basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2024

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 33 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 15) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 16 to 33) contain examples of solutions for Questions 1 to 4 in programming code.

- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 15) 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**
1.1.1	**Button [1.1.1 - Random]** Generate random number ✓ from 5 to 10 (both values included in range) ✓ `edtQ1_1_1.Text :=` ✓ convert number to String ✓	**4**	

1.1.2	**Button [1.1.2 - Round up]**
	
	`spnQ1_1_2.Value :=` ✓
	Round up ✓
	`Ceil (NUMBER)`
	`Round (NUMBER + 0.5)`
	`Floor (NUMBER) + 1`
	`Trunc (NUMBER) + 1`
	With constant as a parameter ✓
		3	

1.2	**Button [1.2 - Surface area]**
	
	Extract the height and radius✓ from the edit boxes
	Converted to Float ✓
	Calculate the Area:
	A := PI * rR * ✓ (rR + ✓ sqrt (✓ sqr(rR) + sqr(rH) ✓))
	
	Display the Area in the label ✓
	formatted to 2 decimal places ✓
	`FormatFloat ('0.00', A)`
	`FloatToStrF (A, ffFixed, 10, 2)`
	`Format ('%.2f', [A])`
	
	Also ACCEPT `Power(rR,2)` instead of `Sqr`.
	
	NOTE: Brackets must be added correctly in the calculation.
		8	

1.3	**Button [1.3 - Read file]** Declare variables for the address and bedrooms ✓ `AssignFile(tFile, 'Houses.txt')` ✓ `Reset(tFile)` ✓ Loop through the text file with correct condition ✓ Readln ✓ (tFile, address variable ✓) Readln (tFile, bedroom variable) ✓ Concatenate the address and bedroom with a dash (-) between the address and bedroom ✓ Display the output in the rich edit ✓ **Also ACCEPT** alternative to read from file: If odd line number, then store in address variable (2) If even line number, then store in bedroom variable (1)	**9**	

1.4	**Button [1.4 - Add name]**
	
	Extract the name from the combo box ✓
	
	Test if ✓
	the check box is checked ✓
	Add name to the paid rich edit component ✓
	Else ✓
	Add name to the not paid rich edit component ✓
	
	Remove the selected name from the combo box ✓
	`cmbQ1_4.DeleteSelected`
	`cmbQ1_4.Items.Delete (cmbQ1_4.ItemIndex)`
	
	NOTE:
	NO mark for changing the content of the combo box item to be an empty string and not removing it.
		7	

1.5	**Button [1.5 - Replace]** Loop through the string ✓ Test if character is NOT a space ✓ Add character to password variable in reverse ✓ Display password in memo ✓ Loop Index from 1 to length of password ✓ If Index MOD 3 is 0 ✓ Generate random value from 1 to 6 ✓ Replace password character at Index with random character from sCharacters ✓ Display updated password in memo ✓ **Alternative for the first 3 marks:** Loop to remove spaces first (1), then loop (1) through the modified string in reverse (1) **In the password part, also ACCEPT:** • Loop Index from 3 to length of password • Randomly generate value from 1 to Length(sCharacters)	**9**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – DATABASE PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**		
2.1.1	**Button [2.1.1 - Low population]** `SELECT *` ✓ `FROM tblLocations` ✓ `WHERE Population < 200000` ✓	**3**	
2.1.2	**Button [2.1.2 - Runners United September runs]** `SELECT MarathonID, MarathonDate, Distance` `FROM tblMarathons` ✓ `WHERE Organiser = "Runners United"` ✓ AND ✓ `Month(MarathonDate) = 9` ✓ **Also ACCEPT:** • `Organiser Like "%Runners United%"` • `MarathonDate Like "%/09/%"` • `Mid (Marathondate,6,2) = 9`	**4**	
2.1.3	**Button [2.1.3 - Marathon locations]** `SELECT City & " - "` ✓ `& LEFT(Province,3)` ✓ `AS Location` ✓ `FROM tblLocations` ✓ **Also ACCEPT:** • + instead of & • `Mid(Province,1,3)`	**4**	
2.1.4	**Button [2.1.4 - Add city]** `INSERT INTO tblLocations` ✓ `VALUES` ✓ `(19,"Welkom","Free State",1198,423016)` ✓✓ (correct order (1 mark), correct number of parameters (1 mark)	**4**	

