizeCategory').AsString;
            CompLevel := cdsCostEquations.FieldName('CompLevel').AsString;
            CostType := cdsCostEquations.FieldName('CostType').AsString;
            StartingPH := cdsCostEquations.FieldName('StartingPH').AsString;
            EndingPH := cdsCostEquations.FieldName('EndingPH').AsString;
            PO4Dose := cdsCostEquations.FieldName('PO4Dose').AsString;

            fnd := false;
            if (Action = sAction) and (Technology = sTechnology) and (SourceWater =
sSourceWater) and (SizeCategory = sSizeCategory) and
                (CompLevel = sCompLevel) and (CostType = sCostType) then
                begin
                    if (StartingPH = fStartingPH) then
                        fnd := true;
                    end;

                    cdsCostEquations.Next;
                end;
        end;
    end;

procedure TCCTCostEquations.MakeCdsCostEquations;
begin
    with cdsCostEquations do begin

```

```

FieldDefs.Add('Action', ftString, 25);
FieldDefs.Add('Technology', ftString, 40);
FieldDefs.Add('StartingPH', ftString, 10);
FieldDefs.Add('EndingPH', ftString, 10);
FieldDefs.Add('PO4Dose', ftString, 10);
FieldDefs.Add('SourceWater', ftString, 5);
FieldDefs.Add('SizeCategory', ftString, 10);
FieldDefs.Add('CompLevel', ftString, 5);
FieldDefs.Add('CostType', ftString, 15);
FieldDefs.Add('UsefulLife', ftFloat);
FieldDefs.Add('C1', ftFloat);
FieldDefs.Add('C2', ftFloat);
FieldDefs.Add('C3', ftFloat);
FieldDefs.Add('C4', ftFloat);
FieldDefs.Add('C5', ftFloat);
FieldDefs.Add('C6', ftFloat);
FieldDefs.Add('C7', ftFloat);
FieldDefs.Add('C8', ftFloat);
FieldDefs.Add('C9', ftFloat);
FieldDefs.Add('C10', ftFloat);

```

```

FieldDefs.Add('FunctionalForm', ftInteger);

```

```

IndexDefs.Add('IdxAll', 'Technology;SourceWater;SizeCategory;CompLevel;CostType;StartingPH;EndingPH;PO4Dose', []);

```

```

IndexDefs.Add('IdxNoDose', 'Technology;SourceWater;SizeCategory;CompLevel;CostType;StartingPH;EndingPH', []);

```

```

    CreateDataSet;
    LogChanges := False;
end;

```

```

end;

```

```

procedure TCCTCostEquations.NewCCT(targetph, targetpo4: double);

```

```

var

```

```

    sBaselinepo4dose: string;
    sStartph: string;
    sEndph: string;
    sTechnology: string;
    sCostType: string;
    sCompLevel: string;
    sAction: string;

```

```

    found: boolean;
    sErrLine: string;

```

```

begin

```

```

    sAction := 'Install CCT';

```



```

sCompLevel := CCTCostEquationLevel;

if arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT] >= 8.4 then
begin
    sTechnology := 'Modify PH';
    sCostType := 'Total Capital';

    sBaselinepo4dose := '';
    sStartph := arrBaselineph_wocct[iSourceWater,iBaselineph_wocct].ToString;
    sEndph := '9.2';

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sDFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);

    pbaseph := 1;
end
else if (arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT] >= 7.2) and
        (arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT] < 8.4) then
begin
    sTechnology := 'Add P04 with PH Post Treatment';
    sCostType := 'Total Capital';

    sBaselinepo4dose := '3.2';
    sStartph := arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT].ToString;
    sEndph := sStartph;

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sDFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);

    pbasepo4 := 1;
end
else begin
    sTechnology := 'Add P04 and Modify PH';
    sCostType := 'Total Capital';

    sBaselinepo4dose := '3.2';
    sStartph := arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT].ToString;
    sEndph := '7.2';

