



GAUTENG PROVINCE
EDUCATION
REPUBLIC OF SOUTH AFRICA

VOORBEREIDENDE EKSAMEN 2015
GEMEENSKAPLIKE EKSAMEN
GRAAD 12
MEMORANDUM/MERKGIDS

VAK	:	IT VRAESTEL 1
TYD	:	3 URE
PUNTE	:	150

Die memorandum/merkgids bestaan uit 21 bladsye

Leerling NAAM & VAN:		
VRAAG 1:	<u>49</u>	Onderwysershandtekening: _____ Datum: _____
VRAAG 2:	<u>58</u>	
VRAAG 3:	<u>43</u>	Moderatorshandtekening: _____ Datum: _____
TOTAAL :	<u>150</u>	
PERSENTASIE:	<u>100</u>	

VRAAG 1: MERKGIDS – ALGEMENE PROGRAMERINGVAARDIGHEDE

VRAAG	BESKRYWING	MAKS PUNTE	LEERDER PUNTE
1.1.1	[btnGenerateClick] redDisplay.Clear ✓ Heading Client Summary (or similar) added to redDisplay ✓ Open line catered for redisplay.Lines.Add("") ✓ <u>OR</u> #13 concatenated to heading. Client number stored from edtCustomer into variable ✓ declared with either unit OR class scope (globally declared) ✓ The correct headings ✓✓ were added to redDisplay and alignment (#9 with or without paragraph tabs) was used effectively ✓. Headings are as follows: • Customer number • Client budget • Purchase item • Payment method Budget displayed (from edtBudget) ✓ using FloatToStrF with second parameter (format) set to ffCurrency <u>OR</u> concatenated ('R') <u>OR</u> Format function used correctly. ✓ Purchase item is displayed successfully in redDisplay using cbxCategory & edtBESKRYWING : cbxCategory.Items[cbxCategory.ItemIndex] ✓+ ' ' ✓+ edtBESKRYWING.text ✓ Payment method displayed successfully in redDisplay from rgpPayment: rgpPayment.Items[rgpPayment.ItemIndex] ✓ Correct use of decision making structure (CASE <u>OR</u> if-else if <u>OR</u> sequential if-statements) ✓ referring to ordinal rgpPayment value of rgpPayment 0, 1 & 2 ✓ with correct corresponding outputs for each payment method message. ✓	17	
1.1.2	[ValidateClick] Correct If-else(if-else) ✓✓ <u>OR</u> if-else(if-if) structure used. NOTE: Same marks allocated for consecutive if-statement	8	

	structures. MUST constitute THREE outcomes! ✓ ✎ AND operator used ✓ correctly TRUE outcome of length (8) <u>and</u> first character match (#) ✓ ✎ <>, AND NOT <u>or</u> NOT used for invalid criteria. ✓ ✎ pnlValidate color change for valid (green) and invalid (red) criteria w.r.t. all three outcomes. ✓ ✎ Successful output caption for pnlValidate for valid or invalid scenarios. ✓		
1.1.3	[bmbRetryClick] ✎ Clear ALL edit components. ✓ ✎ Set the cbxCategory.ItemIndex to -1 <u>and</u> } Set rgpPayment.ItemIndex to -1 } ✓ ✎ Set the validation panel component to colour silver with no caption. ✓ ✎ Cursor must be set to the first edit (awaiting the customer number). ✓	4	
1.2.1	[btnImportClick] ✎ Use of the lblCustNum.<u>Caption</u>. ✓ ✎ Concatenate the globally declared customer number. Note that the variable must have been declared with unit/class scope. ✓	2	
1.2.2	✎ Calculations: ○ Weight multiplied ✓ by correct factors ✓ using CASE <u>or</u> nested if statements ✓ with ordinal value from Category listbox (itemindex) ✓ ○ Correct checklistbox (box sizes) item obtained ✓ using CASE or nested if statements ✓ after which correct boxes were determined. (3 criterias ✓) ○ Fragility taken into consideration and calculated after initial cost was determined according to box sizes and category. ✓ ○ Correct R/box calculations for final cost ✓ and stored into given array ✓ ✎ User-defined procedure constructed ✓ and used successfully with calculations ✓ ✎ Cleared all components in order to accompany user input information for parcel data that follows. Components cleared: ○ Checklistbox ○ SpinEdit ○ Checkbox ○ Listbox (✓ ✓ marks all cleared, ✓ mark some, 0 none) ✎ Message used ✓ to display parcel number ✓, <u>correct</u> parcel cost ✓ and customer number ✓.	18	
TOTAAL:			<u>49</u>