2.1.5	**Button [2.1.5 - City details]** SELECT City, ✓ COUNT(City) ✓ AS NumMarathons, SUM(Prizemoney) AS [Total Prize Money] ✓ FROM tblMarathons , tblLocations ✓ WHERE tblMarathons.LocationID = tblLocations.LocationID ✓ GROUP BY City ✓ HAVING SUM(Prizemoney) > 50000 ✓ **Also ACCEPT:** • Count(*) • Count(A field name) / Count(tblLocations.LocationID)	**7**	
	Subtotal:	**22**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**Database Manipulation**		
2.2.1	**Button [2.2.1 - Remove marathons]** Go to the first record in tblMarathons ✓ Loop through tblMarathons ✓ Test if tblMarathons['Organiser'] = sOrganiser ✓ tblMarathons.Delete ✓ else tblMarathons.Next ✓ End loop	**5**	
2.2.2	**Button [2.2.2 - Qualifying events]** Initialise flag / counter ✓ tblLocations.First (mark with tblLocations.Next)** ✓ Loop through tblLocations ✓ Test if tblLocations['City'] = sCity ✓ Change flag / increment counter ✓ tblMarathons.First (mark with tblMarathons.Next)** ✓ Loop through tblMarathons ✓ Test if (tblMarathons['LocationID'] = tblLocations['LocationID']) ✓ AND ✓ (tblMarathons['Distance'] >= 40) ✓ Display the MarathonName and Distance converted to string ✓ tblMarathons.Next End loop (tblMarathons) tblLocations.Next End loop (tblLocations) Test flag / counter ✓ Display message indicating that the city is not found ✓ **NOTE: **** • The FIRST and NEXT statements for the outer loop (for tblLocations) must both be in the correct position for one mark. • The FIRST and NEXT statements for the inner loop (for tblMarathons) must both be in the correct position for one mark.	**13**	
	Subtotal:	**18**	
	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**Constructor Create** Set the attributes fMarathonName and fRecordHolder to correct parameters ✓ Set fRecordDate and fRecordTime to correct parameters ✓ Set fDistance to correct parameter ✓	**3**	
3.1.2	**Function getRecordTime** Function heading with String value as return data type ✓ Return fRecordTime ✓	**2**	
3.1.3	**Function checkRecord** Function with Boolean return datatype ✓ with string parameter ✓ Test if parameter value < fRecordTime ✓ Return true ✓ Else Return false ✓ **Also ACCEPT:** • StrToTime and toMinutes to compare the times • Test if fRecordTime > parameter value	**5**	
3.1.4	**Function calcPace** Function heading with real return data type ✓ Return ✓ toMinutes(fRecordTime) ✓ / fDistance ✓	**4**	
3.1.5	**Function toString** Function heading with string value as return data type ✓ Build a string with labels: dash, 'km:', 'on' and brackets ✓ Distance converted to string ✓ Contains all attributes✓ Return the string ✓	**5**	
	Subtotal: Object class	**19**	

QUESTION 3: MARKING GRID (CONTINUED)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 - Instantiate Object]** Distance extracted from radio group: Items at ItemIndex ✓ Extract the distance using String manipulation ✓ converted to real ✓ objMRecord := ✓ TMRecord ✓ .create ✓ (marathon name, record holder, record date, record time, distance) ✓ Display object in redQ3 using toString method ✓ **Alternative for extracting the distance:** Use if else / case (1) and assign distance (1)	**8**	
3.2.2	**Button [3.2.2 - Pace]** Call the calcPace method ✓ Display the result in redQ3 converted to a string ✓ Formatted to 3 decimal places ✓ with correct label (min/km) ✓	**4**	
3.2.3	**Button [3.2.3 - Check record]** Extract name and time of runner from edit boxes ✓ If objMRecord.checkRecord ✓ (Time of runner) ✓ Call setRecordHolder (Name of runner) ✓ Call setRecordTime (Time of runner) ✓ Call setRecordDate ✓ (Current date as string) ✓ Display toString method ✓ Else Display current record time in redQ3 by calling the getRecordTime method ✓	**9**	
	Subtotal Form class:	**21**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM SOLVING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX MARKS**	**LEARNER'S MARKS**
4.1	**Button [4.1 - Count marathons]** Outer loop from 1 to 9 ✓ Initialise counter ✓ Inner loop (out + 1 to 10) ✓ Test array[out] ✓ = array[inner] ✓ and array[inner] <> symbol ✓ increment counter ✓ replace array[inner] with a symbol such as ('*', ' ') ✓ if array[out] is NOT symbol used ✓ Display array[out] ✓ and counter converted to string ✓ **Concepts 1: (Replace duplicates when counting)** Loop outer through array (1) Initialise counter (1) Loop inner starting at outer loop counter + 1 (1) Test array[outer] (1) = array[inner] (1) AND array[inner] <> Symbol used (1) Increase counter (1) Replace array[inner] with space or * or other symbol (1) if array[outer] is NOT symbol used (1) Display array[outer] (1) and counter converted to string (1)	**11**	

