

**GAUTENG DEPARTMENT OF EDUCATION
PREPARATORY EXAMINATION**

**INFORMATION TECHNOLOGY
(First Paper: Practical)**

POSSIBLE ANSWERS

GENERAL INFORMATION

- Addenda A to D contains a cover sheet as well as a **marking grid** for each question. Copies should be made for each candidate to be completed during the marking session.
- **ANY WORKABLE solution** for the questions has to be carefully considered. There may be alternative solutions to those suggested in the memo provided.
- **Syntax errors** must only be penalized when it leads to a logical error in the program up to a maximum of 2 marks per question. Other syntax errors must be ignored.
- For Question 2 and 3 the half marks must be carried to the cover sheet. **ONLY the final total must be rounded off** on the cover sheet.

QUESTION 1: DATABASE AND PROGRAMMING

```

unit Q1_MEMO_U;
//PROPOSED MEMORANDUM
interface
    // definition declaration of form and procedures
implementation

{$R *.dfm}

procedure TQUESTION_1.Button1Click(Sender: TObject);
begin
//question 1.1
    adoQ.Close;
    adoQ.SQL.Text := 'SELECT Doctor, Tariff ' + ✓
                    'FROM Doctors ' ✓ +
                    'ORDER BY ✓Doctor DESC'✓;

    adoQ.Open;
end;
//===== (4)
procedure TQUESTION_1.Button2Click(Sender: TObject);
var
    YY : String;
begin
//question 1.2
    YY := InputBox('Births', 'Enter a year.', '2008');✓
    adoQ.Close;
    adoQ.SQL.Text := 'SELECT Surname, FirstName, DOB, Weight ' + }✓
                    'FROM Babies ' +
                    'WHERE (Year(DOB)✓ = ' + YY✓ + ') AND✓ (Weight < 3.5)✓';

    adoQ.Open;
end;
//===== (6)
procedure TQUESTION_1.Button3Click(Sender: TObject);
begin
//question 1.3
    adoQ.Close;
    adoQ.SQL.Text := 'SELECT Count(*)✓ AS [Boys born in Jan 2008]✓ ' +
                    'FROM Babies ' +
                    'WHERE ((Year(DOB)=2008)AND(Month(DOB)=1)) '✓ +
                    'AND (Gender="M")✓';

    adoQ.Open;
end;
//===== (4)

```

Please Note:

To determine the date (Jan 2008) several alternative solutions may be used:

- (DOB LIKE "2008/01/%")
- DOB in (DateValue("2008/01/01"), DateValue("2008/01/31"))

TEST EVERY ALTERNATIVE!

```

procedure TQUESTION_1.Button4Click(Sender: TObject);
begin
//question 1.4
adoQ.Close;
adoQ.SQL.Text := 'UPDATE Babies ✓' +
                 'SET✓ DOB = #2007/06/16# '✓ +
                 'WHERE (Surname="Adam") AND (FirstName="Stanley")✓';

adoQ.ExecSQL;;
adoQ.Close;
adoQ.SQL.Text := 'SELECT * FROM Babies ' +
                 'WHERE (Surname="Adam") AND (FirstName="Stanley")';✓{display}

adoQ.Open;
end;
(5)
//=====
procedure TQUESTION_1.Button5Click(Sender: TObject);
begin
//question 1.5
adoQ.Close;
adoQ.SQL.Text := 'SELECT FirstName, DOB, Doctor, '✓+
                 'Format✓(DaysInHosp*Tariff✓,"Currency") AS [Cost per Child]✓'+
                 'FROM Babies, Doctors ✓' +
                 'WHERE (Surname = "Green")✓ AND (DoctorID=PracNum)✓';
                 {Linking the two tables can also be done with a JOIN}
                 {FROM Doctors INNER JOIN Babies ON Doctors.PracNum = Babies.DoctorID}

adoQ.Open;
end;
(7)
//=====
procedure TQUESTION_1.Button6Click(Sender: TObject);
begin
//question 1.6
adoQ.Close;
adoQ.SQL.Text := 'SELECT TOP 5 Babies.Surname, FirstName, ' +
                 'DOB AS [Date of birth], Suburb ' + ✓
                 'FROM Parents, Babies ' +
                 'WHERE (Parents.AccNom = Babies.AccNom)✓ AND ' +
                 '(Gender = "F")✓ AND ✓ ' +
                 '((Suburb="Sinoville")✓ OR (Suburb="Rooihuiskraal")✓)' +
                 'ORDER BY DOB DESC✓';
                 {Linking the two tables can also be done with a JOIN}
                 {FROM Parents INNER JOIN Babies ON Parents.AccNom = Babies.AccNom}

adoQ.Open;
end;
(7)
//=====
procedure TQUESTION_1.Button7Click(Sender: TObject);
begin
//question 1.7
adoQ.Close;
adoQ.SQL.Text := 'INSERT INTO Doctors ✓(PracNum, Doctor, Tariff)✓ ' +
                 'VALUES ("BEL763", "BELL G", 550.00) ✓';

adoQ.ExecSQL;
adoQ.Close;
adoQ.SQL.Text := 'SELECT * FROM Doctors ORDER BY Doctor '✓; ✓
adoQ.Open;
end;
(4)
//=====
procedure TQUESTION_1.Button8Click(Sender: TObject);
begin
//question 1.8
adoQ.Close;
adoQ.SQL.Text := 'SELECT Surname, FirstName '✓+
                 'FROM Babies ' +
                 'WHERE (FirstName Like "Jo%"✓) AND (Gender = "M")✓ '✓';

adoQ.Open;
end;
(3)
//=====