VRAAG 2: MERKGIDS – OBJEK GEORIENTEERDE PROGRAMERING

VRAAG	BESKRYWING	MAKS PUNTE	LEERDER PUNTE
VRAAG 2.1			
2.1.1	<u>Attribute declaration under private clause:</u> <pre>fAge : integer; fSeverity : string; fSober : char; fPremium : real; fInitialDamage: real;</pre> (✓✓✓- all 5 included with datatypes, ✓- less than 5, 0 - none)	3	
2.1.2	<u>Constructor declaration under public clause:</u> Constructor CREATE ✓(iAge:integer;sSeverity:string;cSober:char; rPremium:real); ✓ <u>Constructor definition after implementation:</u> <pre>fAge := iAge; fSeverity := sSeverity; fSober := cSober; fPremium := rPremium;</pre> (✓✓- correct complete definition, ✓- some attributes set to parameters, 0 - none)	3	
2.1.3 & 2.1.4	✓✓ 4 CORRECT Accessor methods (correct implementation, subtract 1 mark per incorrect/missing accessor – maximum TWO incorrect) ✓✓ 1 CORRECT Mutator method (SET <u>procedures</u> – declaration and implementation, subtract 1 mark for incorrect declaration & 1 mark for incorrect implementation)	4	
2.1.5	<pre>function TClaim.CalcClaim:real; var Liability : real; begin //Q2.1.5 if (fAge > 24) AND (fSeverity = 'minor') then Liability := fPremium * 0.5; else Liability := fPremium * 2; if fSober = 'N' then Liability := Liability * 1.25; if Liability >= fInitialDamage then Result := fInitialDamage else Result := fInitialDamage - Liability; end;</pre> ✓ both conditions correct ✓ all parameters correct	8	
2.1.6	<pre>function TClaim.CalcCurrentVal(rInitial:real;rDeprec:integer;iModel:integer) :real; begin //Q2.1.6 Result := rInitial - ((YearOf(Date)-iModel) * rDeprec); if Result < 0 then Result := 0.0; end;</pre> ✓ all parameters correct	4	
2.1.7	Correct Result for ToString: ✓✓ 2 marks = Age, Severity, Sober, Premium, Initial damage concatenated successfully. Subtract 1 mark if more than 1 attribute is missing or concatenated incorrectly. ✓ 1 mark for correct use of at least Format / FloatToStrF with currency.	3	

VRAAG 2.2			
2.2.1	✎ Add Name to redDisplay successfully✓ ✎ Correct info obtained from form(Age,Severity,Sober,Premium) ✓ ✎ Object Claim instantiated correctly (parameter datatypes must correspond with TClaim class unit) Claim:=TClaim.CREATE✓(sedAge.Value,lbxSeverity.Items[lbxSeverity.ItemIndex],cSober,StrToFloat(edtPremium.Text)); ✓ ✎ Mutator method called correctly with monthly premium parameter✓ ✎ ToString method called correctly✓ ✎ Claim method called successfully✓ & displayed in Rich Edit✓ ✎ btnValue.Enabled := True; ✓	9	
2.2.2	✎ Assignfile correctly✓ with file variable & location string ✎ Test file existence correctly by resetting pointer✓, Showmessage if not✓, Exit✓ ✎ Correct use of UPPERCASE for car make✓ ✎ Conditional loop for Eof✓ and ifFOUND technique (i.e. Boolean) ✓ ✎ Readln correctly✓ ✎ Using POS✓, COPY successfully ✓ (OR DELETE) ✎ Annual depreciation extracted correctly after make is found. ✓ ✎ Correct CurrentValue method used from object Claim✓ ✎ Message in Rich edit for CurrentValue✓ as well as if make is not found✓ ✎ CloseFile used correctly <u>after</u> conditional loop✓	15	
2.2.3	✎ Rewrite used correctly✓ ✎ File name correctly concatenated with .txt in assignfile✓ ✎ New premium calculated correctly and store into real variable✓(5% of Claim PLUS old monthly premium) ✎ Writeln✓ Damage✓, (Old & New)✓ premium successfully ✎ Message displaying Claim & New premium correctly✓	7	
TOTAAL:			<u>58</u>

