WESTERN CAPE EDUCATION DEPARTMENT

INFORMATION TECHNOLOGY P1

2023

These marking guidelines consist of 29 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 to 11) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 12 to 29) contain examples of solutions for Question 1 to Question 4 in programming code.

- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 11) should be made for each learner and completed during the marking session.

ANNEXURE A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

NAME AND SURNAME:		**FOLDER:**	
	DESCRIPTION	**MAX MARKS**	**LEARNER'S MARKS**
1.1	OnShow event created ✓ Code used to do the following: Hide lstInteresting list box ✓ Disable pnlQ1_4 ✓ Load image from file to TImage component ✓ Set proportional or stretch property = true for TImage ✓	5	
1.2	Use Case statement or IF … ELSE IF ✓ If Show numbers selected ✓ Show LstInteresting ✓ Else if Register selected ✓ Enable pnlQ1_4 ✓ Else if nothing selected (depending on how selection statement is structured could use ItemIndex = -1) ✓ display error message on MessageBox ✓	7	
1.3.1	Global variables declared for Number name and Code value (used in 1.3.2) ✓ both Strings ✓ Name of number read from edtName (stored in global variable) ✓ First letter of number name stored in code (using index 1 or copy) ✓ Loop checking for spaces ✓ IF space found ✓ add character at position space + 1 to code ✓ Make character uppercase (or make entire string upper case at the beginning or end) ✓ END IF END LOOP Display code on edtCode ✓	9	
1.3.2	Concatenate code (using global variable) ✓ + ': ' + name of number (using global variable) ✓ Add to items in list box (lstInteresting.Items.Add) ✓	3	

1.4	Extract data from edtFirstname and edtSurname ✓ Generate a random integer in the range 10000 to 99999 inclusive [Random ✓ (90000) + 10000 ✓ OR RandomRange(10000, 100000)] Store string/array of symbols in a variable (?#@*%!$)✓ Generate TWO random numbers ✓ range 1 to 7 inclusive [Random(7) + 1 OR RandomRange(1, 8)] (do not penalise range a second time) Choose TWO random symbols from the string/array ✓ (using loop or indices) (no need to check that they are different symbols) Concatenate registration code: <initial> + <surname> + <random integer> (converted to string) ✓ + <symbols> ✓ Display on caption of pnlQ1_4 ✓	9	
1.5	OUTER LOOP from 1 to 10000 ✓ (could start with a number <= 6 if they make the assume that the first 5 numbers cannot be perfect) Set sum variable = 0 ✓ INNER `I` LOOP from 1 to OUTER `iNumber` loop controller DIV 2 (accept if loop to outer loop controller value, but unnecessary) ✓ IF outer `iNumber` MOD INNER `I` = 0 then ✓ sum = sum + INNER `I` ✓ END INNER `I` LOOP IF sum = OUTER `iNumber` then ✓ Display OUTER `iNumber` (or sum) on rich edit as perfect number ✓ END OUTER `iNumber` LOOP	7	
	Total Section A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – SQL AND DATABASE PROGRAMMING

NAME AND SURNAME:		FOLDER:	
	DESCRIPTION	**MAX MARKS**	**LEARNER'S MARKS**
2.1.1	SELECT * FROM tblUploaders ✓ ORDER BY ✓ Country, UserName ✓ //also accept only a member field	**3**	
2.1.2	SELECT VideoName, NumLikes FROM tblVideos ✓ WHERE (VideoName LIKE ✓ "%dog%") OR ✓ (VideoName LIKE "%puppy%✓")	**4**	
2.1.3	SELECT UserName FROM tblUploaders ✓ WHERE Website IS ✓ NULL ✓ ALTERNATIVE: SELECT username FROM tblUploaders WHERE NOT (website LIKE "%")	**3**	
2.1.4	SELECT VideoName, ROUND✓ (NumLikes / (NumLikes+NumDislikes) *10 ✓) AS Rating ✓ FROM tblVideos WHERE ROUND(NumLikes/(NumLikes+NumDislikes) *10) > 7 ✓	**4**	
2.1.5	SELECT Country, ✓ AVG(Views) ✓ AS [AverageViews] FROM tblUploaders U, tblVideos V ✓ WHERE U.userID = V.userID ✓ GROUP BY Country ✓	**5**	
2.1.6	INSERT ✓ INTO tblUploaders ✓ Values("JDoe","John Doe",#2000-01-01#, "http://www.jdblog.com","South Africa") ✓ // Note will be different for each learner	**3**	
		[22]	

