basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2021

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 22 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 10) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 11 to 22) 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 10) 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	**FormCreate event** Set caption of lblQ1_1 to 'Coding is ' ✓ Set font colour of lblQ1_1 to green ✓ Set font size of lblQ1_1 to 16 ✓ Set font name to 'Arial' ✓	4	
1.2	**Button - [1.2 – Number of rolls]** Create constant SWEETS_PER_ROLL = 8 ✓ Declare integer variables ✓ Extract number of breaks from spin edit ✓ Calculate total number of sweets (breaks x 4) ✓ Calculate total number of rolls (sweets/SWEETS_PER_ROLL) ✓ Round up (Ceil/RoundTo) ✓ Display number of rolls of sweets on lblQ1_2 ✓ converted to String ✓	8	
1.3	**Button – [1.3 – Calculate volume]** rRadius := 3; rTetraVolume := 133 ✓ rVolume := rTetraVolume + 4/3*PI ✓*power(rRadius,3) ✓/2 ✓ Correct values in formula ✓ Display volume ✓ with one decimal ✓	7	
1.4	**Button - [1.4 – Display pattern]** Clear output area ✓ Extract symbol from combo box ✓ iSize = position of symbol in combo box ✓ + 1 ✓ Loop rows from 1✓ to iSize ✓ Initialise output String ✓ (Or adding #13) Loop columns from 1 ✓ to iSize ✓ Add symbol to output line ✓ Display output line ✓	11	

1.5	**Button - [1.5 – Characters]** Randomly generate correct value ✓ Extract/Convert to character ✓ Loop ✓ Add random character ✓ to output String ✓ Set as previous character ✓ Generate new current character ✓ Until current character ✓ = previous character ✓ Display output String ✓	10	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – SQL AND DATABASE PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**		
2.1.1	**Button [2.1.1 – Clubs from Gauteng and SA affiliated]** `SELECT ClubName, ClubTown FROM tblClubs` ✓ `WHERE Province = "GP"` ✓ `AND SA_Affiliated = True` ✓	3	

2.1.2	**Button [2.1.2 – Birth year]**
	`SELECT MemberName, MemberSurname, BirthDate` ✓
	`FROM tblMembers` ✓
	`WHERE YEAR (BirthDate) = 2002` ✓
	
	`Alternative for year:`
	`BETWEEN #01/01/2002# and #31/12/2002#`
		3	

2.1.3	**Button [2.1.3 – Display members]** `SELECT MemberSurname, MemberName` ✓ `FROM tblClubs,tblMembers` ✓ `WHERE tblClubs.ClubId = tblMembers.ClubId` ✓ `AND tblClubs.ClubName = "' + sClubName +'"'` ✓	4	

2.1.4	**Button [2.1.4 - Average membership fee]**
	`SELECT Province, FORMAT(AVG(MemFee)` ✓`,`
	`"CURRENCY")`✓
	`AS AvgFee` ✓
	`FROM tblClubs`
	`GROUP BY Province` ✓
	`HAVING` ✓
	`AVG(MemFee) > 400` ✓
		6	

2.1.5	**Button [2.1.5 - Change member name]** `UPDATE tblmembers` ✓ `SET MemberName = "Ainsley"` ✓ `WHERE MemberName = "Aiensley"` ✓	3	
	Subtotal:	**19**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**Database Manipulation**		
2.2.1	**Button [2.2.1 – Outstanding fees]** Go to the fist record in the tblClubs ✓ Use a loop to step through the tblClubs ✓ Display the club name and annual membership fee for each club ✓ Go to the first record of the tblMembers table ✓ Use a loop to step through tblMembers ✓ Compare the ClubID field in tblClubs ✓ with the ClubID field in the tblMembers ✓ If the comparison is true calculate the difference ✓ between the AmountPaid and the MemFee ✓ Display the surname, amount paid and the difference in the richedit ✓ in currency Move to the next record in the tblMembers ✓ End loop (Members table) Move to the next record in the tblClubs ✓ End loop (tblClubs)	12	
2.2.2	**Button [2.2.2 – Update hikes completed]** Edit mode ✓ Add 1 ✓ to HikesCompleted field ✓ Post ✓	4	
2.2.3	**Button [2.2.3 – Add member]** Retrieve the club Id from the radiogroup ✓ Insert mode ✓ Assign values to correct fields ✓ Assign retrieved club id to the ClubID field ✓ Post ✓	5	
	Subtotal:	**21**	
	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**getNumberOfDays function** Function heading with integer value as return data type ✓ fNumberOfDays assigned to Result ✓	**2**	
3.1.2	**calcDistPerDay** round✓ (fDistance/fNumberOfDays) ✓ Correct attributes ✓	**3**	
