basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/ NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

2019

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 23 pages.

GENERAL INFORMATION:

- These marking guidelines must be used as the basis for the marking session. They were prepared for use by markers. All markers are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' work.

- Note that learners who provide an alternate correct solution to that given as example of a solution in the marking guidelines will be given full credit for the relevant solution, unless the specific instructions in the question paper were not followed or the requirements of the question were not met.

- **ANNEXURES A, B, C and D** (pages 3–9) include the marking grid for each question and a table for a summary of the learner's marks.

- **ANNEXURES E, F, G and H** (pages 10–23) contain examples of a programming solution for QUESTION 1 to QUESTION 4 in programming code.

- Copies of **ANNEXURES A, B, C, D** and the **summary of learner's marks** (pages 3–9) should be made for each learner and completed during the marking session.

ANNEXURE A

SECTION A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
A learner must be penalised only once if the same error is repeated.			
1.1	**Button [1.1 - Random number]** Random value generated✓ In correct range: Lower limit✓upper limit✓ Display the random number converted to a string✓	**4**	
1.2	**Button [1.2 - Calculate minutes]** Correct use of variables ✓ Extract the number of participants ✓ Calculate the number of minutes: Test if number of participants <=20✓ Set participant time = 2.5 ✓ Else ✓ if number of participants <=50 ✓ *or: Test if number of participants > 20 AND <= 50* Set participant time = 2.3 ✓ Else *or: Test if number of participants > 50* Set participant time = 2 ✓ Number of minutes= number of participants * participant time ✓ Display number of minutes ✓ to two decimal places ✓ Display number of minutes ✓ rounded to the next minute✓	**13**	
1.3	**Button [1.3 - Calculate factorial]** Extract the number from an edit box ✓ Set factorial to 1✓ Loop ✓from 1 *(or 2)* to the number✓ Multiply factorial by the value of the loop variable✓ and assign the answer to factorial ✓ Display the factorial in the edit box✓	**7**	
1.4	**Button [1.4 - Reverse words]** Extract the sentence ✓ Add a space to the end✓ Initialise variables for word✓ and new sentence✓ Loop ✓to the length of the sentence ✓ Extract character at index in sentence✓ Test if character✓ = space✓ Add word ✓ and space✓ to new sentence Set word to empty string✓ Else✓ Add character ✓ to front of word✓ Display the new sentence with reversed words✓	**16**	
	TOTAL SECTION A	**40**	

ANNEXURE B

SECTION B

QUESTION 2: MARKING GRID - DATABASE PROGRAMMING

CENTRE NUMBER:		**EXAMINATION NUMBER:**	
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1.1	**Button [2.1.1 – List of roses]** **SQL**: `SELECT * FROM tblPlants WHERE Category = "Rose"` **Concepts**: SELECT all fields ✓ FROM Correct table ✓ WHERE Condition: Category = "Rose" ✓	**3**	

2.1.2	**Button [2.1.2 - Pink roses and flowers]**
	SQL:
	`SELECT PlantCode, Category, Colour, SizeOfPot`
	`FROM tblPlants`
	`WHERE`
	`(Category = "ROSE" OR Category = "FLOWER")`
	`AND Colour LIKE "%Pink%"`
	Concepts:
	SELECT all the correct fields ✓
	FROM correct table ✓
	WHERE
	Conditions: ROSE or FLOWER ✓
	brackets around both OR conditions ✓ AND ✓
	Colour LIKE %Pink% ✓
		6	

2.1.3	**Button [2.1.3 – Average price per category]**
	`SELECT Category, Format(Avg(Price),"Currency")`
	`AS [AverageCost]`
	`FROM tblPlants GROUP BY Category`
	Concepts:
	SELECT Category ✓
	Average of price field ✓ as AverageCost✓– Format as Currency✓
	FROM tblPlants
	GROUP BY category ✓
		5	

QUESTION 2: MARKING GRID – CONTINUE

2.1.4	**Button [2.1.4 – Display information for invoice number F2]** **`SELECT InvoiceNum, Description, NumberOrdered FROM tblPlants, tblOrders WHERE tblPlants.PlantCode = tblOrders.PlantCode AND InvoiceNum = "F2";`** **Concepts**: SELECT InvoiceNum, Description, NumberOrdered fields ✓ FROM both tables ✓ WHERE Conditions: Plantcode = plantcode✓ both table names ✓ AND InvoiceNum = variable (correct format) ✓	**5**	
2.1.5	**Button [2.1.5 – Update item delivery]** **`UPDATE tblOrders SET NumberDelivered = NumberOrdered WHERE ItemNum = QuotedStr(sItemNum);`** **Concepts**: UPDATE correct table ✓SET✓ NumberDelivered = NumberOrdered ✓ WHERE ItemNum = variable (correct format) ✓	**4**	
	Subtotal: SQL	**[23]**	

