basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE12

INFORMATION TECHNOLOGY P1

NOVEMBER 2019 (2)

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 26 pages.

GENERAL INFORMATION:

- These marking guidelines are to 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 paper was not followed or the requirements of the question was not met

- **ANNEXURES A, B, C and D** (pages 3–9) include the marking grid for each question for using either one of the two programming languages.

- **ANNEXURES E, F, G and H** (pages 10–26) contain examples of solutions for Questions 1 to 4 in programming code.

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

ANNEXURE A

QUESTION 1: MARKING GRID- GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
1.1	**Button [1.1 - Welcome message]** Declare a string variable✓ Retrieve name✓ With label ✓ Change font style✓ Change font size✓ Display message✓	**6**	
1.2	**Button [1.2 – Random numbers]** Generate two numbers✓ in the correct range✓ Determine if ✓the first number > second number ✓ Set temp = first number ✓ Set first number = second number✓ Set second number = temp✓ Display heading 'Ascending order:' ✓ Display the sorted numbers ✓ Determine average: add two numbers✓ divide answer by 2✓ Display the average✓ formatted to one decimal ✓	**13**	
1.3	**Button [1.3 – Display the value of y]** Declare x and y as integers ✓ iX := spnQ1_3.Value; ✓ iY := Ceil ✓ (20 / ✓ sqrt(iX)); ✓ display on panel pnlQ1_3✓ 'y = ' + IntToStr(iY) ✓	**7**	
1.4	**Button [1.4 – Show output]** Display column headings✓ Set the year counter to 1✓ Loop while✓ balance >= 0✓ Calculate interest amount: balance * ✓ interest / 100 ✓ Calculate new balance: balance + interest amount ✓ Display in richedit: year counter, balance ✓ interest amount, new balance ✓ Display in column format✓ amounts formatted to currency with 2 decimal places ✓ balance = new balance - 1000 ✓ Increment year counter ✓ Display on richedit: message with year count + 1✓	**14**	
	TOTAL:	**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 – Male guides]** **SQL**: `SELECT FirstName, Surname FROM tblGuides WHERE Gender = 'M' ORDER BY Surname` **Concepts**: SELECT correct fields ✓ FROM correct table ✓ WHERE Condition: Gender = 'M' ✓ ORDER By surname✓ Asc	**4**	
2.1.2	**Button [2.1.2 – Activity]** **SQL:** `SELECT FirstName, Surname, Gender, NumExcursions FROM tblGuides WHERE Activity = '' + sAct + ''` **Concepts**: SELECT correct fields ✓ FROM correct table ✓ WHERE Activity = variable (correct format)✓	**3**	
2.1.3	**Button [2.1.3 – Highest number of excursions]** `SELECT Activity, Gender, max(NumExcursions) as HighestNumber FROM tblGuides GROUP BY Activity, Gender` **Concepts**: SELECT Activity and Gender ✓ Maximum on NumExcursions✓ as HighestNumber ✓ FROM tblGuides GROUP BY Activity ✓ and Gender✓	**5**	
2.1.4	**Button [2.1.4 – Delete guide information]** `'DELETE FROM tblGuides WHERE GuideID = ' + sGuideID` **Concepts**: DELETE FROM✓ correct table ✓ WHERE GuideID = variable (correct format) ✓	**3**	

2.1.5	**Button [2.1.5 – Knysna and Montagu]** `SELECT (Surname + '. ' + left(Firstname,1)) As GuideName, Venue, tblExcursions.Activity FROM tblExcursions,tblGuides WHERE tblExcursions.Activity = tblGuides.Activity AND Venue IN ('Knysna','Montagu')` **Concepts**: SELECT Venue, (Surname + '. ' + left(Firstname,1) ✓) ✓ AS GuideName FROM tblExcursions,tblGuides ✓ WHERE tblExcursions.Activity = tblGuides.Activity ✓ AND ✓ Venue IN ✓ ('Knysna','Montagu') ✓	**7**	
	Subtotal: SQL	**[22]**	

