basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE12

INFORMATION TECHNOLOGY P1

NOVEMBER 2019

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 24 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 3-12) include the marking grid for each question for using a programming language.

- **Annexures E, F, G and H** (pages 13-24) contain examples of solutions for Questions 1 to 4 in programming code.

- Copies of **Annexures A, B, C, D and Summary of learner's marks** (pages 3-12) should be made for each learner and completed during the marking session.

ANNEXURE A

QUESTION 1: MARKING GRID- GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
1.1	**Button [1.1 - Display amount due]** Declare a constant variable PRICE and set it to 14.95 ✓ Declare a real variable for the amount due ✓ Declare a variable for the number of pizzas ✓ Retrieve number from spnQ1_1✓ Calculate amount using PRICE ✓ Change font size of label to 20pt ✓ Display amount on lblQ1_1✓ converted to string and currency format ✓	**8**	
1.2	**Button [1.2 – Pythagoras]** Create variables for sides A, B and C ✓ Assign length of 4 to A ✓ Extract length of side B ✓ and convert to number format✓ C = sqrt ✓ (sqr(A) + sqr(B)) correctly applied sqr(A) OR A * A✓ correctly applied sqr(rB) OR B * B✓ adding up the two values✓ Display length of side C on panel✓ formatted to one decimal place✓ **Accept**: Sqr: Power(A, 2) sqrt : Power((sqr(A) + sqr(B)),0.5)	**10**	
1.3	**Button [1.3 - Determine lowest number]** Assign random number ✓in the correct range✓ to variable Display number✓ convert to string✓ Test (if) ✓ number < lowest ✓ Assign number to lowest ✓ Display lowest in edtQ1_3✓ converted to string✓ **Accept**: RandomRange(1, 101) Random(100) + 1 Ceil(Random() * 100)	**9**	

1.4	**Button [1.4 - Display decrypted string]** Read input string using an input box ✓ With correct parameters ✓ Loop ✓ from first ✓ to last character ✓ Correct test performed (If/Case) ✓ Extract character at index ✓ Test against digit ✓ for all 10 digits ✓ Replace digit at correct index ✓ With correct character ✓ for all 10 digits ✓ Display✓ **Concepts** Getting input from input box (1) with correct parameters (1) Loop (1) from first (1) to last character (1) Correct test (If/Case) (1) Extract character at index (1) Test against digit (1) for all 10 digits (1) Replace digit at correct index (1) With correct character (1) for all 10 characters (1) Display (1)	**13**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – SQL AND DATABASE

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**		
2.1.1	**Button [2.1.1 – Best players]** `SELECT PlayerSurname, PlayerName FROM tblPlayers WHERE SkillsLevel = 10` **Concepts:** SELECT PlayerSurname, PlayerName✓ FROM tblPlayers ✓ WHERE SkillsLevel = 10✓	**3**	
2.1.2	**Button [2.1.2 – B-team coaches]** `SELECT Coach, TeamName FROM tblTeams WHERE TeamName Like "%B"` **Concepts:** SELECT Coach, TeamName ✓ FROM tblTeams ✓ WHERE TeamName LIKE✓ "%B"✓	**4**	
2.1.3	**Button [2.1.3 – Percentage games won]** `SELECT TeamName, Coach, (NumberOfGamesWon/NumberOfGamesPlayed*100) AS PercentageGamesWon FROM tblTeams WHERE TeamName = "' + sTeam + '"` **Concepts:** SELECT TeamName, Coach, ✓ (NumberOfGamesWon/NumberOfGamesPlayed✓*100), ✓ AS PercentageGamesWon FROM tblTeams WHERE TeamName = " ' + sTeam + ' " '✓ **Alternative:** 'WHERE TeamName = ' + QuotedStr(sTeam)	**4**	

2.1.4	**Button [2.1.4 - Team average more than 6]** `SELECT TeamName, ROUND(AVG(SkillsLevel),1) AS AverageSkillsLevel FROM tblPlayers GROUP BY TeamName HAVING AVG(SkillsLevel) > 6` **Concepts:** SELECT TeamName, ROUND(AVG(SkillsLevel) ✓,1) ✓ AS AverageSkillsLevel ✓ FROM tblPlayers GROUP BY TeamName✓ HAVING AVG(SkillsLevel) > 6✓	**5**	
2.1.5	**Button [2.1.5 – Update games won]** `UPDATE tblTeams SET NumberOfGamesWon = NumberOfGamesWon + 1 WHERE TeamName <> "u/14 B"` **Concepts:** UPDATE tblTeams ✓ SET NumberOfGamesWon = NumberOfGamesWon +1 ✓ WHERE teamName <> "u/14 B"✓	**3**	
	Subtotal:	**19**	