2.2.1	**Button [2.2.1 – Check stock]**
	Move to first record✓
	Loop while not end of table ✓
	if (sPlantCode = tblPlants['PlantCode']) ✓
	if (iNumOrdered > tblPlants['InStock']) ✓
	cContinue = Must order be placed ('Y' or 'N') ✓
	if cContinue = 'Y'
	iNumOrdered = tblPlants['InStock'] ✓ else display 'order cancelled'
	if (iNumOrdered <= tblPlants['InStock']) ✓ OR
	order must be placed for available stock ✓
	Display output using correct fields from tblPlants in redDisplay✓
	btnQ2_2_2.Enabled ✓
	Move to next record ✓
		11	

2.2.2	**Button [2.2.2 – Place an order]** Change tblOrders to INSERT mode ✓ Set ['InvoiceNum] to F9 ✓ Set ['PlantCode'] to plant code ✓ Set ['NumberOrdered'] to number of plants ordered✓ Set ['NumberDelivered'] to 0 ✓ POST✓	**6**	
	Subtotal: Delphi code	**[17]**	
	TOTAL SECTION B	**40**	

ANNEXURE C

SECTION C

QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.1(a)	**updateHours method** Declare method ✓ with parameter of correct data type ✓ Increment the fHours attribute ✓ with parameter value ✓	**4**	
3.1.1(b)	**updateSales method** Declare a method with parameter of correct data type ✓ Increment the fSales attribute with parameter value ✓	**2**	
3.1.2	**qualifiesForBonus method** Declare a method that returns a Boolean value ✓ Test if (hours >= 15)✓ AND ✓ (sales >= 1200)✓ Assign TRUE to the result of the method Else Assign FALSE to the result of the method ✓ *Alternative:* Initialise a Boolean return variable to FALSE If test is positive assign TRUE to return variable Assign return variable to result of the method	**5**	
3.1.3	**toString method** Adding the relevant attributes (TraineeID, Hours and Sales) ✓ to the concatenated string ✓ Convert the numerical values to string ✓ Formatting the sales value to currency to two decimal places ✓	**4**	
	Subtotal: Object class	**[15]**	

QUESTION 3: MARKING GRID – CONTINUE

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 – Click to continue]** *Instantiate the objTrainee:* objTrainee✓:= TTrainee.Create✓(parameters in correct order and data type ✓) *{object variable^ := Class using create method^ (parameters^)}* Show the btnQ3_2_2 button using visible property or show method ✓	**4**	
3.2.2	**Button [3.2.2 – Process logbook data]** Test if text file exists ✓ If file does not exists then show message and close program ✓ If the file exists: Initialise a Boolean flag (found) to false ✓ AssignFile and Reset to read from file ✓ Use a loop to read up to the end of the file ✓ Read a line of text ✓ Determine if the entry ✓is related to selected trainee using the getTraineeID method ✓ Split the line of text into the entry type (T/S) and the numerical value (copy/pos/delete/ sLine[1]) ✓✓✓ Test (using IF/CASE-statement) for Tor S ✓call the relevant method with argument ✓✓ Change Boolean flag to true (found)✓ Once all the entries were process – end of file reached: If at least one trainee ID was found use toString method ✓ to display object data set btnQ2_2_3 button to visible ✓ If trainee ID was not found display a suitable message✓	**18**	
3.2.3	**Button [3.2.3 – Qualify for a bonus?]** Test by using the qualifiesForBonus method ✓ to determine if trainee qualifies for bonus ✓ Display suitable messages ✓	**3**	
	Subtotal: Form class	**[25]**	
	TOTAL SECTION C	**40**	