3.1.3	**determineLevel function** Call calcDistPerDay ✓ If distance per day > 15 ✓ and terrain type = rocky ✓ or terrain type = sandy ✓ 'Advanced' assigned to Result ✓ Else ✓ If truncated distance per day is from 10 ✓ to 15 ✓ 'Moderate' assigned to Result ✓ Else 'Easy' assigned to Result ✓	**10**	
3.1.4	**calcCost function** Function heading with real value as return data type ✓ and integer parameter ✓ fCost * parameter ✓ assigned to Result ✓	**4**	
3.1.5	**toString method** Function heading with String return data type ✓ Return trailName, terrainType, distance, number of days, cost ✓ Converted to correct format ✓ Correct text and line breaks ✓	**4**	
	Subtotal: Object class	**23**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Combobox - cmbHikingTrails** Assignfile with extracted file name + '.txt' ✓ Reset file ✓ Read 4 lines from file ✓ *Instantiate the objHikingTrail object:* objHikingTrail:= THikingTrail.Create ✓ Use five arguments ✓ with correct data types ✓ and in correct order ✓ Display message ✓	**8**	
3.2.2	**Button [3.2.2 – Display hiking trail details]** Use toString method ✓ to display hiking trail information in rich edit component ✓	**2**	
3.2.3	**Button [3.2.3 – Display cost]** Input number of members in group ✓ Call calcCost method ✓ And assign value to provided cost variable ✓ With input (size of group) as argument ✓	**4**	
3.2.4	**Button [3.2.4 – Calculate distance per day]** Display number of km using **calcDistPerDay** method ✓ Display number of days using **getNumberOfDays** method ✓ Display difficulty level using **determineLevel** method ✓	**3**	
	Subtotal Form class:	**17**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
SECTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1	**Button - [4.1 – Display distance chart]** Initialize output String Loop from 1 to 5 ✓ Add name to output String ✓ Loop from 1 to 5 ✓ Add distance to output String ✓ Display output String ✓ in rich edit	**5**	
4.2	**Button - [4.2 – Validation]** Loop from 1 to 5 ✓ Loop from 1 to 5 ✓ Test if distance at [iRow,iCol]<>distance at [iCol,iRow] ✓ If distance at [iCol,iRow] = floor distance at [iRow,iCol] ✓ Replace with distance at [iRow, iCol] ✓ Build String with row and col index And distance ✓ Repeat test for distance at [iCol, iRow] ✓ Display output String ✓	**8**	
4.3	**Button - [4.3 – Route planner]** Extract route from combo box and initialise total distance ✓ Loop 4 times ✓ (extract all possible combinations) Extract row index ✓ Extract column index ✓ Delete first 2 characters ✓ (logic to start at next index) Read distance from 2D ✓ using row and column Update total distance ✓ (add distance from 2D) Test if hike between points at 2 and 4 OR 4 and 2 ✓ multiply time with 2 ✓ Calculate time (time X distance) ✓ Update time per day ✓ Display names of two checkpoints ✓ Display distance and time ✓ Test if time per day > 480 ✓ Display name of checkpoint to book ✓ Reset time per day to 0 ✓ Display total distance ✓	**17**	
	TOTAL SECTION D: **GRAND TOTAL:**	**30** **150**	

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

```
// ========================================================================
// Question 1.1          4 marks
// ========================================================================
procedure TfrmQuestion1.FormCreate(Sender: TObject);
begin
  lblQ1_1.Caption := 'Coding is ';
  lblQ1_1.Font.Color := clGreen;
  lblQ1_1.Font.Size := 16;
  lblQ1_1.Font.Name := 'Arial';
end;

// ========================================================================
// Question 1.2         8 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  iNumStops, iNumRolls, iTotalSweets: integer;
const
  SWEETS_PER_ROLL = 8;
begin
  iNumStops := spnQ1_2.Value;
  iTotalSweets := iNumStops * 4;
  iNumRolls := Ceil(iTotalSweets / SWEETS_PER_ROLL);
  lblQ1_2.Caption := IntToStr(iNumRolls);
end;

// ========================================================================
// Question 1.3         7 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  rVolume, rRadius, rTetraVolume: real;
begin
  rRadius := 3;
  rTetraVolume := 133;
  rVolume :=  rTetraVolume + 4/3 * PI * power(rRadius,3)/2;