VRAAG 3: MERKGIDS – PROBLEEMOPLOSSINGSPROGRAMERING

VRAAG	BESKRYWING	MAKS PUNTE	LEERDER PUNTE				
VRAAG 3.1							
3.1	[FormActive] ✎ NewHealth image (TImage) declared ✓ & created ✓ (Owner: frmQ3) correctly. ✎ Parent (frmQ3),Width(193),Height(113),Top(8) & Left properties ALL assigned correctly ✓✓ (-1 per incorrect value, max -2) ✎ <ImageName>.Picture.Loadfromfile := 'new_health.jpg'. ✓	5					
VRAAG 3.2							
3.2.1	String grid OR Memo used and visible on pnlOutput ✓	1					
3.2.2	[function Scale] ✎ arrCompany used (correctly/incorrectly as long as it was used) ✓ ✎ Correct arrCompany[?] used with the use of <u>Number</u> parameter as array index value. ✓ ✎ Age & Child fractions extracted correctly from arrCompany ✓ ✎ Condition created for extracting gender point correctly ✓ ✎ Result (calculation) correct ✓✓ e.g. Result := (fracAge * Age)+(pointsGen)- (fracChil*Children); [btnPremiumsClick] ✎ Nested ✓ FOR-loop used in order to extract surname, age, gender & salary from arrClients. ✎ Extract surname, age, gender & salary from arrClients. ✓✓ (-1 per incorrect extraction, max -2) ✎ Correct use of DELETE and/or COPY ✓ and POS and/or LENGTH function. ✎ Function sale called with correct datatype parameters ✓ correctly extracted information from arrClients ✓ ✎ Premium calculated correctly by dividing scale result by 100 and multiplying it with extracted salary ✓ ✎ strgDisplay.Cell[] used to create some form of output ✓ ✎ Client surnames displayed correctly in column 0 ✓ ✎ All premiums displayed correctly ✓ with suitable currency (format / FloatToStrF) function ✓ ✎ Declared arrPremiums [1..4] globally (Unit or Class scope) ✓ ✎ Stored surname and premiums in arrPremiums successfully ✓	19					
3.2.3	[btnAverageClick] ✎ Extracting 3 real values from arrPremiums ✓ using a loop ✎ Initialise some sort of 'TOTAAL' variable (or 3 variables) ✓ ✎ Calculating average correctly TOTAAL / 10 ✓ ✎ Amend (adding) to Memo or String grid component with simple heading (e.g. Average) ✓ ✎ Correct average calculated and displayed in currency: <table border="1" data-bbox="406 1753 917 1795"> <tr> <td>AVERAGE</td> <td>R 1 200.53</td> <td>R 1 546.83</td> <td>R 1 223.93</td> </tr> </table>	AVERAGE	R 1 200.53	R 1 546.83	R 1 223.93	5	
AVERAGE	R 1 200.53	R 1 546.83	R 1 223.93				

VRAAG 3.3			
3.3.1	✎ Use of arrPremiums✓ to extract surname within an unconditional loop 1 to 10✓ ✎ Correct condition statement✓ used for determining lowest premium between three✓ premiums per client inside the loop✓ ✎ Correct medical aid determined for lowest✓ ✎ Surname and medical aid displayed in memo✓ ✎ Random(9)+1✓ used ✎ arrPremiums[random] client chosen correctly and displayed in memo with suitable message✓ ✎ Number of clients per medical aid calculated correctly✓ (variables used globally✓) for later use in Q3.3.2	11	
3.3.2	✎ Correct medical aid name✓ and client amount✓ displayed under pnlTOTAALs.Caption (no marks if pnlTOTAALs was not used)	2	
TOTAAL:			<u>43</u>

Voorgestelde Programeringskode oplossings:

VRAAG 1: MOONTLIKE OPLOSSING – ALGEMENE PROGRAMERINGSVAARDIGHEDE

//VRAAG – [Name and Surname here]

unit VRAAG1_U;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ComCtrls, ExtCtrls, StdCtrls, Buttons, CheckLst, jpeg, Spin, {given} Math;