ANNEXURE D

SECTION D

QUESTION 4: MARKING GRID – PROBLEM SOLVING

Question	DESCRIPTION	MAX MARKS	LEARNER'S MARKS
4.1	**Button [4.1 – Separate by type]** Loop row from 1 to iTypes ✓ Set column to 0 ✓ Loop index from 1 to 24 ✓ Test if last character ✓of arrList[index] ✓ = first character ✓ of arrTypes[row] ✓ Increment column value✓ arrTrees[row,column] ✓ = arrList[index] ✓ Enable buttons btnQ4_2 and btnQ4_3 ✓	**11**	
4.2	**Button [4.2 – Display]** Loop row from 1 to iTypes ✓ Set string variable sLine to arrTypes[row] +':' ✓ Loop col from 1 to iNum ✓ Join arrTrees[row,col] ✓to the string variable sLine + space ✓ Display the string variable in the output area ✓ Loops correctly nested ✓	**7**	
4.3	**Button [4.3 – Sort alphabetically]** Loop row from 1 to iTypes ✓ Loop col from 1 to iNum ✓ Delete ✓ the last two characters✓ from arrTrees[row,col] ✓ Loop col from 1 to iNum - 1 ✓ Loop c from col + 1 to iNum ✓ Test if arrTrees[row,col] > arrTrees[row,c] ✓ Set temp = arrTrees[row,col] ✓ Set arrTrees[row,col] = arrTrees[row,c] ✓ Set arrTrees[row,c] = temp ✓ Execute code in button btnQ4_2 ✓	**12**	
	TOTAL SECTION D	**30**	

SUMMARY OF LEARNER'S MARKS:

CENTRE NUMBER:		**EXAMINATION NUMBER:**			
	SECTION A	**SECTION B**	**SECTION C**	**SECTION D**	
	QUESTION 1	**QUESTION 2**	**QUESTION 3**	**QUESTION 4**	**GRAND TOTAL**
MAX. MARKS	**40**	**40**	**40**	**30**	**150**
LEARNER'S MARKS					