  ShowMessage('Volume: ' + FloatToStrF(rVolume,ffFixed,10,1);
end;
```

```
// ========================================================================
// Question 1.4          11 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sSymbol, sLine: String;
  iRows, iCol, iRepeat: integer;
begin
  redQ1_4.Clear;
  sSymbol := cmbQ1_4.Text;
  iRepeat := cmbQ1_4.ItemIndex + 1;

  for iRows := 1 to iRepeat do
  begin
    sLine := '';
    for iCol := 1 to iRepeat do
      sLine := sLine + sSymbol + ' ';
    redQ1_4.Lines.Add(sLine);
  end;

//Alternative solution

 for iRows := 1 to iRepeat do
   sLine := sLine + sSymbol + ' ';
 for iRows := 1 to iRepeat do
   redQ1_4.Lines.Add(sLine);
end;

// ========================================================================
// Question 1.5          10 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  cCharCurr: char;
  cCharPrev:cChar;
  sOut: String;
begin
//Provided code
  redQ1_5.Clear;
  sOut := ''; //variable for output String

  cCharCurr := Char(random(90 - 65 + 1) + 65);
  repeat
    sOut := sOut + cCharCurr;
    cCharPrev := cCharCurr;
    cCharCurr := Char(random(90 - 65 + 1) + 65);
  until (cCharCurr = cCharPrev);
  sOut := sOut + cCharCurr;
  redQ1_5.Lines.Add(sOut);
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
//=========================================================================
// Question 2.1 – Section: SQL statements
//=========================================================================

//=========================================================================
//  Question 2.1.1         3 marks
//=========================================================================
   sSQL1 := 'SELECT ClubName, ClubTown
             FROM tblClubs
             WHERE Province = "GP"
             AND SA_Affiliated = true ';
            // or without = true

//=========================================================================
//  Question 2.1.2         3 marks
//=========================================================================
    sSQL2 := 'SELECT MemberName, MemberSurname, BirthDate
             FROM tblMembers
             YEAR (BirthDate) = 2002';
          // Alternative for year: BETWEEN #01/01/2002# AND #31/12/2002#

//=========================================================================
//  Question 2.1.3         4 marks
//=========================================================================
   sSQL3 := 'SELECT  MemberSurname, MemberName FROM tblClubs,tblMembers
             WHERE tblClubs.ClubId = tblMembers.ClubID
             AND tblClubs.ClubName = "' + sClubName +'"';
//=========================================================================
//  Question 2.1.4         6 marks
//=========================================================================
   sSQL4 := 'SELECT Province, FORMAT(AVG(MemFee), "Currency") AS
             AvgFee FROM tblClubs
             GROUP BY Province HAVING AVG(MemFee) > 400';
//=========================================================================
//  Question 2.1.5         3 marks
//=========================================================================
   sSQL5 := 'UPDATE tblMembers
             SET MemberName = "Ainsley"
             WHERE MemberName = "Aiensley"';

//=========================================================================
// Question 2.2 – Section Delphi code
//=========================================================================

//=========================================================================
// Question 2.2.1         12 marks
// =========================================================================
procedure TfrmDBQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  rDifference : real;
begin
// Question 2.2.1
  tblClubs.First;
```

```
  while NOT(tblClubs.EOF) do
    begin
     redQ2_2_1.Lines.Add(tblClubs['ClubName']+','+ 'annual fee='+
                         FloatToStrF(tblClubs['MemFee'],ffCurrency,3,2)+
                         '============================');
     redQ2_2_1.Lines.Add(#13+'Surname'+#9+'Paid'+#9+'Outstanding');
     tblMembers.First;
     while NOT(tblMembers.EOF) do