2.2.1	tblVideos.First;
	while NOT tblVideos.Eof do
	begin
	if (tblVideos['NumLikes'] < tblVideos['NumDislikes']) then
	redOutput.Lines.Add(
	tblVideos['VideoName'] +#9+ IntToStr(tblVideos['NumLikes'])
	+#9+ IntToStr(tblVideos['NumDislikes']));
	tblVideos.Next;
	end;
	end;
	
	Place pointer on first record ✓
	Loop through whole dataset ✓
	Correct if and condition statement ✓
	Display the correct fields in the richedit ✓
	Move pointer to next record ✓
		5	

2.2.2	tblUploaders.First;
	while NOT tblUploaders.Eof do
	begin
	if sSearch = tblUploaders['UserName'] then
	begin
	tblUploaders.Edit;
	tblUploaders['Website'] := sWebsite;
	tblUploaders.Post;
	end;
	tblUploaders.Next;
	end;
	
	Compare username from input box to tblUploaders ✓
	Change state of table to Edit ✓
	Change record with provide string ✓
	Save change to database by Post / navigate ✓
		4	

2.2.3	tblUploaders.First;
	while NOT tblUploaders.eof do
	begin
	sUserName := tblUploaders['UserID'];
	iCount := 0;
	tblVideos.First;
	while NOT tblVideos.eof do
	begin
	if tblVideos['UserID'] = sUserName then
	begin
	Inc(iCount);
	end;
	tblVideos.Next;
	end;
	tblUploaders.Next;
	redOutput.Lines.Add(sUserName + #9 +IntToStr(iCount));
	end;
	
	tblUploaders.first ✓
	loop length of tblUploaders ✓
	set counter = 0 ✓
	tblVideo.first ✓
	loop length of tblVideos ✓
	if Primary Key = Foreign Key ✓
	counter = counter + 1 ✓
	tblVideo.next ✓ (both .next used correct)
	tblUploaders.next
	Display correct names and values ✓
		9	

		[18]	
	Total Section B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTED PROGRAMMING

NAME AND SURNAME:	FOLDER:

	DESCRIPTION	MAX MARKS	LEARNER'S MARKS
3.1.1	**Procedure setGravity** private ✓ procedure without parameter ✓ if statement using fPlanetName for each planet ✓ stores value in fGravity ✓	**4**	
3.1.2	**Constructor method** constructor method declared ✓ two arguments: name (string) and distance (real) ✓ saved in corresponding fValues ✓ setGravity method call ✓	**4**	
3.1.3	**Procedure setDistance** public procedure ✓ with real value argument ✓ stores value to fDistance ✓	**3**	
3.1.4	**function getGravity** public function getGravity : real ✓ result := fGravity ✓	**2**	
3.1.5	**Function getTime** public function returns integer ✓ result := fDistance/300000 ✓ CEIL() ✓ / RoundTo / Round / Floor	**3**	
3.1.6	**Function toString** ToString returns string floatTostr(fGravity) and floatToStr(fDistance) ✓ display both 'km' and 'm/s' units ✓ #9 ✓ #13 / linebreak ✓	**4**	
		20	

3.2.1	Extract spin edit value ✓ Extract name from radiogroup ✓ objPlanet := TPlanet ✓.Create(sName, rDistance) ✓ redOut.lines.add(objPlanet.ToString) ✓	**5**	

3.2.2	minutes = objPlanet.getTime DIV 60 ✓ //alternative correct calculations
	seconds = objPlanet.getTime MOD 60 ✓ //alternative correct calculations
	
	output string formatted to mm:ss ✓
		3	

3.2.3	objPlanet.setDistance() ✓ edtNewDistance.text argument as real ✓ ToString called ✓	**3**	
3.2.4	obtain weight from edit box ✓ strToFloat (for edit box weight) ✓ for loop (from 1 to 4) ✓ instantiate each objPlanet ✓ using arrNames[i],arrDistance[i] ✓ EarthWeight / EARTHGRAVITY ✓ * objPlanet.getGravity ✓ output name and weight to RichEdit in columns ✓ weight formatted to one decimal ✓	**9**	
		20	
	Total Section C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING

NAME AND SURNAME:	FOLDER:		
	DESCRIPTION	**MAX MARKS**	**LEARNER'S MARKS**
4.1	Initialize the counter ✓ to 0 (Check logic) Assign text file name to textfile variable ✓ Open text file for reading from existing file ✓ If file not found, show notification to user ('File can't be accessed.') ✓ Leave method (Exit) ✓ Loop through text file ✓ Increase counter ✓ Read line from text file ✓ Extract athlete name ✓ and assign to correct array ✓ Extract Country ✓ and assign to correct array ✓ Assign distance to correct array ✓ Casted as float ✓ Close the text file	**14**	
4.2.1	Display column headings in neat columns ✓ Loop 1 – 8 (num rows) ✓ Output string athlete name✓ Initialise numJumps and TotalDistance for athlete ✓ Loop 1- 6 (number columns) ✓ If jump not foul✓ Inc numJumps✓ else Inc iFouls✓ TotalDistance TotalDistance + Distance ✓ Output string Output string + 2D element✓ Avg TotalDistance/NumJumps ✓ Output string Output string + Avg + BestJump ✓ Display Output string ✓	**13**	
4.2.2	Use output dialog **(with appropriate message)** ✓ to display the total number of fouls ✓ **typecast integer value to string** ✓	**3**	
	Total Section D:	**30**	

SUMMARY OF LEARNER'S MARKS:

CENTRE NUMBER:		**LEARNER'S 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;
//Place name and surname here

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,
  Vcl.ComCtrls,
  Vcl.Imaging.jpeg, Math, Vcl.Buttons;

type
  TfrmQuestion1 = class(TForm)
    pnlQ1_2: TPanel;
    pnlQ1_3: TPanel;
    pnlQ1_4: TPanel;
    pnlQ1_5: TPanel;
    imgQ1: TImage;
    btnQ1_2: TButton;
    Label1: TLabel;
    edtFirstName: TEdit;
    Label2: TLabel;
    edtSurname: TEdit;
    pnlCode: TPanel;
    btnQ1_4: TButton;
    lstInteresting: TListBox;
    Label3: TLabel;
    Label4: TLabel;
    edtName: TEdit;
    edtCode: TEdit;
    btnQ1_3_1: TButton;
    Q1_3_2: TButton;
    btnQ1_5: TButton;
    redDisplay: TRichEdit;
    rgpQ1_2: TRadioGroup;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    bbnClose: TBitBtn;
    procedure FormShow(Sender: TObject);
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_3_1Click(Sender: TObject);
    procedure Q1_3_2Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure btnQ1_5Click(Sender: TObject);
  private
    { Private declarations }
    sCode, sNumberName: String;
  public
    { Public declarations }
  end;

var
  frmQuestion1: TfrmQuestion1;

implementation

{$R *.dfm}

procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
begin
```

```
  case rgpQ1_2.ItemIndex of
    0:
      lstInteresting.Show; // lstInteresting.visible := true;
    1:
      pnlQ1_4.Enabled := True;
  else
    ShowMessage('Please select an option.');
  end;
end;

procedure TfrmQuestion1.btnQ1_3_1Click(Sender: TObject);
var
  iLetter, i: Integer;
begin
  sNumberName := edtName.Text;
  sCode := sNumberName[1];
  for i := 2 to Length(sNumberName)-1 do
  begin
    if sNumberName[i] = ' ' then  sCode := sCode + (sNumberName[i+1]);
  end;
  edtCode.Text := Uppercase(sCode);

  //OR

{  sCode := '';
  iLetter := 1;
  sNumberName := edtNumber.Text;
  sCode := UpCase(sNumberName[1]);
  while pos(' ', sNumberName) > 0 do
  begin
    Inc(iLetter);
    if sNumberName[iLetter] = ' ' then
    begin
      sCode := sCode + UpCase(sNumberName[iLetter + 1]);
      Delete(sNumberName, 1, iLetter); // or iLetter + 1
      iLetter := 1;
    end;
  end;
  edtCode.Text := sCode;   }

end;

procedure TfrmQuestion1.Q1_3_2Click(Sender: TObject);
var
  sItem: String;
begin
  sItem := sCode + ': ' + sNumberName;
  lstInteresting.Items.Add(sItem);
end;

procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sFirstName, sSurname, sRegCode, sNumberName, sSymbols, sSymbolPart: String;
  iRandom: Integer;
  i: Integer;
begin
  sSymbols := '?#@*%!$';
  sSymbolPart := '';

  sFirstName := edtFirstName.Text;
  sSurname := edtSurname.Text;

  iRandom := Random(90000) + 10000; // randomrange(10000, 100000);
```

```
  for i := 1 to 2 do
  begin
    sSymbolPart := sSymbolPart + sSymbols[random(7) + 1];
  end;

  sRegCode := sFirstName[1] + sSurname + IntToStr(iRandom) + sSymbolPart;
  pnlCode.Caption := sRegCode;
end;

procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  iNumber, iSum: Integer;
  i: Integer;
begin

  for iNumber := 1 to 10000 do
  begin
    iSum := 0;
    for i := 1 to iNumber DIV 2 do  //only need to check factors upto half the value of
the number
    begin
      if iNumber MOD i = 0 then
      begin
        iSum := iSum + i;
      end;