ANNEXURE E: SOLUTION FOR QUESTION 1

```
unit Question1_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, ExtCtrls, Math, Grids, DBGrids;
type
  TfrmQuestion1 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    btnQ1_2: TButton;
    edtMinsRounded: TEdit;
    edtSentence: TEdit;
    btnQ1_4: TButton;
    edtMinutes: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    GroupBox4: TGroupBox;
    Label4: TLabel;
    edtReverse: TEdit;
    Label5: TLabel;
    btnQ1_3: TButton;
    edtFactorial: TEdit;
    btnQ1_1: TButton;
    edtRandomNumber: TEdit;
    spnNumber: TSpinEdit;
    edtParticipants: TEdit;
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion1: TfrmQuestion1;

implementation

{$R *.dfm}
// ============================================================================
// Question 1.1           4 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  edtRandomNumber.Text := IntToStr(Random(21) + 100);
 //edtRandomNumber.Text := IntToStr(RandomRange(100,121));
end;
```

```
// ============================================================================
// Question 1.2           13 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  iNum: integer;
  rMinutes, rHours, rFrac, rParticipantTime: real;
begin
  iNum := StrToInt(edtParticipants.Text);
  if iNum <= 20 then
     rParticipantTime := 2.5;

  if (iNum > 20) AND (iNum <= 50) then
     rParticipantTime := 2.3;

  if iNum > 50 then
     rParticipantTime := 2.0;

  rMinutes := iNum * rParticipantTime;
  edtMinutes.Text := FloatToStrf(rMinutes,ffFixed,6,2);
  edtMinsRounded.Text := IntToStr(Ceil(rMinutes));
end;
```

```
// Alternative:
  if iNum <= 20 then
    rParticipantTime := 2.5
  else
   if iNum <= 50 then
     rParticipantTime := 2.3
   else
     rParticipantTime := 2.0;
```

```
// ============================================================================
// Question 1.3           7 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  iNum, I, iFactorial: integer;
begin
  iNum := spnNumber.Value;
  iFactorial := 1;
  for I := 2 to iNum do
    iFactorial := iFactorial * I;
  edtFactorial.Text := IntToStr(iFactorial);
end;
// ============================================================================
// Question 1.4           16 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sString, sWord, sReverseString: String;
  iPos, I: integer;
begin
  sString := edtSentence.Text + ' ';
  sWord := '';
  sReverseString := '';
  for I := 1 to length(sString) do
  begin
    if sString[I] = ' ' then
    begin
      sReverseString := sReverseString + sWord + ' ';
      sWord := '';
    end
    else
    begin
      sWord := sString[I] + sWord;
    end;
  end;
  edtReverse.Text := sReverseString;
end;
end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;
// ---  Delphi and Database programming  ---
//
// Possible solution for Question 2.
// ----------------------------------------
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, ConnectDB_U, DB, ADODB, Grids,
  DBGrids, ComCtrls, DateUtils, DBCtrls;
type
  TfrmDBQuestion2 = class(TForm)
    pnlBtns: TPanel;
    bmbClose: TBitBtn;
    bmbRestoreDB: TBitBtn;
    grpTblOrders: TGroupBox;
    grpTblPlants: TGroupBox;
    dbgPlants: TDBGrid;
    dbgOrders: TDBGrid;
    tabsQ2_2ADO: TTabSheet;
    tabsQ2_1SQL: TTabSheet;
    redDisplay: TRichEdit;
    grpResults: TGroupBox;
    dbgrdSQL: TDBGrid;
    grpOutput: TGroupBox;
    pgcTabs: TPageControl;
    pnlTables: TPanel;
    btnQ2_1_1: TButton;
    btnQ2_1_3: TButton;
    btnQ2_1_2: TButton;
    btnQ2_1_4: TButton;
    btnQ2_2_1: TButton;
    btnQ2_2_2: TButton;
    GroupBox1: TGroupBox;
    cmbPlantCode: TComboBox;
    Label1: TLabel;
    Category: TLabel;
    cmbCategory: TComboBox;
    Label2: TLabel;
    edtNumPlants: TEdit;
    btnQ2_1_5: TButton;
    procedure bmbRestoreDBClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure cmbCategoryChange(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
  private
  public
  end;

var
  frmDBQuestion2: TfrmDBQuestion2;
  dbCONN: TConnection;
```

```
  // --- Provided global variables to be used ---
  tblPlants, tblOrders: TADOTable;

  // To be used in Question 2.2.1 and 2.2.2
  iNumOrdered: integer;
  sPlantCode: String;

implementation

{$R *.dfm}
{$R+}
// Question 2.1 - SQL SECTION

// ====================================================
// Question 2.1.1             3 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
  // Question 2.1.1
  sSQL1 := 'SELECT * FROM tblPlants WHERE Category = "Rose"';

  // Provided code - do not change
  dbCONN.runSQL(sSQL1);
end;

// ====================================================
// Question 2.1.2             6 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
begin
  // Question 2.1.2
  sSQL2 :=
    'SELECT PlantCode, Category, Colour, SizeOfPot FROM tblPlants WHERE '
    + ' (Category = "Rose" OR Category = "Flower") AND (Colour LIKE "%Pink%")';

  // Provided code - do not change
  dbCONN.runSQL(sSQL2);
end;

// ====================================================
// Question 2.1.3             5 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3: String;
begin
  // Question 2.1.3
  sSQL3 := 'SELECT Category, Format(Avg(Price),"Currency") AS AveragePrice' +
    ' FROM tblPlants GROUP BY Category';

  // Provided code - do not change
  dbCONN.runSQL(sSQL3);
end;
```

```
// ====================================================
// Question 2.1.4             5 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
begin
  // Question 2.1.4
  sSQL4 :=
    'SELECT InvoiceNum, Description,  NumberOrdered FROM tblPlants, tblOrders '
    + ' WHERE tblPlants.PlantCode = tblOrders.PlantCode AND InvoiceNum = "F2"';

  // Provided code - do not change
  dbCONN.runSQL(sSQL4);
end;
// ====================================================
// Question 2.1.5             4 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5, sItemNum : String;
begin
     sItemNum := InputBox('Input','Item number: ','');
  // Question 2.1.5
    sSQL5 :=
    'UPDATE tblOrders SET NumberDelivered = NumberOrdered WHERE ItemNum = ' +
sItemNum;
  // Provided code - do not change
  dbCONN.executeSQL(sSQL5,dbgOrders);
end;
```