        begin
          if tblClubs['ClubID'] = tblMembers['ClubID'] then
            begin
              rDifference := tblClubs['MemFee']-tblMembers['AmountPaid'];
              redQ2_2_1.Lines.Add(tblMembers['MemberSurname'] + #9 +
                                  FloatToStrF(tblMembers['AmountPaid'],
                                  ffCurrency,3,2) + #9 +
                                  FloatToStrF(rDifference,ffCurrency,3,2));
            end;
           tblMembers.Next;
        end;
        redQ2_2_1.Lines.Add('');
       tblClubs.Next;
    end;
  // Provided code
  dbCONN.SetupGrids(dbgrdONE, dbgrdMany, dbgrdSQL);
end;

// ========================================================================
// Question 2.2.2          4 marks
// ========================================================================
procedure TfrmDBQuestion2.btnQ2_2_2Click(Sender: TObject);
begin
  // Question 2.2.2

  tblMembers.Edit;
  tblMembers['HikesCompleted'] := tblMembers['HikesCompleted'] + 1;
  tblMembers.Post;
end;

// ========================================================================
// Question 2.2.3          5 marks
// ========================================================================
procedure TfrmDBQuestion2.btnQ2_2_3Click(Sender: TObject);
var
  sSurName, sName, sYear, sMemberCode : String;
  dBirthDate : TDateTime;
  iHikesCompleted, iClubID : integer;
  rAmountPaid : real;
begin
  //Provided code
  sSurname := 'Nkosi';
  sName := 'Mothupi';
  dBirthDate := 18-08-2003;
  sMemberCode := 'Nko2140';

  //Question 2.2.3
  iClubID := rgpQ2_2_3.ItemIndex + 1;
```

```
  tblMembers.Insert;
  tblMembers['MemberCode'] := sMemberCode;
  tblMembers['MemberSurName'] := sSurname;
  tblMembers['MemberName'] := sName;
  tblMembers['BirthDate'] := dBirthDate;
  tblMembers['ClubID'] := rgpQ2_2_3.Items[rgpQ2_2_3.ItemIndex][1];
  tblMembers.Post;
  tblMembers.Refresh;
end;

// =====================================================
// {$ENDREGION}
// =====================================================
// {$REGION 'Provided code: Setup DB connections - DO NOT CHANGE!'}
// =====================================================
procedure TfrmDBQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  // Restore the Database
  dbCONN.RestoreDatabase;
  redQ2_2_2.Clear;
  dbCONN.SetupGrids(dbgrdONE, dbgrdMany, dbgrdSQL);
end;

// =====================================================
procedure TfrmDBQuestion2.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin // Disconnect from database and close all open connections
  dbCONN.dbDisconnect;
end;

procedure TfrmDBQuestion2.FormCreate(Sender: TObject);
begin
  redQ2_2_2.Paragraph.TabCount := 4;
  redQ2_2_2.Paragraph.Tab[0] := 70;
  redQ2_2_2.Paragraph.Tab[1] := 150;
  redQ2_2_2.Paragraph.Tab[2] := 300;
  redQ2_2_2.Paragraph.Tab[3] := 400;
end;

// =====================================================
procedure TfrmDBQuestion2.FormShow(Sender: TObject);
begin // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblClubs := dbCONN.tblOne;
  tblMembers := dbCONN.tblMany;

  dbCONN.setupGrids(dbgrdONE, dbgrdMany, dbgrdSQL);
  pgcDBAdmin.ActivePageIndex := 0;
end;
// =====================================================
// {$ENDREGION}

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
unit HikingTrail_U;

interface

uses SysUtils;

type

  THikingTrail = class(TObject)

  private
  var
    // Provided code
    fTrailName,fTerrainType: String;
    fNumberOfDays: integer;
    fDistance: integer;
    fCostPP: real;

  public
    constructor create(sTrailName,sTerrainType: String; iNumDays: integer;
                       rDistance: integer; rCost: real);
    function getNumberOfDays: integer;
    function calcDistPerDay: integer;
    function determineLevel: String;
    function calcTotalCost(iNum: integer):real;
    function toString:String;
 end;

implementation

{ THikingTrail }

// Provided code

constructor THikingTrail.Create(sTrailName,sTerrainType: String;
           iNumDays: integer; rDistance: integer; rCost: real);
begin
  fTrailName := sTrailName;
  fTerrainType := sTerrainType;
  fNumberOfDays := iNumDays;