    end;
    if iSum = iNumber then
    begin
      redDisplay.Lines.Add(IntToStr(iSum));
    end;


  end;
  // 6, 28, 496, 8128, 33550336
end;

procedure TfrmQuestion1.FormShow(Sender: TObject);
begin
  lstInteresting.Hide; // lstInteresting.visible := false;
  pnlQ1_4.Enabled := False;

  imgQ1.picture.LoadFromFile('Numbers.jpg');
  imgQ1.Proportional := True;
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Q2_u;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, DBConnection_u, ADODB, DB, Vcl.Grids,
  Vcl.DBGrids, Vcl.ComCtrls, Vcl.StdCtrls, Vcl.Buttons;

type
  TfrmQuestion2 = class(TForm)
    dbgUploaders: TDBGrid;
    dbgVideos: TDBGrid;
    btnRestore: TButton;
    gpbUploaders: TGroupBox;
    gpbVideos: TGroupBox;
    PageControl1: TPageControl;
    tabQ2_1: TTabSheet;
    tabQ2_2: TTabSheet;
    btnQ2_1_1: TButton;
    btnQ2_1_2: TButton;
    btnQ2_1_3: TButton;
    btnQ2_1_4: TButton;
    btnQ2_1_5: TButton;
    btnQ2_1_6: TButton;
    redOutput: TRichEdit;
    gpbReports: TGroupBox;
    BitBtn1: TBitBtn;
    btn2_2_1: TButton;
    btn2_2_2: TButton;
    btn2_2_3: TButton;
    TabDataTables: TTabSheet;
    grbResults: TGroupBox;
    dbgSQL: TDBGrid;
    procedure FormCreate(Sender: TObject);
    procedure btnRestoreClick(Sender: TObject);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
    procedure btnQ2_1_6Click(Sender: TObject);
    procedure btn2_2_1Click(Sender: TObject);
    procedure btn2_2_2Click(Sender: TObject);
    procedure btn2_2_3Click(Sender: TObject);
  private
    // ===PROVIDED CODE - DO NOT MODIFY!==//
    con: TADOConnection;

    tblUploaders: TADOTable;
    dbsUploaders: TDataSource;

    tblVideos: TADOTable;
    dbsVideos: TDataSource;

    qry: TADOQuery;
    dbsSQL: TDataSource;

    sSQL: String;
    procedure DBSetup;
    procedure AdjustColumns(iNumCol: integer);
    // ====================================//
```

```
  public
    { Public declarations }
  end;

var
  frmQuestion2: TfrmQuestion2;

implementation

{$R *.dfm}
{ TForm1 }

procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
begin // QUESTION 2.1.1
  sSQL := 'SELECT * FROM tblUploaders ORDER BY Country, UserName';

  qry.SQL.Clear;
  qry.SQL.Add(sSQL);
  AdjustColumns(5);
end;

procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
begin // QUESTION 2.1.2
  sSQL := 'SELECT VideoName, NumLikes FROM tblVideos WHERE VideoName LIKE "%dog%" OR
VideoName LIKE "%puppy%"';

  qry.SQL.Clear;
  qry.SQL.Add(sSQL);
  AdjustColumns(2);
end;

procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
begin // QUESTION 2.1.3
  sSQL := 'SELECT UserName FROM tblUploaders WHERE Website IS NULL';

  qry.SQL.Clear;
  qry.SQL.Add(sSQL);
  AdjustColumns(1);
end;

procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
begin // QUESTION 2.1.4
  sSQL := 'SELECT VideoName, Round(NumLikes/(NumLikes + NumDislikes)*10,0) AS [Rating]
FROM tblVideos WHERE Round(NumLikes/(NumLikes + NumDislikes)*10,0) > 7';

  qry.SQL.Clear;
  qry.SQL.Add(sSQL);
  AdjustColumns(2);
end;

procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
begin // QUESTION 2.1.5
  sSQL := 'SELECT Country, Avg(Views) As [AverageViews] FROM tblUploaders U, tblVideos V
WHERE U.UserID = V.UserID GROUP BY Country';

  qry.SQL.Clear;
  qry.SQL.Add(sSQL);
  AdjustColumns(2);
end;

procedure TfrmQuestion2.btnQ2_1_6Click(Sender: TObject);
begin // QUESTION 2.1.6
  sSQL := 'Insert INTO tblUploaders Values("JSmith","Jake
```