2.2	**Database manipulation using Delphi code**		
2.2.1	**Button [2.2.1 – Junior players in u/18 A team]** AssignFile ✓& Rewrite✓ Set tblPlayers to start reading first record✓ Loop while NOT tblPlayers.EOF✓ Test if teamName is 'u/18 A' ✓ AND if first 2 characters of IDnumber field is >=3✓ Write surname and name to file✓ Increment counter by 1✓ Go to next record in tblPlayers✓ End loop Close file✓ Display counter for number of junior players on label ✓	**11**	

2.2.2	**Button [2.2.2 – Coach and goalkeeper information]** Loop while NOT tblTeams.EOF✓ Set tblPlayers to start reading first record✓ Loop while NOT tblPlayers.EOF✓ Test if: the TeamName field in tblTeams equals the TeamName field in tblPlayers ✓ AND Goalkeeper = true✓ Add team name, coach name, ✓ player surname and name in the correct format to output string ✓ Go to next record in tblPlayers✓ Display output line✓ Go to next record in tblTeams✓ **Alternative:** Loop while NOT tblPlayers.EOF (1) Test Goalkeeper = true (1) Set tblTeams to start reading first record (1) Loop while NOT tblTeams.EOF (1) Test if the TeamName field in tblTeams equals the TeamName field in tblPlayers (1) Add team name, coach name, (1) player surname and name in the correct format to output string (1) Go to next record in tblTeams (1) Display output line (1) Go to next record in tblPlayers (1)	**10**	
	Subtotal:	**21**	

	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID - OBJECT-ORIENTED PROGRAMMING

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.1	**Constructor method:** Heading with correct parameters and data type✓ Assign name of player parameter value to fPlayerName ✓ Assign weight of player parameter value to fWeightOfPlayer ✓ Set fScore to 0 ✓	**4**	
3.1.2	**getScore method:** Function heading with integer as return data type ✓ fScore assigned to result ✓	**2**	
3.1.3	**updateScore method:** Procedure heading with integer parameter ✓ Increment fScore ✓ Using the parameter value ✓	**3**	
3.1.4	**calculateBMI method:** Function declared with real return data type and a real parameter for the height and return calculated BMI ✓ Calculation: fWeightOfPlayer / ✓ Sqr (parameter value height of player) ✓	**3**	
3.1.5	**eligibleForSelection method:** Test if score is a value between 0 and 7 (inclusive) ✓ result = Low possibility ✓ Test if score is between 8 and 14 (inclusive) result = Medium possibility ✓ Test if score is > 14 result = High possibility ✓	**4**	
3.1.6	**toString method:** Labels (Name, Weight, Current score) ✓ Correct attributes ✓ Correct conversions (weight – float; score – integer) ✓ Return string ✓	**4**	
	Subtotal: Object class	**20**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 - Instantiate object]** *Instantiate the objPlayer object:* objPlayer := ✓ TPlayer.Create ✓ Pass name and weight in correct order ✓ and correct data type (same as constructor) ✓ Use dialog box to indicate object has been instantiated✓	**5**	
3.2.2	**Button [3.2.2 - Calculate BMI]** Use Input Dialog box ✓ to enter height Call the method calculateBMI using the object✓ using the height as argument ✓ converted to float Display the information in **redQ3_2_2** ✓ using the toString method of the object ✓ Display the BMI in **redQ3_2_2** ✓ in the correct format to one decimal place ✓	**7**	
3.2.3	**Button [3.2.3 – Update score]** Retrieve itemindex from the radio group ✓ to get the score ✓ Call the updateScore method ✓ using correct argument ✓ Display the updated score in **pnlQ3_2_3** ✓ by using the getScore method ✓	**6**	
3.2.4	**Button [3.2.4 – Eligible for selection]** Call the relevant method as follows: Display on the label **lblQ3_2_4** ✓ Using objPlayer.eligibleForSelection ✓	**2**	
	Subtotal: Form class	**20**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID–PROBLEM SOLVING