  fDistance := rDistance;
  fCostPP := rCost;
end;

// ============================================================================
// Question 3.1.1         2 marks
// ============================================================================
   function THikingTrail.getNumberOfDays: integer;
   begin
     Result := fNumberOfDays;
   end;
```

```
// ============================================================================
// Question 3.1.2          3 marks
// ============================================================================
   function THikingTrail.calcDistPerDay: integer;
   begin
      Result :=(Round(fDistance/fNumberOfDays));
   end;
// ============================================================================
// Question 3.1.3          10 marks
============================================================================
   function THikingTrail.determineLevel: String;
   var
     distPday : real;
   begin
     //distPday := fDistance / fNumberOfDays;
     distPday := calcDistPerDay;   // call function
     if  ( dPday > 15 ) and ((fTerrainType = 'Rocky') or
           (fTerrainType = 'Sandy')) then
       Result := 'Advanced'
     else if (distPday >=10) and (distPday <=15) then
              Result := 'Moderate'
          else Result := 'Easy' ;
end;
// ============================================================================
// Question 3.1.4         4 marks
// ============================================================================
   function THikingTrail.calcTotalCost(iNum: integer): real;
   begin
     Result := fCostPP * iNum;
   end;
// ============================================================================
// Question 3.1.5          4 marks
// ============================================================================
   function THikingTrail.toString: String;
   begin
     Result := fTrailName + ': ' + fTerrainType + #13 +
               FloatToStrF(fDistance,ffGeneral, 6, 2) + ' km in ' +
               IntToStr(fNumberOfDays) + ' days' + #13 +
               'Cost per person: ' + FloatToStrF(fCostPP,ffCurrency,8,2);
   end;

   end.
```

Main Form Unit

```
// ========================================================================
// Question 3.2.1         8 marks
// ========================================================================
procedure TfrmHiking.cmbQ3_2_1Change(Sender: TObject);
var
  tFile: TextFile;
  sTrail,sType,sDist,sNumber,scost: String;
  rDist,iNumDays: integer;
  rCost: real;
begin
//Provided code - do not change
  sTrail := cmbQ3_2_1.Text;
  imgTrail.Picture.LoadFromFile(sTrail + '.jpg');

//Question 3.2.1
  assignFile(tFile,sTrail+'.txt');
  reset(tFile);
  readln(tFile,sType);
  readln(tFile,sDist);
  readln(tFile,sNumber);
  readln(tFile,sCost);

  objHikingTrail := THikingTrail.create(sTrail,sType,StrToInt(sNumber),
                      StrToInt(sDist),StrToFloat(sCost));
  MessageDlg('Object has been instantiated.',mtInformation,[mbOk],0);
  //Provided code
  btnQ3_2_2.Enabled := True;
  btnQ3_2_3.Enabled := True;
  btnQ3_2_4.Enabled := true;
end;
// ========================================================================
// Question 3.2.2         2 marks
// ========================================================================
    procedure TForm2.btnQ3_2_2Click(Sender: TObject);
    begin
    //Question 3.2.2
      redQ3_2_2.Lines.Clear;
      redQ3_2_2.Lines.Add(objHikingTrail.toString);
    end;
// ========================================================================
// Question 3.2.3         4 marks
// ========================================================================
    procedure TForm2.btnQ3_2_3Click(Sender: TObject);
    var
      iNum : integer;
      rCost: real;
    begin
    //Question 3.2.3
      iNum := StrToInt(InputBox('Enter number of people','','7'));
      rCost := objHikingTrail.calcTotalCost(iNum);

      //Provided code
      pnlQ3_2_4.Caption := FloatToStrF(rCost,ffCurrency,10,2);
    end;
```

```
// ========================================================================
// Question 3.2.4        3 marks