```

```

procedure TQUESTION_1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
//close the connection to thE database
  adoQ.Close;
  adoCONN.Close;
end;

procedure TQUESTION_1.FormCreate(Sender: TObject);
begin
//Auto create the connection string to the database
  IF FileExists(ExtractFilePath(Application.ExeName) + 'BABIESINFO.mdb')
  then
  begin
      adoCONN.ConnectionString :=
          'Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' +
          'BABIESINFO.mdb;Persist Security Info=False';
      adoCONN.Open;
      stbQ1.SimpleText := 'Database is connected.'
  end
  Else
  begin
      stbQ1.SimpleText := 'Database NOT connected.'
  end;
end;

end.

```

Given code
No marks

QUESTION 2: OOP

CLASS UNIT:

```

unit DAYCARE_MEMO_U;
//proposed class unit memo
interface

TYPE
  TPatient = class(TObject) ✓
  private
    fFirstName : String[30] ✓;
    fPhysio,
    fRadio,
    fSpecialist : Boolean; ✓
    fBalance    : Real; ✓
  public
    Constructor Create(Name : String; Physio, Radio, Spes : Boolean);
    function GetFullName : String;
    function GetBalance    : Real;
  end;

implementation

{ TPatient }

constructor TPatient.Create(Name: String ✓; Physio, Radio, Spes: Boolean✓);
var
  F, R, S : real;
begin
  fFirstName := Name; ✓
  fPhysio    := Physio;
  fRadio     := Radio;
  fSpecialist := Spes; ✓{all 3 boolean values}
  fBalance   := 0;

  IF Physio ✓
  then F := 325.00✓
  else F := 0; ✓

  IF Radio
  then R := 423.00
  else R := 0; ✓{Repeating assignments for Boolean values}

```

```

IF Spes
then S := 876.00
else S := 0;

fBalance := F + R + S;✓✓
end;

function TPatient.GetBalance: Real; ✓
begin
Result := fBalance; ✓
end;

function TPatient.GetFullName: String; ✓
begin
Result := fFirstName; ✓
end;
end.

```

FORM UNIT:

```

unit Q2_MEMO_U;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, ComCtrls, ExtCtrls, StdCtrls, Buttons,
DAYCARE_MEMO_U ✓;

type
TQUESTION2 = class(TForm)
pcVr2: TPageControl;
tbsRegistration: TTabSheet;
tbsDischarge: TTabSheet;
MainMenu1: TMainMenu;
mnuQuit: TMenuItem;
bmbRegister: TBitBtn;
edtName: TEdit;
Label1: TLabel;
btnReport: TButton;
redReport: TRichEdit;
GroupBox1: TGroupBox;
chkPhysio: TCheckBox;
chkRadio: TCheckBox;
chkSpec: TCheckBox;
lblCount: TLabel;
GroupBox2: TGroupBox;
Label3: TLabel;
edtNr: TEdit;
bmbRemove: TBitBtn;
rgpTransfer: TRadioGroup;
procedure btnReportClick(Sender: TObject);
procedure bmbRegisterClick(Sender: TObject);
procedure bmbRemoveClick(Sender: TObject);
procedure mnuQuitClick(Sender: TObject);
procedure edtNrKeyPress(Sender: TObject; var Key: Char);
private
{ Private declarations }
PatientList : Array[1..50]✓ of TPatient✓;
Count      : Integer;✓
iRegistered,
iTransferred,
iDischarge : Integer;
Total      : Real;
public
{ Public declarations }
end;

var
QUESTION2: TQUESTION2;

implementation