CENTRE NUMBER:	EXAMINATION NUMBER:		
SECTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1	**Button [4.1 – Display maze]** Select maze from combo box✓ Correct file name + '.txt' ✓ Test and display a message if the text file does not exist ✓ Assign and reset the file ✓ Initialise index variable ✓ Loop through the text file ✓ Increment the value of index ✓ (position depends on initialisation) Read line ✓ Assign the line to the array arrMaze✓ Display the line number ✓ and line ✓	**11**	
4.2	**Button [4.2 – Longest corridor]** Initialize variable to save maximum value to 0✓ Loop through array✓ Initialize variable to keep longest corridor in line✓ Initialize variable to count consecutive '-' in line✓ Loop through each character ✓ on each line ✓ Test if character is dash (-)✓ Increment dash counter✓ If dash counter more than longest in line✓ Replace longest corridor in line with counter✓ Else if character is not –✓ Set dash counter to 0✓ Save longest corridor in line in array/string✓ If longest in line longer than overall longest ✓ – replace ✓ Display message with length of longest corridor✓ Loop through structure with longest corridors per line✓ Display line number✓ where length is same as maximum length✓ **CONCEPTS**: Initialise variable for longest corridor to 0 (1) Loop through the array (1) Initialise dash counter per line (1) and maximum number of continued dashes per line (1) Loop through each character in the line (2) Test if character is a dash (1) Increment the dash counter (1) Test if the dash counter is longest in that line (2) and save to maximum number (1) If character is not a dash (1) Reset the dash counter to zero (1)	**19**	

	Determine longest passage in all the lines in the maze (test and save max) (2) Display longest corridor in the maze (1) Test for lines with same length of longest corridor (2) Display line number(s) (1)		
	TOTAL SECTION D:	**30**	

GRAND TOTAL:	**150**	

SUMMARY OF LEARNER'S MARKS:

NUMBER OF CENTER:		**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, ExtCtrls, ComCtrls, Spin, pngimage;

type
  TfrmQ1 = class(TForm)
    gpbQ1_1: TGroupBox;
    gpbQ1_4: TGroupBox;
    gpbQ1_3: TGroupBox;
    btnQ1_1: TButton;
    btnQ1_4: TButton;
    edtQ1_3: TEdit;
    btnQ1_3: TButton;
    redQ1_3: TRichEdit;
    btnReset: TButton;
    GroupBox1: TGroupBox;
    btnQ1_2: TButton;
    Label1: TLabel;
    pnlQ1_2: TPanel;
    lblQ1_1: TLabel;
    lbl1_2Pizzas: TLabel;
    edtQ1_2Diameter: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    edtQ1_2Width: TEdit;
    Label4: TLabel;
    spnQ1_1: TSpinEdit;
    Label5: TLabel;

    procedure btnResetClick(Sender: TObject);
    procedure btnQ1_1Click(Sender: TObject);
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);

  private
    { Private declarations }
  public
    iLowest: integer; // global
    { Public declarations }
  end;

var
  frmQ1: TfrmQ1;

implementation

{$R *.dfm}
```

```
=============================================================================
                        Question 1.1 = 8 marks
=============================================================================
procedure TfrmQ1.btnQ1_1Click(Sender: TObject);
const
  PRICE = 14.95;
var
  iNumber: integer;
  rAmount: real;
begin
  lblQ1_1.Font.Size := 20;
  iNumber := spnQ1_1.Value;
  rAmount := iNumber * PRICE;
  lblQ1_1.Caption := FloatToStrF(rAmount, ffCurrency, 10, 2);
end;

=============================================================================
                        Question 1.2 = 10 marks
=============================================================================
procedure TfrmQ1.btnQ1_2Click(Sender: TObject);
var
  rA, rB, rC : real;
begin
  // Question 1.2
  rA := 4;
  rB := StrToFloat(edtQ1_2.Text);
  rC := Sqrt(Sqr(rA) + Sqr(rB));
  pnlQ1_2.Caption := FloatToStrF(rC,ffFixed,8,1);
end;
=============================================================================
                        Question 1.3 = 9 marks
=============================================================================
procedure TfrmQ1.btnQ1_3Click(Sender: TObject);
var // Provided code
  iNumber: integer;

begin
  iNumber := random(100) + 1;
  redQ1_3.lines.Add(IntToStr(iNumber));
  if iNumber < iLowest then
    iLowest := iNumber;
  edtQ1_3.Text := IntToStr(iLowest);
end;
```