type

TfrmQ1 = class(TForm)
 PageControl1: TPageControl;
 TabSheet1: TTabSheet;
 TabSheet2: TTabSheet;
 Panel1: TPanel;
 edtCustomer: TEdit;
 lblCustomer: TLabel;
 lblBudget: TLabel;
 edtBudget: TEdit;
 cbxCategory: TComboBox;
 lblCategory: TLabel;
 edtBESKRYWING: TEdit;
 lblBESKRYWING: TLabel;
 rgpPayment: TRadioGroup;
 redDisplay: TRichEdit;
 btnGenerate: TButton;
 bmbRetry: TBitBtn;
 lblSummary: TLabel;
 Panel2: TPanel;
 imgBoxes: TImage;
 clbSizes: TCheckListBox;
 lblBoxSizes: TLabel;
 sedAmount: TSpinEdit;
 lblNumber: TLabel;
 lblFragile: TLabel;
 cbxFragile: TCheckBox;
 btnValidate: TButton;
 pnlValidate: TPanel;
 btnImport: TButton;
 lbxCategory: TComboBox;
 lblCategory2: TLabel;
 btnParcelCost: TButton;


```

    memParcel: TMemo;
    lblCustNum: TLabel;
    procedure btnGenerateClick(Sender: TObject);
    procedure btnValidateClick(Sender: TObject);
    procedure btnImportClick(Sender: TObject);
    procedure btnParcelCostClick(Sender: TObject);
    procedure bmbRetryClick(Sender: TObject);
    procedure btnDisplaySaveClick(Sender: TObject);
private
    TOTAALCost : real; {given}
    sCNumber : string;
    procedure DetermineCost(pCategory,pSizes:integer;pFragile:Boolean;var pTOTAALCost:real);
    { Private declarations }
public
    { Public declarations }
end;

var
    frmQ1: TfrmQ1;

implementation

{$R *.dfm}

procedure TfrmQ1.btnGenerateClick(Sender: TObject);
var
    rBudget : real;
begin
    //Q1.1.1
    redDisplay.Clear;
    redDisplay.Lines.Add('Client Summary');
    redDisplay.Paragraph.Alignment := taLeftJustify;
    redDisplay.Lines.Add("");
    sCNumber := edtCustomer.Text;
    redDisplay.Lines.Add('Customer number: '+#9+edtCustomer.text);
    rBudget := StrToFloat(edtBudget.Text);
    redDisplay.Lines.Add('Client budget: '+#9+FloatToStrF(rBudget,ffCurrency,8,2));
    //OR: redDisplay.Lines.Add('Client budget: '+#9+Format('%8.2m',[rBudget]));
    redDisplay.Lines.Add('Purchase item: '+#9+ cbxCategory.Items[cbxCategory.ItemIndex]+' - '+'
edtBESKRYWING.text);
    redDisplay.Lines.Add('Payment method: '+#9+rgpPayment.Items[rgpPayment.ItemIndex]);
    redDisplay.Lines.Add("");
    redDisplay.SelAttributes.Color :=clRed ;
    case rgpPayment.ItemIndex of
        0 : redDisplay.Lines.Add('Please contact the courier personally for payment arrangements');
        1 : redDisplay.Lines.Add('Please obtain banking details from our website');
        2 : redDisplay.Lines.Add('Please be warned of a 15% interest on goods bought on credit');
    end;
{ OR

```

```
if rgpPayment.ItemIndex = 0 then
    redDisplay.Lines.Add('Please contact the courier personally for payment arrangements')
else if rgpPayment.ItemIndex = 2 then
    redDisplay.Lines.Add('Please be warned of a 15% interest on goods bought on credit');
```

OR

```
if rgpPayment.ItemIndex = 0 then
    redDisplay.Lines.Add('Please contact the courier personally for payment arrangements');
if rgpPayment.ItemIndex = 2 then
    redDisplay.Lines.Add('Please be warned of a 15% interest on goods bought on credit'); }
```

end;

```
procedure TfrmQ1.btnValidateClick(Sender: TObject);
begin
//Q1.1.2
    sCNumber := edtCustomer.Text;
    if (Length(sCNumber) = 8) AND (sCNumber[1] = '#') then
        begin
            pnlValidate.Color := clLime;
            pnlValidate.Caption := 'Valid!';
        end
    else
        begin
            if (Length(sCNumber) <> 8) AND (sCNumber[1] <> '#') then
                begin
                    pnlValidate.Color := clRed;
                    pnlValidate.Caption := 'Invalid!';
                end
            else
                begin
                    if (Length(sCNumber) <> 8) then
                        begin
                            pnlValidate.Color := clRed;
                            pnlValidate.Caption := 'Length!';
                        end
                    else if sCNumber[1] <> '#' then
                        begin
                            pnlValidate.Color := clRed;
                            pnlValidate.Caption := 'Start: #';
                        end;
                end;
        end;
end;
```