Concepts 2: (Create list (array/string) without duplicates)

Build string / populate array with unique values [4]
Initialisation of counter
and bFound (1)
Outer and inner loops (1)
If statements used in testing (1)
Assignment statements (1)

Loop though unique values in Temp array and count from arrMarathons array [5]

Loop x through temporary array with unique values / string (1)
Initialise iNumMarathons (1)
Loop y through arrMarathons (1)
Test if arrTemp[x] equal to arrMarathons[y] (1)
Increment iNumMarathons (1)

Display [2]

Display marathon name (1) and counter (1)

Concepts 3: (Temp arrays Unique marathon names and counting values)

Use `arrTempMarathons` and `arrCountMarathons`

Initialise counter ** (1) `with found := false`
Loop outer through array (1)
Set bFound to FALSE **
Loop inside starting at 1 to counter (1)
Test arrMarathons[outer] =
arrTempMarathons[inside] (1)
Change bFound to TRUE (1)
Increase arrCountMarathons[inside] (1)
If bFound is FALSE (1)
Increase counter (1)
Set arrCountMarathons[counter] to 1 (1) ##
Set arrTempMarathons[counter] to
arrMarathons[outside] ##
Loop from 1 to counter (1)
Display arrTempMarathons and arrCountMarathons (converted to integer) (1)

4.2	**Combobox [cmbQ4_2]** *Extract word from combo box **[1]*** Read word from combobox ✓ *Test if word is in the row **[10]*** Loop rows 1 to 14 ✓ Initialise newString ✓ Loop columns 1 to 14 ✓ build newString✓ using characters from array[R,C] ✓ if newString = selected word ✓ Calculate/Set start column index ✓ Calculate end column index: Start column index ✓ + length of word ✓- 1 ✓ *Display the row number, start and end column **[3]*** Display row number ✓ Start column ✓ and End column ✓ *Change to uppercase **[4]*** Loop through word in 2D array start column index ✓ end column index ✓ Change char in 2D ✓ to upcase char ✓ *Display 2D array **[1]*** Call display method ✓	**19**	

	Concept 1: (Row and Column loops – copy from row) *Extract word from combo box **[1]*** Read word from combo box (1) *Test if word is in the row **[10]*** Loop iR from 1 to 14 (1) Loop iC from 1 to 14 (1) Test if marathon name is in row iR starting at column index iC (4) Calculate/Set start column index of word (1) Calculate end column index of word (3) *Display the row number, start and end cloumn **[3]*** Display row number of word (1) Start column (1) and End column (1) *Change to uppercase **[4]*** Loop from start column (1) to end column (1) Change character (1) to upcase character (1) *Display 2D array **[1]*** Call the display method **(1)** **Concept 2: (Row – using pos directly)** *Extract word from combo box **[1]*** Read word from combo box (1) *Test if word is in the row **[10]*** Row Loop from 1 to 14 (1) Test if marathon name is in the row (4) Calculate start column index of word (2) Calculate end column index of word (3) *Display the row number, start and end column **[3]*** Display row number of word (1) Start column (1) End column (1) *Change to uppercase **[4]*** Loop from start column (1) to end column (1) Change character (1) to upcase character (1) *Display 2D array **[1]*** Call the display method **(1)**		
	TOTAL SECTION D:	**30**	

SUMMARY OF LEARNER'S MARKS:

CENTER 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;

interface

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

type
  TfrmQuestion1 = class(TForm)
    grpQ1_1: TGroupBox;
    grpQ1_2: TGroupBox;
    grpQ1_3: TGroupBox;
    grpQ1_4: TGroupBox;
    grpQ1_5: TGroupBox;
    btnQ1_1_1: TButton;
    btnQ1_1_2: TButton;
    edtQ1_1_1: TEdit;
    spnQ1_1_2: TSpinEdit;
    Label1: TLabel;
    Label2: TLabel;
    edtQ1_2_r: TEdit;
    edtQ1_2_h: TEdit;
    btnQ1_2: TButton;
    btnQ1_3: TButton;
    redQ1_3: TRichEdit;
    Label3: TLabel;
    chbQ1_4: TCheckBox;
    btnQ1_4: TButton;
    Label4: TLabel;
    Label5: TLabel;
    redQ1_4_P: TRichEdit;
    redQ1_4_NP: TRichEdit;
    btnQ1_5: TButton;
    cmbQ1_4: TComboBox;
    edtQ1_5: TEdit;
    lblQ1_2: TLabel;
    Image1: TImage;
    memQ1_5: TMemo;
    procedure btnQ1_1_1Click(Sender: TObject);
    procedure btnQ1_1_2Click(Sender: TObject);
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure btnQ1_5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion1: TfrmQuestion1;
implementation
```