// Question 2.2 – Section with Delphi code

```
// ====================================================
// Question 2.2.1             11 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  cContinue : char;
begin
  // Provided code
  redDisplay.Clear;
  iNumOrdered := StrToInt(edtNumPlants.Text);
  sPlantCode := cmbPlantCode.Text;
  // ====================================================
  // Question 2.2.1
  cContinue := 'Y';
  tblPlants.First;
  while (NOT tblPlants.Eof) do
  begin
    if (sPlantCode = tblPlants['PlantCode']) then
    begin
     if (iNumOrdered > tblPlants['InStock'])  then
      begin
         cContinue := InputBox('Place order?','Not enough stock. Do you want to
place an order for ' + IntToStr (tblPlants['InStock']) + ' plants?
(Y/N)','Y')[1];
         if cContinue = 'Y' then
             iNumOrdered := tblPlants['InStock']
         else
           begin
```

```
            btnQ2_2_2.Enabled := false;
            redDisplay.Lines.Add('Order cancelled');
          end;
      end;
      if (iNumOrdered <= tblPlants['InStock']) OR (cContinue = 'Y') then
      begin
       redDisplay.Lines.Add('Plant code:   ' + tblPlants['PlantCode'] + #13 +
'Colour:   ' + tblPlants['Colour'] + #13 + 'Number ordered:   ' +
IntToStr(iNumOrdered) + #13 + 'Price per item:   ' +
FloatToStrF(tblPlants['Price'],ffCurrency,8,2));
       btnQ2_2_2.Enabled := true;
      end
    end;
    tblPlants.Next;
  end;
end;
// ====================================================
// Question 2.2.2             6 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_2_2Click(Sender: TObject);
begin
  // Question 2.2.2
  tblOrders.Insert;
  tblOrders['InvoiceNum'] := 'F9';
  tblOrders['PlantCode'] := sPlantCode;
  tblOrders['NumberOrdered'] := iNumOrdered;
  tblOrders['NumberDelivered'] := 0;
  tblOrders.Post;
  showMessage('Order placed');
end;

// ========================================================
// Setup of database connections - DO NOT CHANGE!
// ========================================================
procedure TfrmDBQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  dbCONN.RestoreDatabase(dbgPlants, dbgOrders, dbgrdSQL);
  dbCONN.formatTables;
  redDisplay.Clear;
  tblPlants := dbCONN.tblOne;
  tblOrders := dbCONN.tblMany;
end;

procedure TfrmDBQuestion2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  dbCONN.dbDisconnect;
end;

procedure TfrmDBQuestion2.FormCreate(Sender: TObject);
begin
  CurrencyString := 'R';
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblPlants := dbCONN.tblOne;
  tblOrders := dbCONN.tblMany;
  dbCONN.setupGrids(dbgPlants, dbgOrders, dbgrdSQL);
  dbCONN.formatTables;
  pgcTabs.ActivePageIndex := 0;
  // Formatting display
  redDisplay.Clear;
  btnQ2_2_2.Enabled := false;
end;
```

```
procedure TfrmDBQuestion2.cmbCategoryChange(Sender: TObject);
begin
  cmbPlantCode.Clear;
  dbCONN.fillCombo(cmbCategory.Text);
  while NOT tblPlants.Eof do
  begin
    cmbPlantCode.Items.Add(tblPlants['PlantCode']);
    tblPlants.Next;
  end;
  tblPlants.Filtered := false;
  tblPlants.First;
end;


end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

OBJECT CLASS:

```
unit Trainee_U;
interface
// ============================================================================
// Provided code
// ============================================================================
type
   TTrainee = class(TObject)
     private
        fTraineeID: integer;
        fName: String;
        fHours,
        fSales: real;
     public
      constructor Create(iTraineeID: integer; sName: String);
      function toString: String;
      function getTraineeID: integer;
      function getName: String;
      procedure updateHours(rValue: real);
      procedure updateSales(rValue: real);
      function qualifiesForBonus: boolean;
   end;

implementation

uses SysUtils, DateUtils;
// ============================================================================
// Provided code
// ============================================================================
constructor TTrainee.Create(iTraineeID: integer; sName: String);
begin
   fTraineeID := iTraineeID;
   fName := sName;
   fHours := 0;
   fSales := 0;
end;

function TTrainee.getName: String;
begin
  Result := fName;
end;

function TTrainee.getTraineeID: integer;
begin
  Result := fTraineeID;
end;
// ============================================================================
// Question 3.1.1(a)        4 marks
// ============================================================================
procedure TTrainee.updateHours(rValue: real);
begin
  fHours := fHours + rValue;
end;
// ============================================================================
// Question 3.1.1(b)        2 marks
// ============================================================================
procedure TTrainee.updateSales(rValue: real);
begin
  fSales := fSales + rValue;
end;
```