end;

```
procedure TfrmQ1.bmbRetryClick(Sender: TObject);
begin
```

//Q1.1.3

```
edtCustomer.Clear; //OR edtCustomer.Text := ''; as well as for all edits alt.
edtBudget.Clear;
edtBESKRYWING.Clear;
cbxCategory.ItemIndex := -1;
rgpPayment.ItemIndex := -1;
pnlValidate.Color := clSilver;
pnlValidate.Caption := '';
redDisplay.Clear;
edtCustomer.Setfocus;
end;
```

```
procedure TfrmQ1.btnImportClick(Sender: TObject);
```

```
begin
```

```
  //Q1.2.1
```

```
  lblCustNum.Caption := lblCustNum.Caption + ' ' + sCNumber;
```

```
  //OR lblCustNum.Caption := 'Customer number: ' + sCNumber;
```

```
end;
```

```
procedure TfrmQ1.DetermineCost(pCategory,pSizes:integer;pFragile:Boolean;var pTOTAALCost:real);
```

```
var
```

```
  rWeight : real;
```

```
  iBoxes : integer;
```

```
begin
```

```
  //Q1.2.2 - User-defined Parcel TOTAAL cost steps
```

```
  case pCategory of
```

```
    0 : rWeight := sedAmount.Value*1.3;
```

```
    1 : rWeight := sedAmount.Value*10.7;
```

```
    2 : rWeight := sedAmount.Value*2.5;
```

```
    3 : rWeight := sedAmount.Value*5.8;
```

```
    4 : rWeight := sedAmount.Value*0.3;
```

```
  end;
```

```
  case pSizes of
```

```
    0 : begin
```

```
      iBoxes:= Ceil(rWeight/5);
```

```
      pTOTAALCost := iBoxes*55.95;
```

```
    end;
```

```
    1 : begin
```

```
      iBoxes:= Ceil(rWeight/3.5);
```

```
      pTOTAALCost := iBoxes*39.35;
```

```
    end;
```

```
    2 : begin
```

```
      iBoxes:= Ceil(rWeight/6.5);
```

```
      pTOTAALCost := iBoxes*71.45;
```

```
    end;
```

```
  end;
```

```
  if cbxFragile.Checked then
```

```
    pTOTAALCost := pTOTAALCost*1.05;
```

```
  //Refresh components
```

```

    clbSizes.Checked[clbSizes.ItemIndex] := False;
    sedAmount.Value := 0;
    cbxFragile.Checked := False;
    lbxCategory.ItemIndex := -1;
    clbSizes.SetFocus;
end;

procedure TfrmQ1.btnParcelCostClick(Sender: TObject);
begin
    //Q1.2.2 - Parcel Cost
    DetermineCost(lbxCategory.ItemIndex,clbSizes.ItemIndex,cbxFragile.Checked,TOTAALCost);
    ShowMessage('Parcel will cost '+Format('%8.2m',[TOTAALCost])+ ' for customer '
        + sCNumber +#13+'Information stored successfully');
end;

procedure TfrmQ1.btnDisplaySaveClick(Sender: TObject);
var
    k : integer;
begin
    //Q1.2.3
    memParcel.Lines.Add('Parcel Cost: '+Format('%8.2m',[TOTAALCost])+
        '. Customer: '+sCNumber);
    memParcel.Lines.SaveToFile('Parcels.rtf');
end;

end.

```

VRAAG 2: MOONTLIKE OPLOSSING – OBJEKGEORIENTEERDE PROGRAMERING

//VRAAG 2 Class - [Name and Surname here]

unit Claim_U;

interface

uses SysUtils, DateUtils; {given}

type

TClaim = class(TObject) {given}

private

//Q2.1.1

fAge : integer;

fSeverity : string;

fSober : char;

fPremium : real;

fInitialDamage: real;

public

Constructor CREATE(iAge:integer;sSeverity:string;cSober:char; //Q2.1.2

```

                                rPremium:real);
function GETAge:integer;    {given}
function GETSeverity:string; {given}
function GETSober:char;    {given}
function GETPremium : real; {given}
procedure SETDamage(rInitialDamage:real); //Q2.1.4
function CalcClaim:real;    {given}
function CalcCurrentVal(rInitial:real;rDeprec:integer;iModel:integer)
                                :real;

function ToString:string;

end;

implementation

Constructor TClaim.CREATE(iAge:integer;sSeverity:string;cSober:char;
                                rPremium:real);
begin
    //Q2.1.2
    fAge      := iAge;
    fSeverity  := sSeverity;
    fSober     := cSober;
    fPremium   := rPremium;
end;

function TClaim.GETAge:integer; //Q2.1.3
begin
    Result := fAge;
end;

function TClaim.GETSeverity:string; //Q2.1.3
begin
    Result := fSeverity;
end;

function TClaim.GETSober:char; //Q2.1.3
begin
    Result := fSober;
end;

function TClaim.GETPremium : real; //Q2.1.3
begin
    Result := fPremium;
end;

procedure TClaim.SETDamage(rInitialDamage:real); //Q2.1.4
begin
    fInitialDamage := rInitialDamage;
end;
```

```

function TClaim.CalcClaim:real;           {skeleton implementation given}
var
  Liability : real;
begin
  //Q2.1.5
  if (fAge > 24) AND (fSeverity = 'minor') then
    Liability := fPremium * 0.5
  else Liability := fPremium * 2;
  if fSober = 'N' then Liability := Liability * 1.25;

  if Liability >= fInitialDamage then
    Result := fInitialDamage
  else Result := fInitialDamage - Liability;
end;

function TClaim.CalcCurrentVal(rInitial:real;rDeprec:integer;iModel:integer)
                                     :real;
begin
  //Q2.1.6
  Result := rInitial - ((YearOf(Date)-iModel) * rDeprec);
  if Result < 0 then
    Result := 0.0;
end;

function TClaim.ToString:string;      {skeleton implimentation given}
begin
  //Q2.1.7
  Result := 'Age: '+ IntToStr(fAge)+#13
    +'Severity: ' + fSeverity + #13
    +'Sober: ' + fSober + #13
    +'Premium: ' + Format('%10.2m',[fPremium]) + #13
    +'Initial damage: ' + Format('%10.2m',[fInitialDamage]);
end;

end.