```
{$R *.dfm}

// ============================================================================
// 1.1.1 Random                                                      4 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_1_1Click(Sender: TObject);
begin
  edtQ1_1_1.Text := intToStr(randomRange(5, 11));
end;

// ============================================================================
// 1.1.2 Random                                                      3 marks
// ============================================================================

procedure TfrmQuestion1.btnQ1_1_2Click(Sender: TObject);
const
  NUMBER = 5.63247;
begin

  spnQ1_1_2.Value := ceil(NUMBER);

end;

// ============================================================================
// 1.2 Surface area                                                  8 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  rA, rH, rR: real;

begin
  rH := StrToFloat(edtQ1_2_h.Text);
  rR := StrToFloat(edtQ1_2_r.Text);
  rA := PI * rR * (rR + Sqrt(Sqr(rH) + Sqr(rR)));
  lblQ1_2.Caption := FloatToStrF(rA, ffFixed, 8, 2);
end;

// ============================================================================
// 1.3 Read file                                                     9 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  tFile: textfile;
  sAdd, sRooms: String;
begin

  AssignFile(tFile, 'Houses.txt');
  Reset(tFile);

  while NOT Eof(tFile) do
  begin
    readln(tFile, sAdd);
    readln(tFile, sRooms);
    redQ1_3.Lines.Add(sAdd + ' - ' + sRooms);
  end;
  CloseFile(tFile);
end;
```

```
// ============================================================================
// 1.4 Add name                                                      7 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sName: String;
begin
  sName := cmbQ1_4.Text;
  if chbQ1_4.Checked then
  begin
    redQ1_4_P.Lines.Add(sName);
  end
  else
  begin
    redQ1_4_NP.Lines.Add(sName);
  end;

  cmbQ1_4.items.Delete(cmbQ1_4.ItemIndex);
end;
// ============================================================================
// 1.5 Replace                                                       9 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  sNameSurname, sCharacters, sPassword: String;
  iCnt, iLen, iRandom: integer;
begin
  // Provided code
  sNameSurname := edtQ1_5.Text;
  sCharacters := '@#$%^&';
  // Add your code here
  sPassword := '';
  iLen := Length(sNameSurname);
  for iCnt := iLen downto 1 do
  begin
    if sNameSurname[iCnt] <> ' ' then
        sPassword := sPassword + sNameSurname[iCnt];
  end;
  memQ1_5.Lines.Add(sPassword);
  for iCnt := 1 to Length(sPassword) do
  begin
    if(iCnt mod 3 = 0) then
    begin
      iRandom := random(6) + 1;
      sPassword[iCnt] := sCharacters[iRandom];
    end;
  end;
  memQ1_5.Lines.Add(sPassword);
end;
end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
// ============================================================================
// 2.1 - Section: SQL statements
// ============================================================================

// ============================================================================
//  2.1.1 Low population                                              3 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
  // Question 2.1.1

  sSQL1 := 'SELECT * ' + 'FROM tblLocations ' + 'WHERE Population <
200000';

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

// ============================================================================
//  2.1.2 Runners United September runs                               4 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
begin
  // Question 2.1.2
  sSQL2 := 'SELECT MarathonID, MarathonDate, Distance ' +
    'FROM tblMarathons ' + 'WHERE Organiser = "Runners United" AND ' +
    'Month(MarathonDate) = 9';

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

// ============================================================================
//  2.1.3 Marathon locations                                          4 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3: String;
begin
  // Question 2.1.3

  sSQL3 := 'SELECT City & " - " & left(Province,3) AS [Location] ' +
    'FROM tblLocations ';