```

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

```

procedure TQUESTION2.bmbRegisterClick(Sender: TObject);
begin
  IF (Count > 50) ✓
  then
  begin
    ShowMessage('Maximum of 50 patients per day.')} ✓
    Exit;
  end;

  IF NOT chkPhysio.Checked AND
    NOT chkRadio.Checked AND
    NOT chkSpec.Checked } ✓✓
  then
  begin
    ShowMessage('Must register for at least one treatment!')} ✓
    Exit;
  end;

  IF Length(edtName.Text) = 0 ✓
  then
  begin
    ShowMessage('The name of the patient not filled in.')} ✓
    Exit;
  end;

  Inc(Count, 1); ✓
  Inc(iRegistered,1);
  PatientList[Count]✓ := TPatient.Create✓(edtName.Text, chkPhysio.Checked,
    chkRadio.Checked,chkSpec.Checked);✓✓
end;

procedure TQUESTION2.bmbRemoveClick(Sender: TObject);
var
  iX,a      : Integer;
  sLyn      : String;
  TF        : TextFile;✓
begin
  iX := StrToInt(edtNr.Text);
  IF NOT(ix in [1..Count]) } ✓✓✓
  then
  begin
    showMessage('Invalid Input: 1..' + IntToStr(Count));
    Exit;✓
  end;
  //valid number
  ✓case rgpTransfer.ItemIndex of
    0✓ : begin
      //skryf na teksfile
      AssignFile(TF, 'ICUWARD.txt');✓
      IF FileExists('ICUWARD.txt')✓
      then Append(TF)✓
      else✓ Rewrite(TF)✓;
      sLyn := PatientList[iX].GetFullName + ' is transfered to ICU.';✓
      Writeln✓(TF, PatientList[iX].GetFullName); ✓
      CloseFile(TF);✓
      inc(iTransfered);✓ {Mark allocated at 2.3}
    end; //na waakeenheid
    1✓ : begin
      sLyn := PatientList[iX].GetFullName + ' is discharged.';✓
      inc(iDischarge);✓ {Mark allocated at 2.3}
      Total := Total + PatientList[iX].GetBalance;✓ {Mark allocated at 2.3}
    end; //ontslaan
  end;
  //remove from array
  PatientList[iX].Free;✓
  For a := iX to Count✓ do
    PatientList[a] := PatientList[a+1];✓✓
  Dec(Count, 1);✓

  MessageDlg(sLyn, mtInformation, [mbOk], 0);✓

```

```

    btnReport.Click;✓ //refresh display!
end;

procedure TQUESTION2.btnReportClick(Sender: TObject);
var
    a : integer;
begin
    redReport.Lines.Clear;✓
    redReport.Paragraph.TabCount := 1;
    redReport.Paragraph.Tab[0] := 100;
    redReport.SelAttributes.Style := [fsBold]; } ✓{columns}
    redReport.Lines.Add('Patient' + #9 + 'Balance'); ✓ {headings}
    for a := 1 to Count✓ do
        IF (PatientList[A] is TPatient)
            then
                redReport.Lines.Add(IntToStr(a)✓ + '. ' + PatientList[a].GetFullName✓ + #9 +
                    'R' + Format✓('%8.2F', [PatientList[a].GetBalance]✓))
            else
                redReport.Lines.Add(IntToStr(a) + 'No object');

        lblCount.Caption := 'Count = ' + IntToStr(Count);✓
    end;

procedure TQUESTION2.edtNrKeyPress(Sender: TObject; var Key: Char);
begin
    IF NOT(Key in ['0'..'9', #8]) } ✓✓✓✓ { Alternatively TRY...EXCEPT/Val to test
        then Key := #0; } { for valid numerical data. }
end;
Procedure TQuestion2.Formcreate (Sender: TObject);
begin
    iTransferred:= 0;
    iDischarge:= 0;
    ✓ Total:= 0; {Mark allocated at 2.3}
end;

procedure TQUESTION2.mnuQuitClick(Sender: TObject);
var
    a : integer;
    sLyn : String;
begin
    //display boodskap
    slyn := 'Date: ' + FormatDateTime✓('dd mmm yyyy'✓, Date) + #13 + #13 ✓{empty line} +
        ✓ 'Number of patients = ' + Inttostr(iRegistered) + #13 + #13 +
        'Number of transferred to ICU = ' + IntToStr(iTransferred)✓ + #13 + #13 +
        'Number discharged = ' + Inttostr(iDischarge) + #13 + #13 +
        'Total income = ' + FloattostrF(Total, ffCurrency✓, 8, 2✓);
    MessageDlg(slyn, mtInformation, [mbOk], 0);✓

    //remove all objects
    for a := 1 to 50 do✓
        PatientList[a].Free✓;
    //end program
    Application.Terminate✓;
end;

end.