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sDFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);

    pbasephpo4 := 1;
end;

```

```

sErrLine := 'NewCCT Cap equation: ' + sAction + ', ' + sTechnology + ', ' +
sCostType + ', ' + sSourceWater + ', ' +
          sAFlowSize + ', ' +
          sBaselinepo4dose + ', ' + sStartph + ', ' + sEndph;

if found then
begin
    UsefulLifeCap := cdsCostEquations.FieldByName('UsefulLife').AsFloat;
    C1Cap := cdsCostEquations.FieldByName('C1').AsFloat;
    C2Cap := cdsCostEquations.FieldByName('C2').AsFloat;
    C3Cap := cdsCostEquations.FieldByName('C3').AsFloat;
    C4Cap := cdsCostEquations.FieldByName('C4').AsFloat;
    C5Cap := cdsCostEquations.FieldByName('C5').AsFloat;
    C6Cap := cdsCostEquations.FieldByName('C6').AsFloat;
    C7Cap := cdsCostEquations.FieldByName('C7').AsFloat;
    C8Cap := cdsCostEquations.FieldByName('C8').AsFloat;
    C9Cap := cdsCostEquations.FieldByName('C9').AsFloat;
    C10Cap := cdsCostEquations.FieldByName('C10').AsFloat;
    FuncFormCap := cdsCostEquations.FieldByName('FunctionalForm').AsInteger;
end
else
    raise Exception.Create(sErrLine);

if arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT] >= 8.4 then
begin
    sTechnology := 'Modify PH';
    sCostType := 'Annual O&M';

    sBaselinepo4dose := '';
    sStartph := arrBaselineph_wocct[iSourceWater,iBaselineph_wocct].ToString;
    sEndph := '9.2';

    cdsCostEquations.IndexName := 'IdxAll';
    found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sDFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);

    pbaseph := 1;
end
else if (arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT] >= 7.2) and
        (arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT] < 8.4) then
begin
    sTechnology := 'Add P04 with PH Post Treatment';
    sCostType := 'Annual O&M';

    sBaselinepo4dose := '3.2';
    sStartph := arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT].ToString;
    sEndph := sStartph;

```

```

        cdsCostEquations.IndexName := 'IdxAll';
        found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
    end
    else begin
        sTechnology := 'Add P04 and Modify PH';
        sCostType := 'Annual O&M';

        sBaselinepo4dose := '3.2';
        sStartph := arrBaselineph_woCCT[iSourceWater,iBaselineph_woCCT].ToString;
        sEndph := '7.2';

        cdsCostEquations.IndexName := 'IdxAll';
        found := cdsCostEquations.FindKey([sTechnology, sSourceWater, sAFlowSize,
sCompLevel, sCostType,
                                sStartph, sEndph, sBaselinepo4dose]);
    end;

    sErrLine := 'NewCCT O&M equation: ' + sAction + ', ' + sTechnology + ', ' +
sCostType + ', ' + sSourceWater + ', ' +
        sAFlowSize + ', ' +
        sBaselinepo4dose + ', ' + sStartph + ', ' + sEndph;

    if found then
    begin
        UsefulLifeOM := cdsCostEquations.FieldByName('UsefulLife').AsFloat;
        C1OM := cdsCostEquations.FieldByName('C1').AsFloat;
        C2OM := cdsCostEquations.FieldByName('C2').AsFloat;
        C3OM := cdsCostEquations.FieldByName('C3').AsFloat;
        C4OM := cdsCostEquations.FieldByName('C4').AsFloat;
        C5OM := cdsCostEquations.FieldByName('C5').AsFloat;
        C6OM := cdsCostEquations.FieldByName('C6').AsFloat;
        C7OM := cdsCostEquations.FieldByName('C7').AsFloat;
        C8OM := cdsCostEquations.FieldByName('C8').AsFloat;
        C9OM := cdsCostEquations.FieldByName('C9').AsFloat;
        C10OM := cdsCostEquations.FieldByName('C10').AsFloat;
        FuncFormOM := cdsCostEquations.FieldByName('FunctionalForm').AsInteger;
    end
    else
        raise Exception.Create(sErrLine);

        HasNewEquations := true;
    end;

procedure TCCTCostEquations.readSpreadSheet(filename: string);
var
    xls: TExcelFile;
    r: integer;
begin

```