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

```
// ============================================================================
//  2.1.4 Add city                                                    4 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
  bValid: boolean;
begin
  // Question 2.1.4

   sSql4 := 'INSERT INTO tblLocations VALUES
                   (19,"Welkom","Free State",1198,423016)';

// Provided code - do not change
  dbCONN.ExecuteSQL(sSQL4);

end;

// ============================================================================
//  2.1.5 High prize money                                            7 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5: String;
  bChanged: boolean;
begin
  // Question 2.1.5

  sSQL5 :=
    'SELECT City, COUNT(City) AS [NumMarathons],
     SUM(Prizemoney) AS [Total Prize Money] '
    + 'FROM tblMarathons , tblLocations ' +
    'WHERE tblMarathons.LocationID = tblLocations.LocationID ' +
    'GROUP BY City HAVING SUM(Prizemoney) > 50000';

  // Provided code - do not change
  dbCONN.runSQL(sSQL5);
end;
```

```
// ============================================================================
// 2.2 - Section: Delphi code
// ============================================================================

// ============================================================================
// 2.2.1 Remove marathons                                              5 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  sOrganiser: String;
begin
  // Provided code
  sOrganiser := InputBox('Organiser',
    'Enter the name of the organiser to remove', 'Endurance Sports SA');

  // Question 2.2.1
  tblMarathons.First;
  while NOT tblMarathons.Eof do
  begin
    if tblMarathons['Organiser'] = sOrganiser then
      tblMarathons.Delete
    else
      tblMarathons.Next;
  end;
end;

// ============================================================================
// 2.2.2 Qualifying events                                            13 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  sCity: String;
  bFound: boolean;
  iLocation: Integer;
begin
  // Provided code
  sCity := InputBox('City', 'Enter the name of the city', 'Paarl');
  redQ2_2_2.Clear();
  // Question 2.2.2
  bFound := False;
  tblLocations.First;
  while (NOT tblLocations.Eof) AND (bFound = False) do
  begin
    if tblLocations['City'] = sCity then
    begin
      bFound := True;
      iLocation := tblLocations['LocationID'];
    end;
    tblLocations.Next;
  end;

if bFound then
  begin
    tblMarathons.First;
    while NOT tblMarathons.Eof do
```

```
    begin
      if (tblMarathons['LocationID'] = iLocation) AND
        (tblMarathons['Distance'] >= 40) then
      begin
        redQ2_2_2.Lines.Add(tblMarathons['MarathonName'] + #9 +
          FloatToStr(tblMarathons['Distance']));
      end;
      tblMarathons.Next;
    end
  end
   else
     redQ2_2_2.Lines.Add(sCity + ' is not found.');

  // Alternative:

  { bFound := False;
    tblLocations.First;
    while NOT tblLocations.Eof do
    begin
      if tblLocations['City'] = sCity then
       begin
         bFound := True;
         tblMarathons.First;
         while NOT tblMarathons.Eof do
          begin
           if (tblMarathons['LocationID'] = tblLocations['LocationID'])
                AND (tblMarathons['Distance'] >= 40) then
             begin
               redQ2_2_3.Lines.Add(tblMarathons['MarathonName'] + #9 +
                  FloatToStr(tblMarathons['Distance']));
             end;
             tblMarathons.Next;
         end;
    end;
    tblLocations.Next;
  end;

    if bFound = False then
      redQ2_2_3.Lines.Add(sCity + ' is not found.'}
end;

// ================================================================
// {$REGION 'Provided code: Setup DB connections - DO NOT CHANGE!'}
// ================================================================
procedure TfrmQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  // Restores the Database
  dbCONN.RestoreDatabase;
  redQ2_2_2.Clear;
  dbCONN.SetupGrids(dbgLocations, dbgMarathons, dbgrdSQL);
end;

procedure TfrmQuestion2.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  // Disconnects from database and closes all open connections
  dbCONN.dbDisconnect;
```

```
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  // Provided code
  redQ2_2_2.Paragraph.TabCount := 2;
  redQ2_2_2.Paragraph.Tab[0] := 150;
  redQ2_2_2.Paragraph.Tab[1] := 300;
end;

procedure TfrmQuestion2.FormShow(Sender: TObject);
begin
  // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblLocations := dbCONN.tblOne;
  tblMarathons := dbCONN.tblMany;
  dbCONN.SetupGrids(dbgLocations, dbgMarathons, dbgrdSQL);
  pgcDBAdmin.ActivePageIndex := 0;
end;

// ================================================================
// {$ENDREGION}
// ================================================================

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
unit MRecord_U;

interface

type
  TMRecord = class(TObject)
  private
  var
    fMarathonName: String;
    fRecordHolder: String;
    fRecordDate: String;
    fRecordTime: String;
    fDistance: real;
  public
    // Provide code
    constructor create(sMarathonName, sRecordHolder, sRecordDate,
      sRecordTime: String; rDistance: real);
    procedure setRecordHolder(sName: String);
    procedure setRecordTime(sNewRecord: String);
    procedure setRecordDate(sNewDate: String);
    function toMinutes(sTime: String): real;

    function getRecordTime: String;
    function checkRecord(sRecordTime: String): boolean;
    function calcPace: real;
    function toString: String;
  end;

implementation

uses
  SysUtils, Math;

{ TMRecord }

// ============================================================================
// 3.1.1 Constructor Create                                          3 marks
// ============================================================================

constructor TMRecord.create(sMarathonName, sRecordHolder, sRecordDate,
  sRecordTime: String; rDistance: real);
begin
  fMarathonName := sMarathonName;
  fRecordHolder := sRecordHolder;
  fDistance := rDistance;
  fRecordDate := sRecordDate;
  fRecordTime := sRecordTime;
end;
```

```
// ============================================================================
// 3.1.2 Function getRecordTime                                        2 marks
// ============================================================================

function TMRecord.getRecordTime: String;
begin
  Result := fRecordTime;
end;

// ============================================================================
// 3.1.3 Function checkRecord                                          5 marks
// ============================================================================

function TMRecord.checkRecord(sRecordTime: String): boolean;
begin
  Result := sRecordTime < fRecordTime;
  { Alternative:
    if sRecordTime < fRecordTime then
     Result := True
    Else
     Result := False;
    }
end;

// ============================================================================
// 3.1.4 Function calcPace                                             4 marks
// ============================================================================

function TMRecord.calcPace: real;
begin
  Result := toMinutes(fRecordTime) / fDistance;
end;
// ============================================================================
// 3.1.5 Function toString                                             5 marks
// ============================================================================

function TMRecord.toString: String;
begin
  Result := fMarathonName + ' - ' + FloatToStr(fDistance)
            + ' km: ' + fRecordHolder + ' (' + fRecordTime + ' on ' +
            fRecordDate + ')';
end;

// ============================================================================
// Provided code
// ============================================================================
procedure TMRecord.setRecordHolder(sName: String);
begin
  fRecordHolder := sName;
end;

procedure TMRecord.setRecordTime(sNewRecord: String);
begin
  fRecordTime := sNewRecord;
end;
```

```
procedure TMRecord.setRecordDate(sNewDate: String);
begin
  fRecordDate := sNewDate;
end;
function TMRecord.toMinutes(sTime: String): real;
var
  rMin: real;
begin
  rMin := StrTofloat(copy(sTime, 4, 2));
  rMin := rMin + StrToFloat(copy(sTime, 1, 2)) * 60;
  rMin := rMin + StrToFloat(copy(sTime, 7, 2)) / 60;
  Result := rMin;
end;

end.
```