```

QUESTION 3: DELPHI

```

unit Q3_MEMO_U;

interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, Buttons, ExtCtrls, ComCtrls;

type
    TQUESTION3 = class(TForm)
        Panell: TPanel;
    end;

```

```

BitBtn1: TBitBtn;
Panel2: TPanel;
GroupBox1: TGroupBox;
edtNumber: TEdit;
bmbReport: TBitBtn;
gpbSave: TGroupBox;
bmbSaveAs: TBitBtn;
bmbPrint: TBitBtn;
dlgSaveAs: TSaveDialog;
redReport: TRichEdit;
dlgPrint: TPrintDialog;
procedure FormCreate(Sender: TObject);
procedure bmbReportClick(Sender: TObject);
procedure bmbSaveAsClick(Sender: TObject);
procedure bmbPrintClick(Sender: TObject);
private
  { Private declarations }
  sPers_Info      : String;
  function Valid_Account(sNomX : String) : Boolean; } {given code}
public
  { Public declarations }
end;

var
  QUESTION3: TQUESTION3;

implementation

{$R *.dfm}

procedure TQUESTION3.bmbPrintClick(Sender: TObject);
begin
  //save the report of baby X?
  IF ✓ dlgPrint.Execute ✓
    then redReport.Print('Report'); ✓✓
end;

procedure TQUESTION3.bmbSaveAsClick(Sender: TObject);
begin
  IF✓ dlgSaveAs.Execute ✓
    then redReport.Lines.SaveToFile✓(dlgSaveAs.FileName✓);
end;

procedure TQUESTION3.bmbReportClick(Sender: TObject);
var
  sLyn, sNom,
  sV, sN, sG, sB : String;
  TF              : TextFile;
  Tel, a, b       : Integer;
  arrTime         : Array[1..100] of TDateTime; ✓ {declaration of any array}
  arrCode         : Array[1..100] of Char;
  arrValue        : Array[1..100] of Real;
  dumTime         : TDateTime;
  dumCode         : Char;
  dumValue        : Real;
begin
  //Display baby X
  sNom := Uppercase(edtNumber.Text);
  IF NOT Valid_Account(snom) ✓{function call}
    then
      begin
        ShowMessage('The account number is invalid!'); ✓
        Exit; ✓{alternative with an else..begin...end}
      end;

  //Display personal info
  redReport.Clear; ✓
  redReport.Lines.Add('Account number: ' + sNom);
  Delete(sPers_Info, 1, pos(',', sPers_Info));

```

<ul style="list-style-type: none"> ✓ pos ✓ copy ... p-1 (one less than comma). ✓ ✓ delete ✓repeating for field values

```

sV := copy(sPers_Info, 1, pos(',', sPers_Info)-1);
Delete(sPers_Info, 1, pos(',', sPers_Info));
sN := copy(sPers_Info, 1, pos(',', sPers_Info)-1);
Delete(sPers_Info, 1, pos(',', sPers_Info));
sB := copy(sPers_Info, 1, pos(',', sPers_Info)-1);
Delete(sPers_Info, 1, pos(',', sPers_Info));
sG := sPers_Info[1];
redReport.Lines.Add('BABY: ' + sN + ', ' + sV );
redReport.Lines.Add('Date of Birth: ' + sB); ✓ {display in rich edit}

//display report >> set info into array
AssignFile(TF, 'ICU.txt'); ✓
Reset(TF); ✓
Tel := 0; ✓
While NOT EOF(TF) DO ✓
begin
  Readln(TF, sLyn); ✓
  IF Pos(sNom, sLyn) > 0 ✓
  then
    begin
      Inc(Tel, 1); ✓
      arrTime[Tel] := StrToTime(copy(sLyn, 1, pos(',', sLyn)-1));
      Delete(sLyn, 1, pos(',', sLyn)); //time
      Delete(sLyn, 1, pos(',', sLyn)); //rek nom
      arrCode[Tel] := sLyn[1]; ✓
      Delete(sLyn, 1, 2); //Code
      arrValue[Tel] := StrToFloat(sLyn); ✓ //value
    end;
end;
CloseFile(TF); ✓

//Sort the values
For A := Tel downto 2 do
  For B := 1 to (a-1) do
    IF arrTime[b] > arrTime[b+1]
    then
      begin
        dumTime := arrTime[b];
        arrTime[b] := arrTime[b+1];
        arrTime[b+1] := dumTime;

        dumCode := arrCode[b];
        arrCode[b] := arrCode[b+1];
        arrCode[b+1] := dumCode;

        dumValue := arrValue[b];
        arrValue[b] := arrValue[b+1];
        arrValue[b+1] := dumValue;
      end;
    } Sorting: ✓✓✓✓✓ (5)

//Display info in report
redReport.Paragraph.TabCount := 3;
redReport.Paragraph.Tab[0] := 100;
redReport.Paragraph.Tab[1] := 200;
redReport.Paragraph.Tab[2] := 300;
redReport.Paragraph.Tab[3] := 400; } ✓ {columns}
redReport.Lines.Add('');
redReport.SelAttributes.Style := [fsBold];
redReport.Lines.Add('TIME' + #9 + #9 + 'MEASUREMENTS');
redReport.SelAttributes.Style := [fsBold];
redReport.Lines.Add(' ' + #9 + 'TEMP' + #9 + 'HEART' + #9 + 'OXYGEN'); } ✓✓ {headings}
redReport.Lines.Add(' ' + #9 + '°C' + #9 + 'bpm' + #9 + 'ppm');

For a ✓ := 1 to Tel ✓ do
begin
  sLyn := FormatDateTime('hh:mm', arrTime[a]);
  ✓ case arrCode[a] of
    'T' : sLyn := sLyn + #9 + #9 + FloatToStrF(arrValue[a], ffFixed, 0, 2); ✓
    'H' : sLyn := sLyn + #9 + #9 + #9 + FloatToStrF(arrValue[a], ffFixed, 8, 0);
    'O' : sLyn := sLyn + #9 + #9 + #9 + #9 + FloatToStrF(arrValue[a], ffFixed, 8, 0);
  end;
  redReport.Lines.Add(sLyn); ✓
end;