2.2.1	**Combobox – cmbQ2_2_1** Extract sActivity from combo box ✓ Move to first record of tblExcursions ✓ Loop while not end of table ✓ if (sActivity = tblExcursions ['Activity']) ✓ sVenue := tblExcursions ['Venue'] ✓ load picture onto image component using venue name✓ Display venue name on label lblQ2_2_1✓ Move to next record ✓	**8**	
2.2.2	**Button [2.2.2 – Booking]** Extract size of group ✓ Set boolean variable to false✓ Move to first record of tblExcursions Loop while not end of tblExcursions ✓ Test if size of group <= max group size ✓ and activity = sActivity✓ Display booking confirmed at venue✓ Set boolean to true✓ Move to next record of tblExcursions ✓ If group size too large found ✓– display no booking✓	**10**	
	Subtotal: Delphi code	**[18]**	

	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID - OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
SECTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**Constructor method** Method header ✓ with String-parameters ✓ Assign all values to correct attributes ✓ Set fAvailable attribute to True ✓	**4**	
3.1.2	**setAvailable method** Method header ✓ Set fAvailable attribute to parameter ✓	**2**	
3.1.3	**calculateRating method** Correct header and return type ✓ Set sum variable to 0 ✓ Loop from first digit to last digit of fRatings ✓ Typecast fRatings[i] to integer ✓ sum = sum + fRatings[i] ✓ Divide sum by number of ratings (length of fRatings) ✓ Return sum as result✓	**7**	
3.1.4	**hasSkill method** Result✓ := Pos ✓ (sSkill ✓, fSkills ✓) >✓ 0 ✓; **Alternative solution:** Loop while skill not found or skills <> '' (1 mark) Check for pos of ';' Extract skill up to semi-colon (1 mark) Update skills Check if skill is required skill (1 mark) Set Result to True (1 mark) Else Check if remaining skill is required skill (1 mark) Set Result to True (1 mark)	**6**	
3.1.5	**toString method** Fullname, skills and rating attributes with headings✓ Add sAvailable with heading✓ On new line✓	**3**	
	Subtotal:	**21**	

3.2.1	**Button [3.2.1 – Instantiate programmer object]** *Instantiate the objProgrammer object:* objProgrammer := ✓ TProgrammer. Create✓ Using FullName, Skills and Ratings variables ✓	**3**	
3.2.2	**Button [3.2.2 – Display programmer details]** Display object in rich edit using toString method✓ Call calculateRating ✓	**2**	
3.2.3	**Button [3.2.3 – Check availability and skills]** Test if programmer has skill using hasSkill method✓ using correct arguments✓ AND ✓ programmer is available using getAvailable method✓ Display FullName and message for availability ✓ Enable btnQ3_2_4 ✓ If programmer does not hasSkill ✓ Display message does not have skill ✓ If programmer is not getAvailable ✓ Display message not available for employment ✓	**10**	
3.2.4	**Button [3.2.4 – Employ programmer]** Call setAvailable with argument False ✓ Hide btnQ3_2_4 ✓ Display name of programmer ✓ and message	**3**	

	TOTAL SECTION C	**[40]**	

ANNEXURE D

QUESTION 4: MARKING GRID–PROBLEM SOLVING

CENTRE NUMBER:	EXAMINATION NUMBER:		
SECTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1	**Button [4.1 – Calculate average per round]** Initialise variable used to concatenate output string (sOutput) ✓ Loop through columns ✓ Initialise sum ✓ Loop through rows✓ sum = sum + arrRoundScores[iRow, iCol] ✓ Concatenate average (sum/6) to output string ✓ Display sOutput variable in redQ4 ✓	**7**	
4.2	**Button [4.2 – Determine best round(s)]** Extract ItemIndex from cmbArcher Add 1 ✓ Initialise variable to determine largest value ✓ Loop through arrRoundScores columns ✓ If arrRoundScores[iSelected, iCol] > largest ✓ largest =arrRoundScores[iSelected,iCol] ✓ Loop through arrRoundScores columns ✓ If arrRoundScores[iSelected, iCol] = largest ✓ Add round and score to output string ✓ Display output string ✓	**9**	