```

//VRAAG 2 Main - [Name and Surname here]

```
unit VRAAG2_U;
```

```
interface
```

```
uses
```

```
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, ExtCtrls, ComCtrls, Claim_U;
```

```
type
```

```
  TfrmQ2 = class(TForm)
```

```
pnlMain: TPanel;  
pnlSub: TPanel;  
lblDetails: TLabel;  
edtName: TEdit;  
lblName: TLabel;  
lblAge: TLabel;  
sedAge: TSpinEdit;  
lblPremium: TLabel;  
edtPremium: TEdit;  
lbxSeverity: TListBox;  
lblSeverity: TLabel;  
lblInitialCost: TLabel;  
edtInitialDamage: TEdit;  
cbxSober: TCheckBox;  
pnlValueCalc: TPanel;  
lblCarMake: TLabel;  
edtCarMake: TEdit;  
lblCarModel: TLabel;  
edtCarModel: TEdit;  
btnValue: TButton;  
lblCarInitialCost: TLabel;  
edtInitialCost: TEdit;  
redDisplay: TRichEdit;  
btnProcess: TButton;  
btnSave: TButton;  
procedure btnValueClick(Sender: TObject);  
procedure FormActivate(Sender: TObject);  
procedure btnProcessClick(Sender: TObject);  
procedure FormClose(Sender: TObject; var Action: TCloseAction);  
procedure btnSaveClick(Sender: TObject);  
private  
    Claim : TClaim; {given}  
    sClaim : string;  
    { Private declarations }  
public  
    { Public declarations }  
end;
```

```
var  
    frmQ2: TfrmQ2;
```

```
implementation
```

```
{ $R *.dfm }
```

```
procedure TfrmQ2.FormActivate(Sender: TObject);  
begin  
    btnValue.Enabled := False; {given}
```

end;

```
procedure TfrmQ2.btnProcessClick(Sender: TObject);
var
  cSober : char;
begin
  //Q2.2.1
  redDisplay.Clear;
  redDisplay.Lines.Add('Name: '+edtName.Text);
  if cbxSober.Checked then cSober := 'Y'
  else cSober := 'N';
  Claim := TClaim.CREATE(sedAge.Value,lbxSeverity.Items[lbxSeverity.ItemIndex],
                        cSober,StrToFloat(edtPremium.Text));
  Claim.SetDamage(StrToFloat(edtInitialDamage.Text));
  redDisplay.Lines.Add(Claim.ToString);
  sClaim := FloatToStrF(Claim.CalcClaim,ffCurrency,8,2);
  redDisplay.Lines.Add('Claim: '+ sClaim);
  btnValue.Enabled := True;
end;
```

```
procedure TfrmQ2.btnValueClick(Sender: TObject);
var
  tFile : textFile;
  sLine, sMake : string;
  bFound : boolean;
  iDeprec : integer;
  rCurrent : real;
begin
  //Q2.2.2
  AssignFile(tFile,'ValueCars.txt');
  Try
    reset(tFile);
  Except
    ShowMessage('ValueCars.txt does not exist');
    Exit;
  End;
  sMake := UPPERCASE(edtCarMake.Text);
  bFound := False;
  while NOT (EoF(tFile) OR (bFound)) do
    begin
      readln(tFile,sLine);
      if (sMake = COPY(sLine,1,POS('#',sLine)-1)) then
        begin
          bFound := True;
          iDeprec := StrToInt(Copy(sLine,POS('#',sLine)+1,(Length(sLine)-POS('#',sLine))));
          rCurrent :=
            Claim.CalcCurrentVal(StrToFloat(edtInitialCost.Text),iDeprec,StrToInt(edtCarModel.Text));
        end;
    end;
```



```

    end;//While
    CloseFile(tFile);
    if bFound = False then
        redDisplay.Lines.Add('Current Value: no value for '+sMake)
    else
        redDisplay.Lines.Add('Current Value: '+Format('%10.2m',[rCurrent]));
    end;
end;