```

```

end;

procedure TQUESTION3.FormCreate(Sender: TObject);
begin
  dlgSaveAs.InitialDir := ExtractFilePath(Application.ExeName); {given code}
end;

function TQUESTION3.Valid_Account(sNomX: String): Boolean;
var
  sNom, sLyn : String;
  TF          : TextFile; ✓
  Found       : Boolean;
begin
  sNom := Uppercase✓(sNomX);
  IF (length(sNom) <> 4)✓ AND✓
    NOT((sNom[1] in ['A'..'Z']) AND (sNom[2] in ['A'..'Z'])) AND ✓✓
    NOT((sNom[3] in ['0'..'9']) AND (sNom[4] in ['0'..'9'])) ✓✓
  then
    begin
      Result := False;✓
      Exit;✓ {or with a else...begin...end}
    end;
  Found := False;✓
  AssignFile(TF, 'INFO.TXT');✓
  Reset(TF); ✓
  While NOT EOF(TF) ✓AND NOT Found ✓ DO
    begin
      Readln(TF, sLyn); ✓
      sLyn := Uppercase(sLyn);
      IF Copy(sLyn, 1, 4)✓ = sNom ✓
        then
          begin
            Found := True; ✓
            sPers_Info := sLyn; ✓
          end;
        end;
      CloseFile(TF);✓
      IF NOT Found ✓
        then
          Result := False ✓
        else
          Result := True; ✓
    end;
end.

```

ADDENDUM A

CANDIDATE:

QUESTION	DESCRIPTION	MAX MARK	LEARNER MARK	MOD
1	Database and Programming	40		
2	OOP	44		
3	Delphi	36		
	TOTAL	120		

Learner:

I, _____ hereby declare that this exam script is my own, original work.

Learner: _____

Date: _____

Educator: _____

Date: _____

Cluster moderator: _____

Date: _____

ADDENDUM B

QUESTION 1: DELPHI – DATABASE AND PROGRAMMING

Marking rubric

CANDIDATE:

QUESTION	Concepts	Max mark	Learner mark	MOD
1.1.	SELECT Doctor, Tariff (1) FROM Doctors (1) ORDER BY (1) Doctor DESC (1)	4		
1.2.	{SELECT Surname,FirstName, DOB,Weight FROM Babies} (1)WHERE (YEAR(DOB) (1) = ' + yy (1) + ') AND(1) (Weight < 3.5)(1)	6		
1.3.	SELECT Count(*) (1) AS [Boys born in Jan 2008] (1) FROM Babies WHERE (Gender = "M") (1) AND (Year(DOB) = 2008) AND (Month(DOB) = 1) (1) {any correct date method}	4		
1.4.	UPDATE Babies (1) SET (1) DOB = #2007/06/16# (1) WHERE (Surname = "Adam") AND (FirstName = "Stanley") (1) <i>Display of record (1)</i> SELECT * FROM Babies WHERE (Surname="Adam") AND (FirstName="Stanley")	5		
1.5.	SELECT FirstName, DOB, Doctor, (1) Format (1) (DaysInHosp * Tariff (1), "Currency") AS [Cost per Child] (1) FROM Docters, Babies (1 <i>both tables</i>) WHERE (Surname="Green")(1) AND (DocterID =PracNum)(1 <i>linking tables</i>)	7		
1.6.	SELECT TOP 5 Babies.Surname,FirstName , DOB AS [Date of Birth], Suburb (1) FROM Parents, Babies WHERE (Parents.AccNom = Babies.AccNom) (1 <i>linking tables</i>) AND (Gender= "F")(1) AND (1) ((Suburb="Sinoville")(1) OR (1) <i>(The OR statements must be in brackets)</i> (Suburb="Rooihuiskraal")) ORDER BY DOB DESC (1)	7		
1.7.	INSERT INTO Doctors (1) (PracNum, Doctor, Tariff) (1 <i>fieldnames</i>) VALUES ("BEL763", "BELL G", 550.00) (1 <i>values</i>) <i>Display alphabetical list (1)</i> Select * FROM Doctors Order BY Doctor	4		
1.8.	{SELECT Surname, FirstName FROM Babies (1)} WHERE (FirstName Like "Jo%" (1)) AND (Gender="M") (1)	3		
	SYNTAX ERRORS: Penalise only if the syntax error results in a logic error (-1 for each), up to a maximum of 2 marks.			
	SUBTOTAL	40		