```
=========================================================================
                      Question 1.4 = 13 marks
=========================================================================
procedure TfrmQ1.btnQ1_4Click(Sender: TObject);
const
  ALPHA = 'ABCDEFGHIJ';

var // Provided code
  sString, sDisplay: String;
  I, iIndex: integer;

begin
  // Provided code
  sString := InputBox('Enter an encrypted string: ', '', '');
  sDisplay := '';
  for I := 1 to Length(sString) do
    if sString[I] in ['0' .. '9'] then
      begin
        iIndex := strtoint(sString[I]) + 1;
        sDisplay := sDisplay + ALPHA[iIndex];
      end
    else
      sDisplay := sDisplay + sString[I];
  ShowMessage(sDisplay);

end;

=========================================================================
                      Provided code
=========================================================================
procedure TfrmQ1.btnResetClick(Sender: TObject);
begin
  // given code - do not change
  iLowest := 100;
  redQ1_3.Clear;
  edtQ1_3.Clear;
end;

procedure TfrmQ1.FormActivate(Sender: TObject);
begin
  // given code - do not change
  iLowest := 100;
  redQ1_3.Clear;
  edtQ1_3.Clear;
  CurrencyString := 'R';
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

QUESTION 2.1: SQL code

```
=====================================================================
  Question 2.1: 2.1.1(3), 2.1.2(4), 2.1.3(4), 2.1.4(5), 2.1.5 (3)
=====================================================================

2.1.1: SELECT PlayerSurname, PlayerName
       FROM tblPlayers WHERE SkillsLevel = 10

2.1.2: SELECT Coach, TeamName
       FROM tblTeams
       WHERE TeamName Like "%B"

2.1.3: SELECT TeamName, Coach,
       (NumberOfGamesWon/NumberOfGamesPlayed*100)
       AS [PercentageGamesWon]
       FROM tblTeams
       WHERE TeamName = "' + sTeam + '"

2.1.4: SELECT TeamName, ROUND(AVG(SkillsLevel),1)
       AS [AverageSkillsLevel]
       FROM tblPlayers
       GROUP BY TeamName HAVING AVG(SkillsLevel) > 6

2.1.5: UPDATE tblTeams
       SET NumberOfGamesWon = NumberOfGamesWon + 1
       WHERE TeamName <> "u/14 B"
```