Main Form Unit

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    gbxQ3_2_1: TGroupBox;
    gbxQ3_2_2: TGroupBox;
    redQ3: TRichEdit;
    btnQ3_2_1: TButton;
    gbxQ3_2_3: TGroupBox;
    btnQ3_2_3: TButton;
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ3_2_2: TButton;
    Label6: TLabel;
    edtQ3_2_1_Marathon: TEdit;
    Label2: TLabel;
    Label1: TLabel;
    edtQ3_2_1_RecordHolder: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    edtQ3_2_1_RecordTime: TEdit;
    rgpQ3_2_1: TRadioGroup;
    Label7: TLabel;
    edtQ3_2_3_Name: TEdit;
    edtQ3_2_3_Time: TEdit;
    Image1: TImage;
    edtQ3_2_1_RecordDate: TEdit;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
  private
  public
  end;
var
  frmQuestion3: TfrmQuestion3;
implementation
{$R *.dfm}
uses
  MRecord_U;

var
  objMRecord: TMRecord;
```

```
// ============================================================================
// 3.2.1 Instantiate object                                          8 marks
// ============================================================================

procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
  sMarathonName, sRecordHolder, sRecordDate, sRecordTime: String;
  rDistance: real;
  sDistance: String;
begin
  // Provided code
  redQ3.Clear;
  sMarathonName := edtQ3_2_1_Marathon.Text;
  sRecordHolder := edtQ3_2_1_RecordHolder.Text;
  sRecordDate := edtQ3_2_1_RecordDate.Text;
  sRecordTime := edtQ3_2_1_RecordTime.Text;

  // Question 3.2.1

  sDistance := rgpQ3_2_1.Items[rgpQ3_2_1.ItemIndex];
  rDistance := StrToFloat(Copy(sDistance, 1, Pos(' ', sDistance) - 1));
  objMRecord := TMRecord.create(sMarathonName, sRecordHolder,
                sRecordDate, sRecordTime, rDistance);
  redQ3.lines.Add(objMRecord.toString);
end;