ADDENDUM C

QUESTION 2: DELPHI – OOP
Marking rubric

CANDIDATE:

QUESTION	Concepts	Max mark	Learner mark	MOD
2.1.	CLASS UNIT			
	<i>Type</i> definition (1); private fields (1 – string; 1 – real; 1 – Boolean); <i>Constructor</i> : Parameters (2); assign to fields (2); IF physio (1) then (1) else (1); repeating of IF (1); determine balance (2) <i>Accessors</i> : GetBalance (2); GetFullName (2)	18		
2.2.	FORM UNIT			
	Uses unit (1); Declare Array(1) of TPatient(1); counter for array(1)	4		
2.2.1	REGISTRATION			
	Test max 50 patients (2); Test for one treatment(3); Test name(2); Inc(count) (1); create array index object (4){N[a] ✓ := TPatient.Create✓(parameters✓✓)}	12		
2.2.2.	DISCHARGE			
a.	LIST OF NAMES: Clear richedit (1); Columns (1); Headings(1); For-loop to count(1); R.Lines.Add (1); Nr (1); X.GetFullName (1); Format (1 =right align); two decimals (1); X.GetBalance (1); Display count on lbl.Caption (1)	11		
b.	DISCHARGE: VAR Textfile (1); Test valid interval (4); Numerical validation (4) Exit/Else Begin..End(1) Case ItemIndex (1) OPTION ICU: (1)AssignFile(1); Test if file exists (1); Append(1); else (1) Rewrite(1); Message (1); Write info to textfile (1); CloseFile(1); OPTION DISCHARGE: (1)Message(1); REMOVE FROM ARRAY: X[a].Free (1); for loop index to count (1); move element up N[a]✓ := N[a+1✓] (2); Dec(Count) (1); display message(1); Click List of Names-button.Click (1)	29		
2.3.	EXIT OPTION			
	Three non-local variables > Inc(transfer) (1); Inc(discharge) (1); calculate total (2); MESSAGE: Time format (2); empty line(1); subheadings (1); Currency (1); two decimals (1); display message(1) REMOVE ALL OBJECTS: For-loop (1); N[a].Free ✓(1) APPLICATION.TERMINATE (1)	14		
	SYNTAX ERRORS: Penalise only if the syntax error results in a logic error (-1 for each), up to a maximum of 2 marks			
	SUBTOTAL			

ADDENDUM D

QUESTION 3: DELPHI
Marking rubric

CANDIDATE:

QUESTION	Concepts	Max mark	Learner mark	MOD
3.2.	FUNCTION VALID:			
	Var Textfile (1); Uppercase(1); Test conditions: (length of number(1), AND (1), two alphabetical letters(2), two numerical characters(2)); Invalid result = false (1); Exit(1 or with else); flag (1); AssignFile(1); Reset(1); While loop with begin...end (2); Readln(1); Copy (first 4 characters) (1); IF test= (1); flag true(1); sPers_Info(1); CloseFile(1); Not found(1); Result(2)	24		
3.3.	REPORT			
3.3.1.	Test and use of function (3)	3		
3.3.2.	DISPLAY PERSONAL INFO: Clear richedit (1); Pos (1); Copy (1); (copy upto comma: p-1)(1) ; Delete(1); repeat for all the fields(1); Add in richedit (1)	7		
3.3.3.	DISPLAY DATA CHRONOLOGICALLY: A) READ TEXT FILE INTO ARRAY: {11} Declare array (1) AssignFile (1); Reset (1); Counter init (1); While-loop (1); Readln from textfile (1); Test if number is part of line (1); <i>If Number appears in line:</i> Inc teller (1); StrToTime(1); arrCode(1); arrValue(1); CloseFile(1); B) SORT ARRAYS: {5} Sort all 3x arrays(5) C) DISPLAY INFO: {12} Columns (1); Headings (2); For-loop (2); FormatDateTime function (1); Case arrCode[x] (1); slyn := slyn (1); Correct #9 {tab's} in line added(3); Array value (1); Lines.Add (1)	30		
3.4.	SAVE AS...: IF (1); dlg.Execute (1); Get filename (1); SaveToFile(1)	4		
3.5.	PRINT: IF (1);dlg.Execute (1); richedit.Print (2);	4		
	SYNTAX ERRORS: Penalise only if the syntax error results in a logic error (-1 for each), up to a maximum of 2 marks			
	SUBTOTAL			

ALTERNATIVE MEMO: QUESTION 3

Hier is 'n alternatiewe memo vir Vraag 3 wat gebruik maak van classes en objects.