```
Smith",#2021/05/22#,"http://www.jsblog.com","South Africa")';

  qry.SQL.Clear;
  qry.SQL.Add(sSQL);
  qry.ExecSQL;

  qry.SQL.Clear;
  qry.SQL.Add('SELECT * FROM tblUploaders');
  qry.Open;
end;

procedure TfrmQuestion2.btn2_2_1Click(Sender: TObject);
begin // QUESTION 2.2.1
  // =====GIVEN CODE===== //
  redOutput.Paragraph.TabCount := 2;
  redOutput.Paragraph.Tab[0] := 220;
  redOutput.Paragraph.Tab[1] := 250;
  redOutput.Clear;
  redOutput.Lines.Add('Video Name:'#9'Likes'#9'Dislikes');
  // =====WRITE YOUR CODE HERE=====//
  tblVideos.First;
  while NOT tblVideos.eof do
  begin
    if tblVideos['NumLikes'] < tblVideos['NumDislikes'] then
    begin
      redOutput.Lines.Add(tblVideos['VideoName'] + #9 +
        IntToStr(tblVideos['NumLikes']) + #9 + IntToStr(tblVideos['NumDislikes']))
    end;
    tblVideos.Next;
  end;
end;

procedure TfrmQuestion2.btn2_2_2Click(Sender: TObject);
var
  sSearch, sWebsite: String;
begin // QUESTION 2.2.2
  // =====GIVEN CODE===== //
  sSearch := Inputbox('Username', 'Enter the username:', '');
  sWebsite := Inputbox('New Website', 'Enter the website:', 'http://');
  // =====WRITE YOUR CODE HERE=====//
  tblUploaders.First;
  while NOT tblUploaders.eof do
  begin
    if tblUploaders['UserName'] = sSearch then
    begin
      tblUploaders.Edit;
      tblUploaders['Website'] := sWebsite;
      tblUploaders.Post;
    end;
    tblUploaders.Next;
  end;
end;

procedure TfrmQuestion2.btn2_2_3Click(Sender: TObject);
var
  sUserName: string;
  iCount : integer;
begin // QUESTION 2.2.3
  // ===== GIVEN CODE =====//
  redOutput.Paragraph.TabCount := 1;
  redOutput.Paragraph.Tab[0] := 80;
  redOutput.Clear;
  redOutput.Lines.Add('Username'#9'Num of Videos');
  // =====WRITE YOUR CODE HERE=====//
```

```
  tblUploaders.First;
  while NOT tblUploaders.eof do
  begin
    sUserName := tblUploaders['UserID'];
    iCount := 0;
    tblVideos.First;
    while NOT tblVideos.eof do
    begin
      if tblVideos['UserID'] = sUserName then
      begin
         Inc(iCount);
      end;
      tblVideos.Next;
    end;
    tblUploaders.Next;
    redOutput.Lines.Add(sUserName + #9 +IntToStr(iCount));
  end;
end;


procedure TfrmQuestion2.AdjustColumns(iNumCol: integer);
var    // PROVIDED CODE - DO NOT MODIFY!
  iCnt, iW: integer;
begin
//DISPLAY
  dbgSQL.DataSource := dbsSQL;
  qry.Open;

//ADJUSTCOLUMNS
  iW := 680 DIV iNumCol;
  for iCnt := 0 to iNumCol - 1 do
  begin
    dbgSQL.Columns[iCnt].Width := iW;
  end;

end;

procedure TfrmQuestion2.btnRestoreClick(Sender: TObject);
begin
  // PROVIDED CODE - DO NOT MODIFY!
  con.Close;
  tblUploaders.Close;
  tblVideos.Close;

  DeleteFile('Question2.mdb');
  CopyFile('Question2_BackUp.mdb', 'Question2.mdb', False);

  TDBConnection.DBConnect(con, Self);
  tblUploaders.Open;
  dbgUploaders.Columns[0].Width := 105;
  dbgUploaders.Columns[1].Width := 85;
  dbgUploaders.Columns[2].Width := 105;
  dbgUploaders.Columns[3].Width := 220;
  dbgUploaders.Columns[4].Width := 95;

  tblVideos.Open;
  dbgVideos.Columns[0].Width := 25;
  dbgVideos.Columns[1].Width := 220;
  dbgVideos.Columns[2].Width := 90;
  dbgVideos.Columns[3].Width := 65;
  dbgVideos.Columns[4].Width := 65;
  dbgVideos.Columns[5].Width := 65;
  dbgVideos.Columns[6].Width := 105;
  dbgVideos.Columns[7].Width := 305;
```

```
  ShowMessage('Database Restored!');
end;

procedure TfrmQuestion2.DBSetup;
begin
  // PROVIDED CODE - DO NOT MODIFY!
  con := TADOConnection.Create(Self);
  TDBConnection.DBConnect(con, Self);