// ============================================================================
// 3.2.2 Pace                                                        4 marks
// ============================================================================

procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
begin
// Provided code
redQ3.Clear;

// Question 3.2.2
redQ3.lines.Add('Record holder''s pace:'+
                FloatToStrF(objMRecord.calcPace, ffFixed, 8, 3) + '
                min/km');
end;

// ============================================================================
// 3.2.3 Check record                                                9 marks
// ============================================================================

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
var
  sName, sTime: String;
begin
  // Provided code
  redQ3.Clear;

  // Question 3.2.3
  sName := edtQ3_2_3_Name.Text;
  sTime := edtQ3_2_3_Time.Text;
```

```
if objMRecord.checkRecord(sTime) then
  begin
    objMRecord.setRecordHolderName(sName);
    objMRecord.setRecordTime(sTime);
    objMRecord.setRecordDate(DateToStr(Date()));
    redQ3.lines.Add(objMRecord.toString);
  end
  Else
  begin
    redQ3.lines.Add
      ('The current record remains: ' + objMRecord.getRecordTime);
  end;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_u;

interface

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

type
  TfrmQ4_1 = class(TForm)
    pgcQ4: TPageControl;
    tshQ4_1: TTabSheet;
    tshQ4_2: TTabSheet;
    redQ4_1: TRichEdit;
    btnQ4_1: TButton;
    pnlQ4_1Heading: TPanel;
    redQ4_2: TRichEdit;
    cmbQ4_2: TComboBox;
    pnlQ4_2Heading: TPanel;
    btnReset: TBitBtn;
    memQ4_2: TMemo;
    imgQ4_1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    procedure btnQ4_1Click(Sender: TObject);
    procedure display2D;
    procedure FormCreate(Sender: TObject);
    procedure cmbQ4_2Change(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQ4_1: TfrmQ4_1;
  iCheck: Integer = 0;
  arrMarathons: array [1 .. 10] of String = (
    'Wally Hayward Marathon',
    'Sasol Marathon',
    'Soweto Marathon',
    'Jacaranda City Marathon',
    'Sasol Marathon',
    'Durban City Marathon',
    'Soweto Marathon',
    'Soweto Marathon',
    'Wally Hayward Marathon',
    'Soweto Marathon'
  );
```

```
arrChar: array [1 .. 14, 1 .. 14] of char =
   (('u', 'x', 'v', 'm', 's', 'a', 's', 'o', 'l', 'f', 'k', 'j', 't', 'r'),
   ('u', 'm', 'g', 'e', 'n', 'i', 'w', 'a', 't', 'e', 'r', 'd', 's', 'e'),
   ('g', 'v', 'o', 'e', 't', 'v', 'a', 'n', 'a', 'f', 'r', 'i', 'k', 'a'),
   ('e', 'p', 'o', 'y', 'i', 'l', 'c', 'k', 'h', 'j', 's', 'd', 'f', 'd'),
   ('n', 'k', 'n', 'y', 's', 'n', 'a', 'f', 'o', 'r', 'e', 's', 't', 'u'),
   ('i', 's', 'y', 'd', 'b', 'c', 'r', 'g', 'h', 'k', 'c', 's', 'a', 'r'),
   ('w', 'a', 'l', 'l', 'y', 'h', 'a', 'y', 'w', 'a', 'r', 'd', 's', 'b'),
   ('a', 's', 'q', 'r', 't', 'n', 'n', 'j', 'h', 'e', 'r', 't', 'h', 'a'),
   ('t', 'o', 'e', 'r', 'y', 'b', 'd', 'r', 'h', 'k', 'l', 'g', 'd', 'n'),
   ('e', 'j', 'a', 'c', 'a', 'r', 'a', 'n', 'd', 'a', 'c', 'i', 't', 'y'),
   ('r', 'y', 'j', 'f', 'g', 'f', 'c', 'f', 'g', 'u', 'h', 'v', 'c', 'i'),
   ('k', 'h', 'h', 'l', 'p', 'h', 'i', 'l', 'l', 'c', 'r', 'e', 's', 't'),
   ('a', 'd', 'e', 'v', 'd', 's', 'o', 'w', 'e', 't', 'o', 'm', 'k', 'y'),
   ('p', 'd', 'u', 'r', 'b', 'a', 'n', 'c', 'i', 't', 'y', 'z', 'c', 'l'));