Here is an alternative memorandum for Question 3, making use of classes and objects

```

unit VR3_CLASS_UNIT_MEMO_U;
//*****
//**  ALTERNATIEWE MEMORANDUM VIR VRAAG 3  **
//**  Class                               **
//*****

interface

USES SysUtils;

TYPE

  TWaarneming = class(TObject)
  private
    fTyd      : TDateTime;
    fKode     : Char;
    fMeting  : Real;
  public
    constructor Create;
    procedure SetLesing(Tyd : TDateTime; Kode : Char; Meting : Real);
    function GetTyd : TDateTime;
    function ToString : String;
  end;

  TBaby = class(TObject)
  private
    fNaam      : String;
    fRekNom    : String;
    fGebDat    : String;
    fGender    : String;
    fLesings   : Array[1..100] of TWaarneming;
    fTeller    : Integer;
    procedure SetLesingX(Index : Integer; LX : TWaarneming);
  public
    constructor Create(Line: String);
    procedure SetLesings(sLine : String);
    function GetLesingX(Index : Integer) : TWaarneming;
    function GetNaam : String;
    function GetRekeningNom : String;
    function GetDOB : String;
    function GetTeller : Integer;
    procedure Sorteer;
    procedure StelVry;
  end;

implementation

{ TPasient }

constructor TBaby.Create(Line: String);
var
  a : Integer;
  sX, sY : String;
begin
  fRekNom := Copy(Line, 1, pos(',', Line)-1);
  Delete(Line, 1, pos(',', Line));
  sX := Copy(Line, 1, pos(',', Line)-1);
  Delete(Line, 1, pos(',', Line));
  sY := Copy(Line, 1, pos(',', Line)-1);
  Delete(Line, 1, pos(',', Line));
  fNaam := sY + ' ' + sX;
  fGebDat := Copy(Line, 1, pos(',', Line)-1);
  Delete(Line, 1, pos(',', Line));
  fGender := Line[1];
  fTeller := 0;
  For a := 1 to 100 do
    fLesings[a] := TWaarneming.Create;
  end;
end;

```

```

procedure TBaby.StelVry;
var
  a : integer;
begin
  for a := 1 to 100 do
    self.fLesings[A].Free;
    self.Free;
  end;

function TBaby.GetDOB: String;
begin
  result := fGebDat;
end;

function TBaby.GetLesingX(Index: Integer):
Twaarneming;
begin
  //*****
procedure TBaby.SetLesings(sLine: String);
var
  st, sk, sl : String;
begin
  // cop/pos
  st := copy(sLine, 1, pos(',', sLine)-1);
  Delete(sline,1, pos(',', sline));
  Delete(sline,1, pos(',', sline)); //acc nom
  sk := copy(sLine, 1, pos(',', sLine)-1);
  Delete(sline,1, pos(',', sline)); //code
  sl := sLine;
  Inc(fTeller, 1);
  fLesings[fTeller].SetLesing(StrToTime(st), sk[1], StrToFloat(sL));
end;

procedure TBaby.SetLesingX(Index: Integer; LX: Twaarneming);
begin
  fLesings[Index] := LX;
end;

procedure TBaby.Sorteer;
var
  a, b : integer;
  Dummy : Twaarneming;
begin
  For A := Self.fTeller downto 2 do
    for B := 1 to (a-1) do
      IF (Self.GetLesingX(B).GetTyd > Self.GetLesingX(B+1).GetTyd)
      then
        begin
          Dummy := Self.GetLesingX(B);
          Self.SetLesingX(B, Self.GetLesingX(B+1));
          Self.SetLesingX(B+1, Dummy);
        end;
    end;
  end;

{ Twaarneming }

constructor Twaarneming.Create;
begin
  fTyd := 0;
  fKode := #0;
  fMeting := 0.00;
end;

function Twaarneming.GetTyd: TDateTime;
begin
  Result := fTyd;
end;

procedure Twaarneming.SetLesing(Tyd: TDateTime; Kode: Char; Meting: Real);
begin
  fTyd := Tyd;
  fKode := Kode;
  fMeting := Meting;
end;

function Twaarneming.ToString: String;
var
  sDum : String;
begin

```

```

sDum := FormatDateTime('hh:mm', Self.fTyd);
Case Self.fKode of
  'T' : sDum := sDum + #9 + Format('%0.2f', [Self.fMeting]);
  'H' : sDum := sDum + #9+#9 + Format('%0.0f', [Self.fMeting]);
  'O' : sDum := sDum + #9+#9+#9 + Format('%0.0f', [Self.fMeting]);
End;
Result := sDum;
end;
end.