  // ============TABLE 1=============//
  tblUploaders := TADOTable.Create(Self);
  tblUploaders.Connection := con;
  tblUploaders.TableName := 'tblUploaders';
  tblUploaders.Open;

  dbsUploaders := TDataSource.Create(Self);
  dbsUploaders.DataSet := tblUploaders;

  dbgUploaders.DataSource := dbsUploaders;

  dbgUploaders.Columns[0].Width := 105;
  dbgUploaders.Columns[1].Width := 85;
  dbgUploaders.Columns[2].Width := 105;
  dbgUploaders.Columns[3].Width := 220;
  dbgUploaders.Columns[4].Width := 95;

  // ===========TABLE 2============//
  tblVideos := TADOTable.Create(Self);
  tblVideos.Connection := con;
  tblVideos.TableName := 'tblVideos';
  tblVideos.Open;

  dbsVideos := TDataSource.Create(Self);
  dbsVideos.DataSet := tblVideos;

  dbgVideos.DataSource := dbsVideos;

  dbgVideos.Columns[0].Width := 25;
  dbgVideos.Columns[1].Width := 220;
  dbgVideos.Columns[2].Width := 90;
  dbgVideos.Columns[3].Width := 65;
  dbgVideos.Columns[4].Width := 65;
  dbgVideos.Columns[5].Width := 65;
  dbgVideos.Columns[6].Width := 105;
  dbgVideos.Columns[7].Width := 305;

  // ============QUERY===============//
  qry := TADOQuery.Create(Self);
  qry.Connection := con;

  dbsSQL := TDataSource.Create(Self);
  dbsSQL.DataSet := qry;
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  // PROVIDED CODE - DO NOT MODIFY!
  DBSetup;
end;