4.3	**Button [4.3 – Determine median score]** **Concepts:** Sorting the data correctly – 10 marks Correct indices referenced for median – 2 marks Adding middle two elements and divide by 2 – 1 mark Displaying median – 1 mark **1D array solution** Declare a suitable array used to sort Initialise placeholder used as a counter for array✓ //Move data from the two dim array to one dim array Loop through arrRoundScores rows ✓ Loop through arrRoundScores columns ✓ arrTempScores[iPH] = arrRoundScores[i, j] ✓ Increment placeholder ✓ Outer loop for sorting ✓ Inner loop for sorting ✓ If arrTempScores[i] > arrTempScores[j] ✓ iTemp = arrTempScores[i] ✓ arrTempScores[i] = arrTempScores[j] ✓ arrTempScores[j] = iTemp Determine first middle element in array ✓ Determine second middle element in array ✓ Add two middle elements together and divide by 2 ✓ Display median ✓ **2D array solution** Loop i from 1 to 5 (1 mark) Loop j from 1 to 4 (1 mark) Loop k from 1 to 5 (2 marks) Loop L from 1 to 4 (2 marks) If arr [i, j] (1 mark) > arr [k, L] (1 mark) iTemp = arr[i, j] (1 mark) arr[i, j] = arr[k, L] (1 mark) arr[k, L] = iTemp Determine first middle element in array (1 mark) Determine second middle element in array (1 mark) Add two middle elements and divide by 2 (1 mark) Display median (1 mark)	**14**	
	TOTAL SECTION D: **GRAND TOTAL:**	**30** **150**	

SUMMARY OF LEARNER'S MARKS:

	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, ComCtrls, ExtCtrls, Spin, math, DateUtils;

type
  TfrmQuestion1 = class(TForm)
    gpbQ1_2: TGroupBox;
    btnQ1_2: TButton;
    gpbQ1_3: TGroupBox;
    spnQ1_3: TSpinEdit;
    Label6: TLabel;
    btnQ1_3: TButton;
    gpbQ1_4: TGroupBox;
    gpbQ1_1: TGroupBox;
    Label1: TLabel;
    edtQ1_1: TEdit;
    lblQ1_1: TLabel;
    btnQ1_1: TButton;
    redQ1_2: TRichEdit;
    pnlQ1_3: TPanel;
    edtQ1_4: TEdit;
    Label2: TLabel;
    btnQ1_4: TButton;
    redQ1_4: TRichEdit;
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_1Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion1: TfrmQuestion1;

implementation

{$R *.dfm}
```

```
================================================================================
                          Question 1.1 – 6 marks
================================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
var
  sName: String;

begin
  // Question 1.1
  sName := edtQ1_1.Text;

  with lblQ1_1 do
  begin
    Font.Style := [fsBold];
    Font.Size := 14;
    Caption := 'Welcome ' + sName;
  end;
end;
================================================================================
                          Question 1.2 – 13 marks
================================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  // Provided variables
  iFirstNum, iSecondNum: integer;
  rAverage: real;
  // Complete
  iHoldNum: integer;

begin
  // Provided code
  redQ1_2.Clear;

  //Question 1.2
  iFirstNum := randomrange(50, 76);
  iSecondNum := randomrange(50, 76);

  if iFirstNum > iSecondNum then
  begin
    iHoldNum := iFirstNum;
    iFirstNum := iSecondNum;
    iSecondNum := iHoldNum;
  end;
  redQ1_2.Lines.Add('Ascending order:');
  redQ1_2.Lines.Add(IntToStr(iFirstNum));
  redQ1_2.Lines.Add(IntToStr(iSecondNum));
  rAverage := (iFirstNum + iSecondNum) / 2;
  redQ1_2.Lines.Add('Average: ' + FloatToStrF(rAverage, ffFixed, 7, 1));
end;
```

```
==========================================================================
                        Question 1.3 – 7 marks
==========================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  iX, iY: integer;

begin
//Question 1.3

  iX := spnQ1_3.Value;
  iY := Ceil(20 / sqrt(iX));
  pnlQ1_3.Caption := 'y = ' + IntToStr(iY);
end;
==========================================================================
                        Question 1.4 – 14 marks
==========================================================================
rocedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
// Provided constant
const
  INTEREST_RATE = 8.1;

  // Provided variables
var
  rBalance, rInterest, rNewBalance: real;
  iYearCount: integer;

begin
  // Provided code
  redQ1_4.Clear;
  rBalance := StrToFloat(edtQ1_4.Text);