implementation

{$R *.dfm}

// ========================================================================
// 4.1 Count marathons                                          11 marks
// ========================================================================

procedure TfrmQ4_1.btnQ4_1Click(Sender: TObject);
var
  iOut, iIn, iNumMarathons: Integer;
begin
  for iOut := 1 to 10 do
  begin
    iNumMarathons := 1;
    for iIn := iOut + 1 to 10 do
    begin
      if (arrMarathons[iOut] = arrMarathons[iIn]) and
                                  (arrMarathons[iIn] <> '') then
      begin
        inc(iNumMarathons);
        arrMarathons[iIn] := '';
      end;
    end;
    if arrMarathons[iOut] <> '' then
      redQ4_1.Lines.Add(arrMarathons[iOut] + #9 +
                                  IntToStr(iNumMarathons));
  end;
end;
```

```
// ============================================================================
// 4.2 Find hidden marathons                                          19 marks
// ============================================================================

procedure TfrmQ4_1.cmbQ4_2Change(Sender: TObject);
var
  iOut, iIn, iLen, iR, iC, iC2, iPos: Integer;
  sWord, sNewWord1, sNewWord2: String;
  bFound: boolean;
begin
  bFound := false;
  sWord := cmbQ4_2.Text;
  iLen := Length(sWord);
  for iOut := 1 to 14 do
  begin
    for iIn := 1 to 14 do
    begin
      iR := iOut;
      iC := iIn;
      if sWord[1] = arrChar[iOut, iIn] then
      begin

        sNewWord2 := '';
        while (iC <= 14) AND (bFound = false) do
        begin
          sNewWord2 := sNewWord2 + arrChar[iR, iC];

          if sNewWord2 = sWord then
          begin
            bFound := true;
            memQ4_2.Text := 'Row ' + IntToStr(iR)
              + ' @ column ' + IntToStr(iIn) + ' to ' + IntToStr
              (iIn + iLen - 1); // iC - iLen + 1);
            for iC2 := iC downto iC - iLen + 1 do
              arrChar[iR, iC2] := upCase(arrChar[iR, iC2]);
            display2D;
          end;
          inc(iC);
        end;
      end;
    end;
  end;
end;
//============================================================================
// Provided code - Do not change
//============================================================================
procedure TfrmQ4_1.display2D;
var
  iOut, iIn: Integer;
  sOut: string;
begin
  redQ4_2.Clear;
  for iOut := 1 to 14 do
  begin
```

```
      sOut := sOut + #9 + IntToStr(iOut);
  end;
  redQ4_2.Lines.Add('' + #9 + sOut + #13);
  for iOut := 1 to 14 do
  begin
    sOut := #13 + IntToStr(iOut) + #9;
    for iIn := 1 to 14 do
    begin
      if iCheck = 1 then
      Begin
        arrChar[iOut, iIn] := lowercase(arrChar[iOut, iIn] + '')[1];
        sOut := sOut + #9 + arrChar[iOut, iIn];
      End
      else
        sOut := sOut + #9 + arrChar[iOut, iIn];
    end;
    redQ4_2.Lines.Add(sOut);
  end;
  iCheck := 0;
end;
procedure TfrmQ4_1.FormCreate(Sender: TObject);
var
  iOut, iIn: Integer;
  sOut: String;
begin
   pgcQ4.ActivePageIndex := 0;
  // Q4.2
  redQ4_2.Clear;
  redQ4_2.Paragraph.TabCount := 15;
  redQ4_2.Paragraph.Tab[0] := 20;
  redQ4_2.Paragraph.Tab[1] := 40;
  redQ4_2.Paragraph.Tab[2] := 60;
  redQ4_2.Paragraph.Tab[3] := 80;
  redQ4_2.Paragraph.Tab[4] := 100;
  redQ4_2.Paragraph.Tab[5] := 120;
  redQ4_2.Paragraph.Tab[6] := 140;
  redQ4_2.Paragraph.Tab[7] := 160;
  redQ4_2.Paragraph.Tab[8] := 180;
  redQ4_2.Paragraph.Tab[9] := 200;
  redQ4_2.Paragraph.Tab[10] := 220;
  redQ4_2.Paragraph.Tab[11] := 240;
  redQ4_2.Paragraph.Tab[12] := 260;
  redQ4_2.Paragraph.Tab[13] := 280;
  redQ4_2.Paragraph.Tab[14] := 300;
  display2D;
end;

procedure TfrmQ4_1.btnResetClick(Sender: TObject);
Var
  i: Integer;
begin
  iCheck := 1;
  display2D;
end;

end.
```