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

```
unit clsPlanets_u;

interface

type
  TPlanet = class
  private
    fPlanetName: string;
    fDistance: real;
    fGravity: real;

    procedure setGravity;
    function getDistance: real; // CAN THIS BE METHOD BE PROVIDED?
  public
    constructor create(sPlanetName: string; rDistance: real);
    procedure setDistance(rDistance: real);
    function getTime: integer;
    function getGravity: real;
    function toString: string;
  end;

implementation

uses strutils, sysutils, dateutils, math;

{ TPlanet }

{ TPlanet }

constructor TPlanet.create(sPlanetName: string; rDistance: real);
begin
  fPlanetName := sPlanetName;
  fDistance := rDistance;
  setGravity;
end;

function TPlanet.getDistance: real;
begin

end;

function TPlanet.getGravity: real;
begin
  Result := fGravity;
end;

function TPlanet.getTime: integer;
begin
  Result := ceil(fDistance / 300000);
end;

procedure TPlanet.setDistance(rDistance: real);
begin
  fDistance := rDistance;
end;

procedure TPlanet.setGravity;
```

```
begin
  if fPlanetName = 'Mercury' then
    fGravity := 3.7;
  if fPlanetName = 'Jupiter' then
    fGravity := 23.1;
  if fPlanetName = 'Venus' then
    fGravity := 8.9;
  if fPlanetName = 'Saturn' then
    fGravity := 9.0;
  if fPlanetName = 'Earth' then
    fGravity := 9.8;
  if fPlanetName = 'Uranus' then
    fGravity := 8.7;
  if fPlanetName = 'Mars' then
    fGravity := 3.7;
  if fPlanetName = 'Neptune' then
    fGravity := 11.0;

end;

function TPlanet.toString: string;
begin
  Result := 'Planet Name: ' + #9 + fPlanetName + #13 + 'Distance from Sun: ' +
    #9 + floattostr(fDistance) + ' km' + #13 + 'Gravity: ' + #9 +
    floattostr(fGravity) + ' m/s';

end;

end.
```

```
unit Question3_u;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,
  Vcl.Samples.Spin, Vcl.ComCtrls, math, clsPlanets_u;

type
  TfrmQuestion3 = class(TForm)
    grbQ3_2_1: TGroupBox;
    lblPlanetDistance: TLabel;
    sedDistance: TSpinEdit;
    btnQ3_2_1: TButton;
    grbOutput: TGroupBox;
    redOut: TRichEdit;
    grbMethods: TGroupBox;
    edtNewDistance: TEdit;
    btnQ3_2_2: TButton;
    btnQ3_2_4: TButton;
    lblNewRadius: TLabel;
    grbInformation: TGroupBox;
    btnQ3_2_5: TButton;
    btnQ3_2_3: TButton;
    rgpPlanet: TRadioGroup;
    edtWeight: TEdit;
    lblWeight: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_4Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnQ3_2_5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;
  objPlanet: TPlanet;
  arrName: array [1 .. 8] of string = (
    'Mercury',
    'Venus',
    'Earth',
    'Mars',
    'Jupiter',
    'Saturn',
    'Uranus',
    'Neptune'
  );
  arrDistance: array [1 .. 8] of real = (
    69770800,
    108940425,
    151681457,
```

```
    245599145,
    742405641,
    1462757650,
    2950720300,
    4474074840
  );
  arrWeight: array [1 .. 8] of real;

implementation

{$R *.dfm}

procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
  sPlanet: string;
  rDist: real;
begin
  sPlanet := rgpPlanet.Items[rgpPlanet.ItemIndex];
  rDist := sedDistance.Value;
  objPlanet := TPlanet.create(sPlanet, rDist);
end;

procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
begin
  redOut.Lines.Add(objPlanet.toString);
end;

procedure TfrmQuestion3.btnQ3_2_4Click(Sender: TObject);
begin
  // provided code
  redOut.Clear;
  // add your code here
  objPlanet.setDistance(StrToFloat(edtNewDistance.Text));
  redOut.Lines.Add(objPlanet.toString);
end;

procedure TfrmQuestion3.btnQ3_2_5Click(Sender: TObject);
var
  rGravity, rEarthWeight: real;
  rPlanetWeight: real;
  iCnt: integer;
const

  EARTHGRAVITY = 9.8;  // OR provide name and distance as Constants - can then be
used to create earth object.
begin
  rEarthWeight := StrToFloat(edtWeight.Text);
  // instantiate Earth object

  for iCnt := 1 to 8 do
  begin
    objPlanet := TPlanet.create(arrName[iCnt], arrDistance[iCnt]);
    rGravity := objPlanet.getGravity;
    rPlanetWeight := rEarthWeight / EARTHGRAVITY * objPlanet.getGravity;
    redOut.Lines.Add(arrName[iCnt] + #9 + FloatToStrF(rPlanetWeight,ffFixed,8,1) +
'kg');
  end;
```

```
end;

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
var
  iMinutes, iSeconds: integer;
begin
  iMinutes := objPlanet.getTime DIV 60;
  iSeconds := objPlanet.getTime MOD 60;
  redOut.Lines.Add('Time for light (mm:ss): ' + #9 + IntToStr(iMinutes) + ':' +
    IntToStr(iSeconds));

end;

procedure TfrmQuestion3.FormCreate(Sender: TObject);
begin
  redOut.Clear;
  redOut.Paragraph.TabCount := 1;
  redOut.Paragraph.tab[0] := 120;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_u;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,