 //Question 1.4

  redQ1_4.Lines.Add('Year' + #9 + 'Balance' + #9 + 'Interest' + #9 +
'Balance with interest');
  iYearCount := 1;
  while rBalance >= 0 do
  begin
    rInterest := rBalance * INTEREST_RATE / 100;
    rNewBalance := rBalance + rInterest;
    redQ1_4.Lines.Add(IntToStr(iYearCount) + #9 + FloatToStrF(rBalance,
        ffCurrency, 8, 2) + #9 + FloatToStrF(rInterest, ffCurrency, 8,
       2) + #9 + FloatToStrF(rNewBalance, ffCurrency, 8, 2));
    rBalance := rNewBalance - 1000;
    Inc(iYearCount);
  end;
  redQ1_4.Lines.Add
    (#13+'Number of years to withdraw all the money from the account: '
+ IntToStr(iYearCount));
end;
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, DB, ADODB, Grids, DBGrids, DBConnection_U, ComCtrls,
StdCtrls, jpeg, ExtCtrls, Math, Buttons, Spin;

type
  TfrmQuestion2 = class(TForm)
    pgcDatabase: TPageControl;
    dbgGuides: TDBGrid;
    dbgExcursions: TDBGrid;
    tshQ2_1: TTabSheet;
    tshQ2_2: TTabSheet;
    dbgSQL: TDBGrid;
    btnQ2_1_1: TButton;
    btnQ2_1_3: TButton;
    btnQ2_1_2: TButton;
    btnRestore: TBitBtn;
    btnQ2_1_4: TButton;
    grpBox1: TGroupBox;
    cmbQ2_2_1: TComboBox;
    btnQ2_2_2: TButton;
    grpBox2: TGroupBox;
    GroupBox4: TGroupBox;
    GroupBox5: TGroupBox;
    imgQ2_2_1: TImage;
    lblQ2_2_1: TLabel;
    spnQ2_2_2: TSpinEdit;
    redQ2_2_2: TRichEdit;
    Label1: TLabel;
    btnQ2_1_5: TButton;
    procedure FormCreate(Sender: TObject);
    procedure UpdateDBGrid();
    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 cmbQ2_2_1Change(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure btnRestoreClick(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion2: TfrmQuestion2;
```

```
  database: TConnection;
  qry: TADOQuery;
  tblGuides, tblExcursions: TADOTable;
  dsrGuides, dsrExcursions, dsrSQL: TDataSource;

  // Provided variable - Question 2.2.1 and Question 2.2.2
  sActivity: String;

implementation

{$R *.dfm}
```

Question 2.1 - SQL section

```
==========================================================================
```

Question 2.1.1 – 4 marks

```
==========================================================================
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
  // Question 2.1.1
  sSQL1 :=
    'SELECT FirstName, Surname FROM tblGuides WHERE Gender = 'M' ORDER
BY Surname';

  // Provided code
  database.RunSQL(sSQL1);
  database.setupGrids(dbgExcursions, dbgGuides, dbgSQL);
end;
==========================================================================
```

Question 2.1.2 – 3 marks

```
==========================================================================
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
  sAct: String;
begin
  // Provided code
  sAct := InputBox('Input', 'Please enter the Activity of the guide',
    'Cycling');

  // Question 2.1.2
  sSQL2 :=
    'SELECT FirstName, Surname, Gender, NumExcursions FROM tblGuides
WHERE Activity = '' + sAct + ''';

  // Provided code
  database.RunSQL(sSQL2);
  database.setupGrids(dbgExcursions, dbgGuides, dbgSQL);
end;
```

```
==========================================================================
                    Question 2.1.3 – 5 marks
==========================================================================
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3: String;
begin
  // Question 2.1.3
  sSQL3 :=
    'SELECT Activity, Gender, max(NumExcursions) as HighestNumber FROM
tblGuides GROUP BY Activity, Gender';

  // Provided code
  database.RunSQL(sSQL3);
  database.setupGrids(dbgExcursions, dbgGuides, dbgSQL);
end;
==========================================================================
                    Question 2.1.4 – 3 marks
==========================================================================
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
  sGuideID: String;
  bChange: boolean;
begin
  //Provided code
  sGuideID := InputBox('Delete guide',
    'ID number of guide to remove from table', '');

    // Question 2.1.4
  sSQL4 := 'DELETE FROM tblGuides WHERE GuideID = ' + sGuideID;

  // Provided code
  database.ExecuteSQL(sSQL4, bChange);
  database.setupGrids(dbgExcursions, dbgGuides, dbgSQL);
end;

==========================================================================
                    Question 2.1.5 – 7 marks
==========================================================================

procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5: String;
begin
  // Question 2.1.5
  sSQL5 :=
    'SELECT (Surname + '. ' + left(Firstname,1)) As GuideName, Venue,
tblExcursions.Activity FROM tblExcursions,tblGuides WHERE
tblExcursions.Activity = tblGuides.Activity AND Venue IN
('Knysna','Montagu')';