```

```

procedure TfrmQ2.btnSaveClick(Sender: TObject);
var
    tFile : textfile;
    tName : string;
    rNew : real;
begin
    //Q2.2.3
    tName := edtName.Text + ' - ' + sClaim + '.txt';
    rNew := StrToFloat(edtPremium.Text) + (Claim.CalcClaim * 0.05);
    AssignFile(tFile,tName);
    Rewrite(tFile);

    Writeln(tFile,'Damage cost: '+Format('%10.2m',[StrToFloat(edtInitialDamage.Text)]));
    Writeln(tFile,'Old premium: '+Format('%10.2m',[StrToFloat(edtPremium.Text)]));
    Writeln(tFile,'New premium: '+Format('%10.2m',[rNew]));
    ShowMessage(tName+' has been saved!'+#13+'New premium: '+Format('%10.2m',[rNew]));
    CloseFile(tFile);
end;

procedure TfrmQ2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Claim.Free;
end;

end.

```

VRAAG 3: MOONTLIKE OPLOSSING – PROBLEEMOPLOSSINGSPROGRAMERING (1D array approach)

//VRAAG 3 - [Name and Surname here]

```
unit VRAAG3_U;
```

```
interface
```

```
uses
```

```
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, Grids, jpeg, ExtCtrls, StdCtrls, ComCtrls, Math;
```

```
type
```

```

TfrmQ3 = class(TForm)
  pnlCalc: TPanel;
  imgUSave: TImage;
  imgMyCover: TImage;
  btnPremiums: TButton;
  btnBest: TButton;
  pnlTOTAALS: TPanel;
  memDisplay: TMemo;
  btnAverage: TButton;
  btnTOTAALS: TButton;
  pnlOutput: TPanel;
  strgDisplay: TStringGrid;
  procedure btnPremiumsClick(Sender: TObject);
  procedure FormActivate(Sender: TObject);
  procedure btnAverageClick(Sender: TObject);
  procedure btnBestClick(Sender: TObject);
  procedure btnTOTAALSClick(Sender: TObject);
private
  imgNewHealth : TImage;
  USave,MyCover,NewHealth : integer;
  arrPremiums : array[1..10] of string;
  function Scale(Age,Children:integer;Gender:char;Number:integer):real; {given}

  { Private declarations }
public
  { Public declarations }
end;

var
  frmQ3: TfrmQ3;

const
  arrCompany : array[1..3] of string = ('0.21M5.1F4.3C1.5','0.29M7.4F3.9C2.3',
    '0.26M4.2F8.1C2.6');
  arrClients : array[1..10] of string = ('Anderson$36$M3$8750',
    'Kekana$21$F0$10800','Gouws$68$M4$3400','Nhleko$47$F42$6700',
    'Campbell$19$M0$22750','Thusi$18$F0$13400','Vermaak$49$M5$26000',
    'Clapton$72$M2$17900','Tankian$43$M1$4800','Mahlangu$26$F3$8700');

implementation

{$R *.dfm}

procedure TfrmQ3.FormActivate(Sender: TObject);
//Q3.1
begin
  imgNewHealth := TImage.Create(frmQ3);
  with imgNewHealth do
    begin