  Vcl.ComCtrls;

type
  TfrmQuestion4 = class(TForm)
    pnlQ4_1: TPanel;
    btnQ4_1: TButton;
    redQ4: TRichEdit;
    btnQ4_2_1: TButton;
    pnlQ4_2: TPanel;
    Image1: TImage;
    btnQ4_2_2: TButton;
    procedure btnQ4_1Click(Sender: TObject);
    procedure btnQ4_2_1Click(Sender: TObject);
    procedure btnQ4_2_2Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    procedure Display(sTitle: string);
  end;

var
  frmQuestion4: TfrmQuestion4;
  arrAthletes: array [1 .. 8] of string;
  arrCountries: array [1 .. 8] of string;
  arrDistances: array [1 .. 8] of real;

  iFouls: integer = 0;

  // if read from text file fails use these

  // arrAuxAthletes: array [1 .. 8] of string = (

  // 'Nairn LaQuan',
  // 'Sreeshankar Murali',
  // 'Van Vuuren Jovan',
  // 'Thompson Shawn-D',
  // 'Yahiya Muhammed',
  // 'Frayne Henry',
  // 'James Tristan',
  // 'Otuonye Ifeanyichukwu'
  // );

  // arrAuxCountries: array [1 .. 8] of string = (
  // 'Bahamas',
```

```
  // 'India',
  // 'South Africa',
  // 'Jamaica',
  // 'India',
  // 'Australia',
  // 'Dominica',
  // 'Turks and Caicos Islands');

  // arrAuxDistances: array [1 .. 8] of real = (
  // 8.09,
  // 8.08,
  // 8.06,
  // 8.05,
  // 7.97,
  // 7.94,
  // 7.85
  // 7.80);

  // 2_D array containing the final 6 jump results of each finalist :

  arrJumps: array [1 .. 8, 1 .. 6] of real = ((7.94, 8.09, 0, 0, 7.84, 7.98),
    (7.60, 7.84, 7.84, 0, 8.08, 0), (7.92, 8.06, 7.83, 7.49, 7.75, 0),
    (7.62, 0, 8.05, 7.75, 7.73, 7.70), (0, 7.65, 7.72, 7.74, 7.58, 7.97),
    (7.89, 0, 0, 0, 7.94, 0), (7.85, 7.79, 0, 0, 7.80, 7.69),
    (7.53, 7.80, 0, 7.65, 7.52, 7.67));

implementation

{$R *.dfm}

procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  tfTextfile: Textfile;
  sOneLine: string;
  ipos: integer;
  icounter: integer;

begin

  // Answer Question 4.1 here
  icounter := 0;
  AssignFile(tfTextfile, 'Longjump.txt');
  try
    Reset(tfTextfile);
  except
    showmessage('File cannot be accessed.');
    exit;
  end;

  while (not EOF(tfTextfile)) do
  begin
    inc(icounter);
    Readln(tfTextfile, sOneLine);
    ipos := pos('#', sOneLine);
    arrAthletes[icounter] := Copy(sOneLine, 1, ipos - 1);
    delete(sOneLine, 1, ipos);
```

```
    ipos := pos('#', sOneLine);
    arrCountries[icounter] := Copy(sOneLine, 1, ipos - 1);
    delete(sOneLine, 1, ipos);

    ipos := pos('#', sOneLine);
    arrDistances[icounter] := strtoFloat(sOneLine);

  end;
  Closefile(tfTextfile);

  // Start of provided code, DO NOT DELETE
  Display('MEN LONG JUMP FINAL RESULTS');
  // End of provided code, DO NOT DELETE

end;


procedure TfrmQuestion4.btnQ4_2_1Click(Sender: TObject);
var
  i, j, iJumps: integer;
  sOUt: string;
  rAvgJumps: real;

begin

  // Start of provided code, DO NOT DELETE
  redQ4.Clear;
  redQ4.Paragraph.TabCount := 8;
  redQ4.Paragraph.Tab[0] := 100;
  redQ4.Paragraph.Tab[1] := 130;
  redQ4.Paragraph.Tab[2] := 160;
  redQ4.Paragraph.Tab[3] := 190;
  redQ4.Paragraph.Tab[4] := 220;
  redQ4.Paragraph.Tab[5] := 250;
  redQ4.Paragraph.Tab[6] := 280;
  redQ4.Paragraph.Tab[7] := 310;
  redQ4.Lines.Add
    ('MEN LONG JUMP FINAL 6 JUMPS WITH AVERAGES AND BEST DISTANCE' + #13);
  // End of provided code, DO NOT DELETE

  redQ4.Lines.Add('Athlete' + #9 + '#1' + #9 + '#2' + #9 + '#3' + #9 + '#4' + #9 +
    '#5' + #9 + '#6' + #9 + 'Avg' + #9 + 'Best distance');
  for i := 1 to 8 do
  begin
    sOUt := arrAthletes[i];
    iJumps := 0;
    rAvgJumps := 0;
    for j := 1 to 6 do
    begin
      if arrJumps[i, j] > 0 then
      begin
        rAvgJumps := rAvgJumps + arrJumps[i, j];
        inc(iJumps);
      end
      else
        inc(iFouls);
      sOUt := sOUt + #9 + FloatToStrF(arrJumps[i, j], ffFixed, 8, 2);
```

```
    end;
    rAvgJumps := rAvgJumps / iJumps;
    redQ4.Lines.Add(sOUt + #9 + FloatToStrF(rAvgJumps, ffFixed, 8, 2) + #9 +
      FloatToStrF(arrDistances[i], ffFixed, 8, 2));
  end;
end;

procedure TfrmQuestion4.btnQ4_2_2Click(Sender: TObject);
begin
  showmessage('Total number of fouls = ' + IntToStr(iFouls));
end;

// Start of provided code, DO NOT DELETE
procedure TfrmQuestion4.Display(sTitle: string);
var
  i: integer;
begin
  redQ4.Clear;
  redQ4.Paragraph.TabCount := 2;
  redQ4.Paragraph.Tab[0] := 120;
  redQ4.Paragraph.Tab[1] := 240;
  redQ4.Lines.Add(sTitle + #13);
  redQ4.Lines.Add('Athletes' + #9 + 'Countries' + #9 + 'Distances');
  for i := 1 to length(arrAthletes) do
  begin
    redQ4.Lines.Add(arrAthletes[i] + #9 + arrCountries[i] + #9 +
      FloatToStrF(arrDistances[i], ffFixed, 8, 2));
  end;
end;
// End of provided code, DO NOT DELETE

end.
```