  // Provided code
  database.RunSQL(sSQL5);
  database.setupGrids(dbgExcursions, dbgGuides, dbgSQL);
end;
```

Question 2.2 - Delphi section

Question 2.2.1 – 8 marks

```
procedure TfrmQuestion2.cmbQ2_2_1Change(Sender: TObject);
var
  sVenue: String;
begin
// Question 2.2.1
  sActivity := cmbQ2_2_1.Text;
  tblExcursions.First;
  while NOT tblExcursions.eof do
  begin
    if sActivity = tblExcursions['Activity'] then
    begin
      sVenue := tblExcursions['Venue'];
      imgQ2_2_1.Picture.LoadFromFile(sVenue + '.jpg');
      lblQ2_2_1.Caption := sVenue;
    end;
    tblExcursions.Next;
  end;
end;
```

Question 2.2.2 – 10 marks

```
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  iSizeGroup: integer;
  bFound: boolean;
begin
  redQ2_2_2.Clear;
  bFound := false;
  iSizeGroup := spnQ2_2_2.Value;

  tblExcursions.First;
  while NOT tblExcursions.eof do
  begin
    if (tblExcursions['Activity'] = sActivity) then
    begin
      redQ2_2_2.Lines.Add('Maximum group size: ' + IntToStr
          (tblExcursions['MaxGroupSize']));
      if (iSizeGroup <= tblExcursions['MaxGroupSize']) then
      begin
        bFound := true;
      end;
    end;
    tblExcursions.Next;
  end;

  if NOT bFound then
    redQ2_2_2.Lines.Add('Group too large')
  else
    redQ2_2_2.Lines.Add('Booking confirmed');
end;
```

```
================================================================================
                    Setup DB connections - DO NOT CHANGE!
================================================================================
procedure TfrmDBQuestion2.FormCreate(Sender: TObject);
begin
  database := TConnection.Create();
  database.DBConnect();
  tblExcursions := database.tblOne;
  tblGuides := database.tblMany;
  database.setupGrids(dbgExcursions, dbgGuides, dbgSQL);
  pgcDatabase.ActivePageIndex := 0;
end;

procedure TfrmDBQuestion2.btnRestoreClick(Sender: TObject);
var
  bFail: boolean;
begin
  database.RestoreDatabase;
  redQ2_2_2.Clear;
  database.setupGrids(dbgExcursions, dbgGuides, dbgSQL);
end;

procedure TfrmDBQuestion2.UpdateDBGrid;
var
  i: integer;
begin
  for i := 0 to dbgSQL.Columns.Count - 1 do
  begin
    dbgSQL.Columns[i].Width := 79;
  end;
end;
end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

OBJECT CLASS UNIT:

```
unit Programmer_U;

interface

type
  TProgrammer = class(TObject)
    private
      fFullName, fSkills, fRatings: String;
      fAvailable: Boolean;
    public
      constructor Create(sFullName, sSkills, sRatings: String);
      procedure setAvailable(bAvailable: boolean);
      function getFullName: String;
      function getAvailable: boolean;
      function calculateRating: double;
      function hasSkill(sSkill: String): boolean;
      function toString: String;
  end;

implementation

uses
  SysUtils;

{ TProgrammer }
```

```
==========================================================================
                        Question 3.1.1 – 4 marks
==========================================================================
constructor TProgrammer.Create(sFullName, sSkills, sRatings: String);
begin
  fFullName := sFullName;
  fSkills := sSkills;
  fRatings := sRatings;
  fAvailable := True;
end;

==========================================================================
                        Question 3.1.2 – 2 marks
==========================================================================
procedure TProgrammer.setAvailable(bAvailable: boolean);
begin
  fAvailable := bAvailable;
end;
```

Question 3.1.3 – 7 marks

```
function TProgrammer.calculateRating: double;
var
    i, iTotal: integer;
begin
  iTotal := 0;
  for i := 1 to Length(fRatings) do
    begin
      iTotal := iTotal + StrToInt(fRatings[i]);
    end;
  Result := iTotal / Length(fRatings);
end;
```

Question 3.1.4 – 6 marks

```
function TProgrammer.hasSkill(sSkill: String): boolean;
begin
  Result := Pos(sSkill, fSkills) > 0;
end;
```