```

xls := TXLSFile.Create;
xls.Open(FileName);
xls.ActiveSheetByName := 'Equations';

for r := 2 to xls.RowCount do
begin
    cdsCostEquations.Append;

    cdsCostEquations.FieldByName('Action').AsString := xls.GetStringFromCell(r,1);
    cdsCostEquations.FieldByName('Technology').AsString :=
xls.GetStringFromCell(r,2);
    cdsCostEquations.FieldByName('StartingPH').AsString :=
xls.GetStringFromCell(r,3);
    cdsCostEquations.FieldByName('EndingPH').AsString := xls.GetStringFromCell(r,4);
    cdsCostEquations.FieldByName('P04Dose').AsString := xls.GetStringFromCell(r,5);
    cdsCostEquations.FieldByName('SourceWater').AsString :=
xls.GetStringFromCell(r,6);
    cdsCostEquations.FieldByName('SizeCategory').AsString :=
xls.GetStringFromCell(r,7);
    cdsCostEquations.FieldByName('CompLevel').AsString :=
xls.GetStringFromCell(r,8);
    cdsCostEquations.FieldByName('CostType').AsString := xls.GetStringFromCell(r,9);
    cdsCostEquations.FieldByName('UsefulLife').AsFloat :=
xls.GetCellValue(r,10).AsVariant;
    cdsCostEquations.FieldByName('C1').AsFloat := xls.GetCellValue(r,11).AsVariant;
    cdsCostEquations.FieldByName('C2').AsFloat := xls.GetCellValue(r,12).AsVariant;
    cdsCostEquations.FieldByName('C3').AsFloat := xls.GetCellValue(r,13).AsVariant;
    cdsCostEquations.FieldByName('C4').AsFloat := xls.GetCellValue(r,14).AsVariant;
    cdsCostEquations.FieldByName('C5').AsFloat := xls.GetCellValue(r,15).AsVariant;
    cdsCostEquations.FieldByName('C6').AsFloat := xls.GetCellValue(r,16).AsVariant;
    cdsCostEquations.FieldByName('C7').AsFloat := xls.GetCellValue(r,17).AsVariant;
    cdsCostEquations.FieldByName('C8').AsFloat := xls.GetCellValue(r,18).AsVariant;
    cdsCostEquations.FieldByName('C9').AsFloat := xls.GetCellValue(r,19).AsVariant;
    cdsCostEquations.FieldByName('C10').AsFloat := xls.GetCellValue(r,20).AsVariant;

    if xls.GetCellValue(r,11).AsVariant + xls.GetCellValue(r,12).AsVariant <> 0 then
        cdsCostEquations.FieldByName('FunctionalForm').AsInteger := 1
    else if xls.GetCellValue(r,13).AsVariant + xls.GetCellValue(r,14).AsVariant <> 0
then
        cdsCostEquations.FieldByName('FunctionalForm').AsInteger := 2
    else if xls.GetCellValue(r,15).AsVariant + xls.GetCellValue(r,16).AsVariant <> 0
then
        cdsCostEquations.FieldByName('FunctionalForm').AsInteger := 3
    else if xls.GetCellValue(r,17).AsVariant + xls.GetCellValue(r,18).AsVariant +
        xls.GetCellValue(r,19).AsVariant + xls.GetCellValue(r,20).AsVariant <> 0
then
        cdsCostEquations.FieldByName('FunctionalForm').AsInteger := 4;

    cdsCostEquations.Post;

```

```

end;

xls.Free;
end;

function TCCTCostEquations.sDFlowSize: string;
begin
    if DFlowEP < 1 then
        Result := 'Small'
    else if DFlowEP < 10 then
        Result := 'Medium'
    else
        Result := 'Large';
    end;

function TCCTCostEquations.sAFlowSize: string;
begin
    // GW
    if iSourceWater = 1 then
        begin
            if AFlowEP < 0.349 then
                Result := 'Small'
            else if AFlowEP < 4.481 then
                Result := 'Medium'
            else
                Result := 'Large';
            end
        end
    // SW
    else begin
        if AFlowEP < 0.355 then
            Result := 'Small'
        else if AFlowEP < 4.023 then
            Result := 'Medium'
        else
            Result := 'Large';
        end;
    end;
end;

function TCCTCostEquations.sSourceWater: string;
begin
    case iSourceWater of
        1: Result := 'GW';
        2: Result := 'SW';
    end;
end;

end.

```