QUESTION 2.2: DATABASE MANIPULATION using Delphi Code

```
// {$REGION 'QUESTION 2.2'}

=====================================================================
                      Question 2.2.1 = 11 marks
=====================================================================
procedure TfrmDBQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  tFile: textfile;
  iCnt, iYear: integer;
begin
  AssignFile(tFile, 'Junior18A.txt');
  Rewrite(tFile);
  tblPlayers.first;
  iCnt := 0;
  while not tblPlayers.eof do
  begin
    iYear := strToInt(copy(tblPlayers['IDNumber'], 1, 2));
    if (tblPlayers['TeamName'] = 'u/18 A') AND(iYear >= 3) then
      begin
        Writeln(tFile,
         tblPlayers['PlayerSurname'] + ' ' + tblPlayers['PlayerName']);
        inc(iCnt);
      end;
    tblPlayers.Next;
  end;
```

```
  CloseFile(tFile);
  lblQ2_2_1.Caption := 'Number of young players: ' + IntToStr(iCnt);

  // Provided code
  dbCONN.setupGrids(dbgrdONE, dbgrdMANY, dbgrdSQL);
end;

 ========================================================================
                      Question 2.2.2 = 10 marks
 ========================================================================
procedure TfrmDBQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  sOut: String; // variable to save selected team

begin
  // -- Provided code  ----
  redQ2_2_2.Clear;
  redQ2_2_2.Paragraph.TabCount := 2;
  redQ2_2_2.Paragraph.Tab[0] := 100;
  redQ2_2_2.Paragraph.Tab[1] := 200;
  redQ2_2_2.Lines.Add('TeamName' + #9 + 'Coach' + #9 + 'Goalkeeper');

  // ------------------------------------------------
  // Type your code here:

  tblTeams.first;
  while NOT tblTeams.eof do
  begin
    sOut := tblTeams['TeamName'] + #9 + tblTeams['Coach'] + #9;
    tblPlayers.first;
    while NOT tblPlayers.eof do
    begin
      if (tblTeams['TeamName'] = tblPlayers['TeamName']) AND
(tblPlayers['GoalKeeper'] = true)
        then
      begin
        sOut := sOut + tblPlayers['PlayerSurname'] + ', ' + tblPlayers
          ['PlayerName'];
      end;
      tblPlayers.Next;
    end;
    tblTeams.Next;
    redQ2_2_2.Lines.Add(sOut);
  end;
```

```
//=============================================================
{$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.setupControls(grpTB_1,grpTB_2);
  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
  CurrencyString := 'R';
end;

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

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

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
unit Player_U;

interface

uses StdCtrls, SysUtils;

type
  TPlayer = class(TObject)

  //Provided code - do not modify
  private
      fPlayerName: String;
      fWeightOfPlayer : real;
      fScore : integer;

  public
    constructor create(sPlayerName : String; rWeightOfPlayer : real);
    function getScore : integer;
    function calculateBMI (rHeightOfPlayer : real) : real;
    procedure updateScore (iScore : integer);
    function eligibleForSelection: String;
    function toString : String;
end;

implementation

{ TPlayer }
=============================================================================
                      Question 3.1.1 = 4 marks
=============================================================================
constructor TPlayer.create(sPlayerName : String; rWeightOfPlayer :
real);
begin
   fPlayerName := sPlayerName;
   fWeightOfPlayer := rWeightOfPlayer;
   fScore := 0;
end;

=============================================================================
                      Question 3.1.2 = 2 marks
=============================================================================
function TPlayer.getScore: integer;
begin
  result := fScore;
end;

=============================================================================
                      Question 3.1.3 = 3 marks
=============================================================================
procedure TPlayer.updateScore(iScore: integer);
begin
  fScore := fScore  + iScore;
end;
```

```
=============================================================================
                        Question 3.1.4 = 3 marks
=============================================================================
function TPlayer.calculateBMI(rHeightOfPlayer: real) : real;
begin
  Result := fWeightOfPlayer / sqr(rHeightOfPlayer);
end;

=============================================================================
                        Question 3.1.5 = 4 marks
=============================================================================
function TPlayer.eligibleForSelection: String;
begin
 if fScore < 8 then
   result := 'Low possibility'
 else
   if fScore < 15 then
      result := 'Medium possibility'
   else
      result := 'High possibility'
end;

=============================================================================
                        Question 3.1.6 = 4 marks
=============================================================================
function TPlayer.toString : String;
begin
   result := 'Name: ' + fPlayerName + #13 + 'Weight: '+
FloatToStr(fWeightOfPlayer)+ #13+ 'Current score is: ' + intToStr
(fScore);
end;
end.
```