```

```
    Parent := frmQ3;
    Left := 400;
    Top := 8;
    Width := 193;
    Height := 113;
    Picture.LoadFromFile('new_health.jpg');
end;
end;

function TfrmQ3.Scale(Age:integer;Gender:char;Number:integer):real;
//Q3.2.2 - function
var
    sCompany : string;
    fracAge, fracChil, pointsGen, percentage : real;
begin
    sCompany := arrCompany[Number];
    fracAge := StrToFloat(COPY(sCompany,1,4));
    fracChil := StrToFloat(COPY(sCompany,POS('C',sCompany)+1,3));
    if Gender = 'M' then
        pointsGen := StrToFloat(COPY(sCompany,POS('M',sCompany)+1,3))
    else pointsGen := StrToFloat(COPY(sCompany,POS('F',sCompany)+1,3));
    Result := (fracAge * Age)+(pointsGen)-(fracChil*Children);
end;

procedure TfrmQ3.btnPremiumsClick(Sender: TObject);
//Q3.2.2 - Premiums OnClick
var
    c,r : integer;
    sClient,sName : string;
    iA,iC : integer;
    cG : char;
    rSal, rPrem : real;
begin
    strgDisplay.Cells[1,0] := 'U Save';
    strgDisplay.Cells[2,0] := 'My Cover';
    strgDisplay.Cells[3,0] := 'New Health';
    for r := 1 to 10 do
        for c := 0 to 3 do
            begin
                //Extract & Display client Name, Age, Gender, Children & Salary
                sClient := arrClients[r];
                sName := COPY(sClient,1,POS('$',sClient)-1);
                DELETE(sClient,1,POS('$',sClient));
                if c > 0 then//else and if condition can swop as well for surname first
                    begin
                        iA := StrToInt(COPY(sClient,1,POS('$',sClient)-1));
                        DELETE(sClient,1,POS('$',sClient));
                        cG := sClient[1];
                        iC := StrToInt(sClient[2]);
```

```

        DELETE(sClient,1,POS('$',sClient));
        rSal := StrToFloat(sClient);
        //Use Scale in order to determine and display Premium
        rPrem := Scale(iA,iC,cG,c) / 100 * rSal;
        strgDisplay.Cells[c,r] := format('%10.2m',[rPrem]);
        arrPremiums[r] := arrPremiums[r] + FloatToStrF(rPrem,ffFixed,8,2)
                                + '#';
    end
else
    begin
        strgDisplay.Cells[c,r] := sName;
        arrPremiums[r] := sName+'#';
    end;
end; //for c
end;

```

```

procedure TfrmQ3.btnAverageClick(Sender: TObject);
//Q3.2.3
var
    i,k: integer;
    Tot1,Tot2,Tot3 : real;
    sPrem : string;
begin
    Tot1 := 0.0;
    Tot2 := 0.0;
    Tot3 := 0.0;
    for i := 1 to 10 do
        begin
            sPrem := arrPremiums[i];
            DELETE(sPrem,1,POS('#',sPrem));
            Tot1 := Tot1 + StrToFloat(COPY(sPrem,1,POS('#',sPrem)-1));
            DELETE(sPrem,1,POS('#',sPrem));
            Tot2 := Tot2 + StrToFloat(COPY(sPrem,1,POS('#',sPrem)-1));
            DELETE(sPrem,1,POS('#',sPrem));
            Tot3 := Tot3 + StrToFloat(COPY(sPrem,1,POS('#',sPrem)-1));
        end;
    strgDisplay.Cells[0,11] := 'AVERAGE';
    strgDisplay.Cells[1,11] := format('%10.2m',[Tot1/10]);
    strgDisplay.Cells[2,11] := format('%10.2m',[Tot2/10]);
    strgDisplay.Cells[3,11] := format('%10.2m',[Tot3/10]);
end;

```

```

procedure TfrmQ3.btnBestClick(Sender: TObject);
//Q3.3.1
var
    k,i,R : integer;
    sPremium,sLine : string;
    p1,p2,p3: real;
begin

```

```

uSave := 0;
MyCover := 0;
NewHealth := 0;
memDisplay.Clear;
memDisplay.Lines.Add('Best medical scheme:'+#13);
for k := 1 to 10 do
begin
  sPremium := arrPremiums[k];
  sLine := COPY (sPremium,1,POS('#',sPremium)-1)+#9;
  DELETE(sPremium,1,POS('#',sPremium));
  p1 := StrToFloat(COPY(sPremium,1,POS('#',sPremium)-1));
  DELETE(sPremium,1,POS('#',sPremium));
  p2 := StrToFloat(COPY(sPremium,1,POS('#',sPremium)-1));
  DELETE(sPremium,1,POS('#',sPremium));
  p3 := StrToFloat(COPY(sPremium,1,POS('#',sPremium)-1));

  if (p1 < p2) AND (p1 < p3) then
    begin
      sLine := sLine +#9+'U Save';
      inc(USave);
    end
  else if (p2 < p1) AND (p2 < p3) then
    begin
      sLine := sLine +#9+'My Cover';
      inc(MyCover);
    end
  else
    begin
      sLine := sLine +#9+'New health';
      inc(NewHealth);
    end;
  memDisplay.Lines.Add(sLine);
  R := random(9)+1;
end; //for k
memDisplay.Lines.Add(Copy(arrPremiums[R],1,POS('#',arrPremiums[R])-1)+
  ' wins 1 month zero premium!');
end;

procedure TfrmQ3.btnTOTAALsClick(Sender: TObject);
//Q3.3.2
begin
  //TOTAALs calculated in q3.3.1 & displayed here:
  pnlTOTAALs.Caption := 'U Save: '+IntToStr(USave)+
    ' | My Cover: '+ IntToStr(MyCover)+
    ' | New Health: '+ IntToStr(NewHealth);
end;
end.

```