```
// ============================================================================
// Question 3.1.2           5 marks
// ============================================================================
function TTrainee.qualifiesForBonus: boolean;
begin
  Result := (fHours >= 15) AND (fSales >= 1200);
end;
// ============================================================================
// Question 3.1.3           4 marks
// ============================================================================
function TTrainee.toString: String;
begin
  Result := fName + ' (' + IntToStr(fTraineeID) + ') ' +
            ' attended ' + FloatToStrF(fHours, ffFixed, 6, 2) +
            ' hours of training and ' +
            'sold plants to the value of ' +
            FloatToStrF(fSales, ffCurrency, 8, 2) + '.' ;
end;

end.
```

FORM UNIT

```
unit Question3_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls,  ComCtrls, Trainee_U, pngimage;
type
  TfrmQuestion3 = class(TForm)
    pnl1: TPanel;
    bmbClose: TBitBtn;
    pnlH: TPanel;
    imgLogo: TImage;
    imgL2: TImage;
    btnQ3_1_1: TBitBtn;
    redQ3: TRichEdit;
    btnQ3_2_2: TBitBtn;
    cmbTrainee: TComboBox;
    lbl1: TLabel;
    btnQ3_2_3: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure btnQ3_1_1Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
  public
  end;

var
  frmQuestion3: TfrmQuestion3;
  objTrainee : TTrainee;

implementation
{$R *.dfm}
{$R+}
//============================================================
```

```
procedure TfrmQuestion3.btnQ3_1_1Click(Sender: TObject);
var
   sName : String;
   iTraineeID : integer;
begin  //Question 3.2.1
 //Provided code - do not change!
 sName := cmbTrainee.Items[cmbTrainee.ItemIndex];
 iTraineeID := INTEGER(cmbTrainee.Items.Objects[
         cmbTrainee.Items.IndexOf(cmbTrainee.Items[cmbTrainee.ItemIndex])]);
// ============================================================================
// Question 3.2.1           4 marks
// ============================================================================
 objTrainee := TTrainee.Create(iTraineeID,sName);
 btnQ3_2_2.Visible := true; //OR  btnQ3_2_2.Show;
end;
// ============================================================================
// Question 3.2.2           18 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  TFile  : TextFile;
  sLine  : String;
  cType  : char;
  rValue : real;
  bFound : boolean;
  iIDx   : integer;
begin  //Question 3.2.2
  bFound := False;
  if NOT FileExists('Logbook.txt') then
   begin
     MessageDlg('File NOT found', mtInformation, [mbOK], 0);
     Exit;
   end;

  AssignFile(TFile, 'Logbook.txt');
  Reset(TFile);
  While not Eof(TFile) do
   begin
     Readln(TFile, sLine);
     iIDx := StrToInt(Copy(sLine, 1, Pos(';', sLine)-1));
     if (objTrainee.getTraineeID = iIDx) then
      begin
        Delete(sLine, 1, Pos(';', sLine));
        cType := sLine[1];
        Delete(sLine, 1, Pos('#', sLine));
        rValue := StrToFloat(sLine);
        case cType of
        'T' : objTrainee.updateHours(rValue);
        'S' : objTrainee.updateSales(rValue);
        end;
        bFound := True;
      end;
   end;
  CloseFile(TFile);
  redQ3.Clear;
  if bFound
  then
    begin
     redQ3.Lines.Add(objTrainee.toString);
     btnQ3_2_3.Visible := True;
    end
      else
```

```
     begin
       redQ3.Lines.Add('The trainee is not registered.');
       btnQ3_2_3.Visible := False;
     end;
end;

// ============================================================================
// Question 3.2.3           3 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
begin  //Question 3.2.3
   if objTrainee.qualifiesForBonus then
    begin
      redQ3.Lines.Add('The trainee qualifies for a bonus.')
    end
   else
    begin
      redQ3.Lines.Add('The trainee does NOT qualify for a bonus.');
    end;
end;
// ============================================================================
// Provided code
// ============================================================================
//{$REGION 'Provided code - do not change!'}
procedure TfrmQuestion3.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if Assigned(objTrainee) then
   begin
     objTrainee.Free;
   end;
end;