```

```

unit VR3_CLS_MEMO_2_U;
//*****
/** ALTERNATIEWE MEMORANDUM VIR VRAAG 3 **
/** wat gebruik maak van classes & objects! **
//*****
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, ComCtrls,
  VR3_CLASS_UNIT_MEMO_U;
type
  TVRAAG_3 = class(TForm)
  private
    { Private declarations }
    Baby : TBaby;
    sPers_Info : String;
    function Geldige_Rekening(sNomX : String) : Boolean;
  public
    { Public declarations }
  end;
var
  VRAAG_3: TVRAAG_3;
implementation
{$R *.dfm}
procedure TVRAAG_3.BitBtn1Click(Sender: TObject);
begin
  Application.Terminate;
end;
procedure TVRAAG_3.bmbPrintClick(Sender: TObject);
begin
  //Maak 'n drukstuk van verslag van baba X?
  IF dlgPrint.Execute
  then redX.Print('Verslag');
end;
procedure TVRAAG_3.bmbSaveAsClick(Sender: TObject);
begin
  //Stoor die verslag van baba X?
  if dlgSaveAs.Execute
  then redX.Lines.SaveToFile(dlgSaveAs.FileName);
end;
procedure TVRAAG_3.bmbVerslagClick(Sender: TObject);
var
  sLyn, sX : String;
  TF : TextFile;
  A : Integer;
begin
  //Vertoon die verslag van baba X?
  sx := uppercase(edtNommer.Text);
  if not Geldige_Rekening(sx)
  then
  begin
    ShowMessage('Rekeningnommer is ONGELDIG.');
```

```

AssignFile(TF, 'ICU.txt');
Reset(TF);
While NOT EOF(TF) DO
Begin
  readln(tf, sLyn);
  IF pos(Baby.GetRekeningNom, sLyn) > 0
  then
    begin
      Baby.SetLesings(sLyn);
    end;
  End;
  CloseFile(TF);
//STEP 2: sort CLASS array
Baby.Sorteer;
//Step 3: Display array
redX.Lines.Clear;
redX.Lines.Add('ACCOUNT NUMBER: ' + Baby.GetRekeningNom);
redX.Lines.Add('BABY          : ' + Baby.GetNaam);
redX.Lines.Add('DATE OF BIRTH : ' + Baby.GetDOB);
redX.Lines.Add(' ');
redX.Paragraph.TabCount := 3;
redX.Paragraph.Tab[0] := 100;
redX.Paragraph.Tab[1] := 200;
redX.Paragraph.Tab[2] := 300;
redX.Lines.Add('TIME' + #9 + #9 + 'MEASUREMENTS');
redX.Lines.Add(#9 + 'TEMP' + #9 + 'HEART' + #9 + 'OXYGEN');
redX.Lines.Add(#9 + '°C'  + #9 + 'bpm'  + #9 + 'ppm');
for A := 1 to Baby.GetTeller do
  redx.Lines.Add(Baby.GetLesingX(A).ToString);
end;

procedure TVRAAG_3.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Baby.StelVry;
end;

procedure TVRAAG_3.FormCreate(Sender: TObject);
begin
  dlgSaveAs.InitialDir := ExtractFilePath(Application.ExeName);
end;

function TVRAAG_3.Geldige_Rekening(sNomX:
String): Boolean;
var
  sNom, sLyn : String;
  TF          : TextFile;
  Gevind      : Boolean;
begin
  //vraag 3.2.
  sNom := Uppercase(edtNommer.Text);
  IF (length(sNom) <> 4) AND
  NOT((sNom[1] in ['A'..'Z']) AND
(sNom[2] in ['A'..'Z'])) AND
  NOT((sNom[3] in ['0'..'9']) AND
(sNom[4] in ['0'..'9']))
  then
    begin
      Result := False;
      Exit;
    end;
  Gevind := False;
  AssignFile(TF, 'INFO.TXT');
  Reset(TF);
  While NOT EOF(TF) AND NOT Gevind DO
  begin
    Readln(TF, sLyn);
    sLyn := Uppercase(sLyn);
    IF Copy(sLyn, 1, 4) = sNom
    then
      begin
        Gevind := True;
        sPers_Info := sLyn;
      end;
    end;
  CloseFile(TF);
  IF NOT Gevind
  then
    begin
      Result := False;
      Exit;
    end;
  Result := True;
end;
end.

```