// ========================================================================
   procedure TForm2.btnQ3_2_4Click(Sender: TObject);
   begin
      //Question 3.2.4
       redQ3_2_4.Lines.Clear;
       redQ3_2_4.Lines.Add('You have to cover at least ' +
        IntToStr(objHikingTrail.calcDistPerDay) + ' km per day to complete
        this ' +  UpperCase(objHikingTrail.DetermineLevel) + ' hiking trail
       in ' + IntToStr(objHikingTrail.getNumberOfDays) + ' days.' );
   end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
// ========================================================================
// Question 4.1          5 marks
// ========================================================================
procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  iRow, iCol: integer;
  sOut: String;
begin
  // Provided code
  redQ4.Lines.Add(sHeading);
  redQ4.Lines.Add(' ');

  // Question 4.1

  for iRow := 1 to 5 do
  begin
    sOut := #13 + arrNames[iRow] + #9;
    for iCol := 1 to 5 do
    begin
      sOut := sOut + FloatToStr(arrDistances[iRow, iCol]) + #9;
    end;
    redQ4.Lines.Add(sOut);
  end;
end;
// ============================================================
// Question 4.2          8 marks
// ============================================================
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  iRow, iCol: integer;
  sOut: String;
begin
// Provided code
  redQ4.Clear;
  redQ4.Lines.Add('Replace distance at:');

  // Question 4.2
  for iRow := 1 to 5 do
  begin
    for iCol := 1 to 5 do
    begin
      if arrDistances[iRow, iCol] <> arrDistances[iCol, iRow] then
      begin

        if arrDistances[iRow, iCol] = floor(arrDistances[iCol, iRow]) then
        begin
          arrDistances[iRow, iCol] := arrDistances[iCol, iRow];
          sOut := '[' + IntToStr(iRow) + ',' + IntToStr(iCol)
            + '] with ' + FloatToStr(arrDistances[iCol, iRow]);
        end;
        if arrDistances[iCol, iRow] = floor(arrDistances[iRow, iCol]) then
        begin
          arrDistances[iCol, iRow] := arrDistances[iRow, iCol];
```

```
            sOut := '[' + IntToStr(iCol) + ',' + IntToStr(iRow)
            + '] with ' + FloatToStr(arrDistances[iRow, iCol]);
        end;
        redQ4.Lines.Add(sOut);
      end;
    end;
  end;
end;
//============================================================================
// Question 4.3          17 marks
// ============================================================================
procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  iCnt, iRow, iCol: integer;
  rTotalDistance, rTimePerDay, rTimeMin: real;
  sRoute: String;

begin
  // Provided code
  redQ4.Clear;

  // Question 4.3
  sRoute := cmbRoutes.Text;
  redQ4.Lines.Add('Route: ' + sRoute + #13);
  rTotalDistance := 0;
  rTimePerDay := 0;
  for iCnt := 1 to 4 do
  begin
    iRow := StrToInt(copy(sRoute, 1, 1));
    iCol := StrToInt(copy(sRoute, 3, 1));
    Delete(sRoute, 1, 2);

    if (iRow IN [2, 4]) AND (iCol IN [2, 4]) then
      rTimeMin := 20 * 2 * arrDistances[iRow, iCol]
    else
      rTimeMin := 20 * arrDistances[iRow, iCol];

    redQ4.Lines.Add(arrNames[iRow] + ' to ' + arrNames[iCol] + ': '
                + FloatToStr(arrDistances[iRow, iCol]) +
                ' (' + FloatToStr(rTimeMin) + ' minutes)');

    rTotalDistance := rTotalDistance + arrDistances[iRow, iCol];

    rTimePerDay := rTimePerDay + rTimeMin;

    if rTimePerDay > 480 then
    begin
      redQ4.Lines.Add('Book at ' + arrNames[iCol] + #13);
      rTimePerDay := 0;
    end;
  end;

  redQ4.Lines.Add(#13 + 'Total distance: ' + FloatToStrF
      (rTotalDistance, ffFixed, 8, 1));
end;
```

```
// ============================================================
// Provided code - do not change
// ============================================================
procedure TfrmQuestion4.FormActivate(Sender: TObject);
var
  i, iPos: integer;
begin
  redQ4.Paragraph.TabCount := 6;
  iPos := 78;
  for i := 1 to 6 do
  begin
    redQ4.Paragraph.Tab[i] := iPos;
    inc(iPos, 78);
  end;
  sHeading := '' + #9 + 'Morgan' + #9 + 'Haga Haga' + #9 + 'Cintsa' + #9 +
    'Beacon' + #9 + 'Gonubie';
end;


end.
```