Question 3.1.5 – 3 marks

```
function TProgrammer.toString: String;
var
    sAvailable: String;
begin
```

Provided Code

```
  if fAvailable then
    begin
      sAvailable := 'Yes';
    end
  else
    begin
      sAvailable := 'No';
    end;
```

Complete Code

```
  Result := 'Full name: ' + fFullName + #13 +
            'Skills: ' + fSkills + #13 +
            'Ratings: ' + fRatings + #13 +
            'Available: ' + sAvailable;
end;
```

Provided Code

```
function TProgrammer.getAvailable: Boolean;
begin
  Result := fAvailable;
end;

function TProgrammer.getFullName: String;
begin
  Result := fFullName;
end;
end.
```

MAIN CLASS UNIT:

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    gbxQ3_2_1: TGroupBox;
    btnQ3_2_1: TButton;
    gbxQ3_2_2: TGroupBox;
    redQ3_2_2: TRichEdit;
    btnQ3_2_2: TButton;
    gbx3_2_3: TGroupBox;
    cmbQ3_2_3: TComboBox;
    btnQ3_2_3: TButton;
    Label1: TLabel;
    GroupBox1: TGroupBox;
    btnQ3_2_4: TButton;
    lblQ3_2_4: TLabel;
    pnlQ3_2_2: TPanel;
    pnlQ3_2_3: TPanel;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnQ3_2_4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;uses Programmer_U;

{$R *.dfm}

const
  sFullName = 'Rupert Grint';
  sSkills = 'Web design;JavaScript';
  sRatings = '434545445';

var
  objProgrammer: TProgrammer;
```

```
================================================================================
                        Question 3.2.1 – 3 marks
================================================================================
procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
begin
  objProgrammer := TProgrammer.Create(sFullName, sSkills, sRatings);
end;

================================================================================
                        Question 3.2.2 – 2 marks
================================================================================
procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
begin
  redQ3_2_2.Text := objProgrammer.toString;
  pnlQ3_2_2.Caption := 'Average rating: ' +
FloatToStrF(objProgrammer.calculateRating,ffFixed,8,2);
end;

================================================================================
                        Question 3.2.3 – 10 marks
================================================================================
procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
  var
    sRequiredSkill: String;
begin
  sRequiredSkill := cmbQ3_2_3.Text;

  if (objProgrammer.hasSkill(sRequiredSkill) = True) and
(objProgrammer.getAvailable = True) then
    begin
      pnlQ3_2_3.Caption := objProgrammer.getFullName + ' is available
for employment.';
      btnQ3_2_4.Enabled := True;
    end;
  if objProgrammer.hasSkill(sRequiredSkill) = False then
    begin
      pnlQ3_2_3.Caption := objProgrammer.getFullName + ' does not have
the required skill.';
    end;
  if objProgrammer.getAvailable = False then
    begin
      pnlQ3_2_3.Caption := objProgrammer.getFullName + ' is not
available for employment.';
    end;
end;

================================================================================
                        Question 3.2.4 – 3 marks
================================================================================
procedure TfrmQuestion3.btnQ3_2_4Click(Sender: TObject);
begin
  objProgrammer.setAvailable(False);
  btnQ3_2_4.Hide;
  lblQ3_2_4.Caption := objProgrammer.getFullName + ' has been employed
successfully!';
end;
end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;

interface

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

type
  TfrmQuestion4 = class(TForm)
    gpbQ4_3: TGroupBox;
    gpbQ4_2: TGroupBox;
    lblQ4_2: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    cmbArcher: TComboBox;
    gpbQ4_1: TGroupBox;
    redQ4_1: TRichEdit;
    btnQ4_1: TButton;
    btnQ4_2: TButton;
    btnQ4_3: TButton;
    pnlQ4_3: TPanel;
    procedure btnQ4_1Click(Sender: TObject);
    procedure btnQ4_3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);
  private
    procedure DisplayTournamentData();
  public
    { Public declarations }
  end;

var
  frmQuestion4: TfrmQuestion4;

implementation

{$R *.dfm}

var
  arrArchers: array [1 .. 6] of string = (
    'Nicolas Green',
    'Thomas Ndlovu',
    'Alice Garcia',
    'Jayshree Patel',
    'Tristan Evermore',
    'Eugene Geary'
  );
  arrRoundScores: array [1 .. 6, 1 .. 4] of Integer = ((7, 7, 6, 7),
    (9, 10, 10, 8), (6, 6, 10, 8), (5, 10, 9, 6), (10, 4, 8, 4), (8, 5,
7, 8));
```