//-------------------------------------------------------------
procedure TfrmQuestion3.FormCreate(Sender: TObject);
begin
 //Add names and uniques numbers of trainees to combobox
 cmbTrainee.AddItem('Kody Shaw', TObject(10));
 cmbTrainee.AddItem('Luvuyo Bertola', TObject(11));
 cmbTrainee.AddItem('Tyrone Kemsley', TObject(12));
 cmbTrainee.AddItem('Craig Biggie', TObject(13));
 cmbTrainee.AddItem('Lindi Mahlati', TObject(14));
 cmbTrainee.AddItem('Sandy Brown', TObject(15));
 cmbTrainee.AddItem('Lindiwe Dlamini', TObject(16));
 //Ensure that the first trainee is selected.
 cmbTrainee.ItemIndex := 0;
end;

//{$ENDREGION}

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
// A possible solution for Question 4
unit Question4_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls;
type
  TfrmQuestion4 = class(TForm)
    redQ4: TRichEdit;
    btnQ4_1: TButton;
    btnQ4_2: TButton;
    btnQ4_3: TButton;
    procedure btnQ4_2Click(Sender: TObject);
    procedure btnQ4_1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btnQ4_3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion4: TfrmQuestion4;
  arrTypes: array [1..4] of String = (
    'Citrus',
    'Deciduous',
    'Nuts',
    'Tropical'
  );

  arrList: array [1..24] of String = (
    'Orange#C',
    'Hazelnut#N',
    'Apple#D',
    'Banana#S',
    'Pecan#N',
    'Pear#D',
    'Lemon#C',
    'Papaya#S',
    'Kiwi#S',
    'Apricot#D',
    'Grapefruit#C',
    'Walnut#N',
    'Lime#C',
    'Mango#S',
    'Peach#D',
    'Cashew#N',
    'Almond#N',
    'Tangerine#C',
    'Avocado#S',
    'Cherry#D',
    'Plum#D',
    'Macadamia#N',
    'Kumquat#C',
    'Guava#S'
  );
  arrTrees: array [1..4, 1..6] of String;
  iTypes: integer = 4;
  iNum: integer = 6;
```

```
implementation
{$R *.dfm}

// ============================================================================
// Question 4.1           11 marks
//============================================================================
procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  i, r, c: integer;
begin
  redQ4.Clear;
  for r := 1 to iTypes do
  begin
    c := 0;
    for i := 1 to 24 do
      if arrList[i][length(arrList[i])] = arrTypes[r][1] then
      begin
        Inc(c);
        arrTrees[r, c] := arrList[i];
      end; // if
  end; // r
  btnQ4_2.Enabled := True;
  btnQ4_3.Enabled := True;
end;

// ============================================================================
// Question 4.2           7 marks
//============================================================================
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  r, c: integer;
  sLine: String;
begin
  redQ4.Clear;

  for r := 1 to iTypes do
  begin
    sLine := arrTypes[r] + ':' +  #9 ;
    for c := 1 to iNum do
      sLine := sLine + arrTrees[r, c] + #9 ;
    redQ4.Lines.Add(sLine);
  end;
end;
```

```
// ============================================================================
// Question 4.3           12 marks
//============================================================================
procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  sTemp: String;
  i, j, c: integer;
begin
  for i := 1 to iTypes do
  begin
     for j := 1 to iNum do
       delete(arrTrees[i,j],length(arrTrees[i,j]) - 1,2);
     // end delete loop
     // sorting
     for j := 1 to iNum -1 do
       begin
       for c :=  j + 1 to iNum do
         if arrTrees[i,j] > arrTrees[i,c] then
         begin
            sTemp := arrTrees[i,j];
            arrTrees[i,j] := arrTrees[i,c];
            arrTrees[i,c] := sTemp;
         end;//if
       end; //j  for sorting
  end;//i
  btnQ4_2.Click;
end;

// ============================================================================
// Provided code
//============================================================================
procedure TfrmQuestion4.FormActivate(Sender: TObject);
begin
  redQ4.Paragraph.TabCount := 7;
  redQ4.Paragraph.Tab[0] := 20;
  redQ4.Paragraph.Tab[1] := 110;
  redQ4.Paragraph.Tab[2] := 200;
  redQ4.Paragraph.Tab[3] := 290;
  redQ4.Paragraph.Tab[4] := 380;
  redQ4.Paragraph.Tab[5] := 470;
  redQ4.Paragraph.Tab[6] := 570;
  btnQ4_2.Enabled := False;
  btnQ4_3.Enabled := False;
end;


end.
```