Main Form Unit

```
unit Question3_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,Dialogs, StdCtrls, Player_U, ComCtrls, ExtCtrls, Spin, DateUtils;

type
  TfrmQuestion3 = class(TForm)
    lblNameOfPlayer: TLabel;
    edtNameOfPlayer: TEdit;
    btnQ3_2_2: TButton;
    redQ3_2_2: TRichEdit;
    lblLatestScore: TLabel;
    pnlQ3_2_3: TPanel;
    btnQ3_2_3: TButton;
    btnQ3_2_1: TButton;
    rgpQ3_2_3: TRadioGroup;
    btnQ3_2_4: TButton;
    lblQ3_2_4: TLabel;
    Label1: TLabel;
    edtWeightOfPlayer: TEdit;
    Label3: TLabel;
    grpQ3_2_1: TGroupBox;
    grpQ3_2_2: TGroupBox;
    grpQ3_2_3: TGroupBox;
    grpQ3_2_4: TGroupBox;
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnQ3_2_4Click(Sender: TObject);
  //  procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;
  // variables already declared
  objPlayer : TPlayer;
implementation

{$R *.dfm}
```

```
=========================================================================
                      Question 3.2.1 = 5 marks
=========================================================================
procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
  sPlayerName : String;
  rPlayerWeight : real;
begin
      sPlayerName := edtNameOfPlayer.Text;
      rPlayerWeight := StrToFloat(edtWeightOfPlayer.Text);
      objPlayer := TPlayer.create(sPlayerName,rPlayerWeight);
      ShowMessage('Player object has been instantiated');
end;

=========================================================================
                      Question 3.2.2 = 7 marks
=========================================================================
procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var rHeightOfPlayer, rBMI : real;
    sBMI : String;

begin
  rHeightOfPlayer := StrToFloat(InputBox('Height of the player',
           'Enter the height of the player in meters: ',''));
  rBMI := objPlayer.calculateBMI(rHeightOfPlayer);
  sBMI := FloatToStrF(rBMI,ffFixed,3,1);
  redQ3_2_2.Lines.Add(objPlayer.toString);
  redQ3_2_2.Lines.Add ('BMI-index is '+ sBMI);
end;

=========================================================================
                      Question 3.2.3 = 6 marks
=========================================================================
procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
var
  iScore: integer;
begin
   iScore := StrToInt(rgpQ3_2_3.Items[rgpQ3_2_3.ItemIndex]);
   objPlayer.updateScore(iScore);
   pnlQ3_2_3.Caption :=('New current score: ' +
           IntToStr(objPlayer.getScore));
end;

=========================================================================
                      Question 3.2.4 = 2 marks
=========================================================================
procedure TfrmQuestion3.btnQ3_2_4Click(Sender: TObject);
begin
  lblQ3_2_4.Caption := objPlayer.eligibleForSelection;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_u;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, ComCtrls;
type
  TfrmQ4 = class(TForm)
    gpbQuestions: TGroupBox;
    btnQ4_1: TButton;
    btnQ4_3: TButton;
    cmbQ4_1: TComboBox;
    Label1: TLabel;
    redQ4: TRichEdit;
    procedure btnQ4_1Click(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
// Provided code
const
  iRowCount = 16;

var
  frmQ4: TfrmQ4;
  arrMaze: array [1..iRowCount] of String;

implementation
{$R *.dfm}
```

Question 4.1 = 11 marks

```
procedure TfrmQ4.btnQ4_1Click(Sender: TObject);
var
  iSelectMaze: integer;
  index: integer;
  sFileName, sLine: String;
  tNameFile: TextFile;
begin
```

Provided Code

```
  redQ4.Clear;
```

Code required to complete

```
index := 1; //first index in array arrMaze
sFileName := cmbQ4_1.Text + '.txt';
if FileExists(sFileName) then
  begin
    AssignFile(tNameFile, sFileName);
    Reset(tNameFile);
```

```
    While not(eof(tNameFile)) do
    begin
      Readln(tNameFile, sLine);
      arrMaze[index] := sLine;
       redQ4.Lines.Add(IntToStr(index) + #9 + sLine);
      Inc(index);
    end;
  end;
end;

================================================================================
                          Question 4.2 = 19 marks
================================================================================
procedure TfrmQ4.btnQ4_2Click(Sender: TObject);
var
  iRow, iCol, iCount, iLongestInLine, iMax: integer;
  arrRows: array [1..iRowCount] of integer;
  sRow: String;

begin
  iMax := 0;
  for iRow := 1 to iRowCount do
  begin
    sRow := arrMaze[iRow];

    iLongestInLine := 0;
    iCount := 0;
    for iCol := 1 to Length(sRow) do
    begin
      if sRow[iCol] = '-' then
      begin
        Inc(iCount);
        if iCount > iLongestInLine then
          iLongestInLine := iCount
      end
      else
        iCount := 0;
    end;
    arrRows[iRow] := iLongestInLine;

    if iMax < arrRows[iRow] then
      iMax := arrRows[iRow];
  end;

  redQ4.Lines.Add('');
  redQ4.Lines.Add('Longest corridor(s) with ' + IntToStr(iMax)
      + ' spaces in row(s):');
  for iRow := 1 to Length(arrRows) do
  begin
    if arrRows[iRow] = iMax then
      redQ4.Lines.Add('  ' + IntToStr(iRow));
  end;
end;

end.
```