```
================================================================================
                          Question 4.1 – 7 marks
================================================================================
procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  iRow, iCol, iSum: Integer;
  sOutput: String;
begin
  // Question 4.1
  sOutput := 'Average' + #9;
  for iCol := 1 to Length(arrRoundScores[1]) do
  begin
    iSum := 0;
    for iRow := 1 to Length(arrRoundScores) do
    begin
      iSum := iSum + arrRoundScores[iRow, iCol];
    end;
    sOutput := sOutput + FloatToStrF(iSum / 6, ffFixed, 6, 2) + #9;
  end;
  redQ4_1.Lines.Add(sOutput);
end;

================================================================================
                          Question 4.2 – 9 marks
================================================================================
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  iCol, iSelected, iMax: Integer;
  sOut: String;
begin
  // Question 4.2
  iSelected := cmbArcher.ItemIndex + 1;

  iMax := 0;
  for iCol := 2 to Length(arrRoundScores[1]) do
  begin
    if iMax < arrRoundScores[iSelected, iCol] then
    begin
      iMax := arrRoundScores[iSelected, iCol];
    end;
  end;

  sOut := '';
  for iCol := 1 to Length(arrRoundScores[1]) do
  begin
    if iMax = arrRoundScores[iSelected, iCol] then
    begin
      sOut := sOut + 'Round ' + IntToStr(iCol) + ' (' + IntToStr(iMax) +
'), ';
    end;
  end;
  Delete(sOut, Length(sOut) - 1, 2);

  lblQ4_2.Caption := sOut;
end;
```

```
================================================================================
                          Question 4.3 – 14 marks
================================================================================
procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  i, j, iTemp: Integer;
  arrTempScores: array [1 .. 24] of Integer;
  dMedian: double;
begin
  // Question 4.3
  iTemp := 1;
  for i := 1 to Length(arrRoundScores) do
  begin
    for j := 1 to Length(arrRoundScores[1]) do
    begin
      arrTempScores[iTemp] := arrRoundScores[i, j];
      Inc(iTemp);
    end;
  end;

  for i := 1 to Length(arrTempScores) - 1 do
  begin
    for j := i + 1 to Length(arrTempScores) do
    begin
      if arrTempScores[i] > arrTempScores[j] then
      begin
        iTemp := arrTempScores[i];
        arrTempScores[i] := arrTempScores[j];
        arrTempScores[j] := iTemp;
      end;
    end;
  end;

  dMedian := (arrTempScores[Length(arrTempScores) div 2] + arrTempScores
      [Length(arrTempScores) div 2 + 1]) / 2;
  pnlQ4_3.Caption := 'The median score is: ' + FloatToStr(dMedian);
end;
{$REGION}
================================================================================
Provided code – do not modify
================================================================================
procedure TfrmQuestion4.DisplayTournamentData;
var
  sLine: string;
  iRow, iCol, iSum: Integer;
begin
  sLine := '';
  for iCol := 1 to Length(arrRoundScores[1]) do
  begin
    sLine := sLine + #9 + 'Round ' + IntToStr(iCol);
  end;
  redQ4_1.Lines.Add(sLine);

  for iRow := 1 to Length(arrRoundScores) do
  begin
    sLine := arrArchers[iRow];
```

```
    for iCol := 1 to Length(arrRoundScores[1]) do
    begin
      sLine := sLine + #9 + IntToStr(arrRoundScores[iRow, iCol]);
    end;
    redQ4_1.Lines.Add(sLine);
  end;
end;

procedure TfrmQuestion4.FormCreate(Sender: TObject);
var
  i: Integer;
begin
  // Provided code - do not modify.
  redQ4_1.Paragraph.TabCount := 7;
  redQ4_1.Paragraph.Tab[1] := 140;
  redQ4_1.Paragraph.Tab[2] := 220;
  redQ4_1.Paragraph.Tab[3] := 300;
  redQ4_1.Paragraph.Tab[4] := 380;
  redQ4_1.Paragraph.Tab[5] := 460;

  for i := 1 to Length(arrArchers) do
  begin
    cmbArcher.Items.Add(arrArchers[i]);
  end;
  cmbArcher.ItemIndex := 0;

  DisplayTournamentData();

end;
{$ENDREGION}

end.
```