basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2016

MEMORANDUM

MARKS: 150

This memorandum consists of 31 pages.

GENERAL INFORMATION:

- These marking guidelines must 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 and C**(pages 3–9) include the marking grid for each question for using either one of the two programming languages.

- **Annexures D, E and F** (pages 10–19) contain examples of solutions for Java for Questions 1 to 3 in programming code.

- **Annexures G, H and I** (pages 20–31) contain examples of solutions for Delphi for Questions 1 to 3 in programming code.

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

ANNEXURE A

SECTION A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
	If a learner has a problem reading from a combo box, penalise only once for the error.		
1.1	**Button - [Question 1.1]** Extract the distance from the text box as an integer✓ *Formula: Distance = Miles * 1.6*✓ Display distance in textbox with labels✓	**3**	
1.2	**Button - [Question 1.2]** Set temperature = 18 and Set height = 1000✓ Extract the height of mountain from the text box and convert to number✓ Validate height <>1000✓ Display message✓, Clear text box✓ and set Focus✓ Display the heading✓ Loop with correct condition✓ Display height and temperature inside loop✓ Increase height by 100✓ Decrease temperature by 1✓	**11**	
1.3	**Button - [Question 1.3]** Set lowest to value at first index of array heights✓ (or to 9999) Loop through heights array✓ If height at loop index < lowest✓ lowest = height at loop index✓ lowestName = name at loop index✓ Display pass name and lowest height✓	**6**	

1.4	**Button - [Question 1.4]** Extract the number of persons from text box as an integer Extract type of accommodation selected from combo box✓ Case / Switch / If (accommodation type)✓ and allocate correct cost per person to cost variable✓ Multiply cost * number of persons✓ Check if Wi-Fi has been selected✓ and add R150 to total cost✓ Check if radio button for Card is selected✓ Use a dialog box to enter a card number✓ Set valid flag to True✓ If length of card number is Not 9✓ - Set valid flag to False✓ Check if all characters are digits: Loop through ckaracters of card number✓ Use method/function to check if each char is a digit✓ If Not digit – set valid flag to False✓ If the card number is valid Add 3% of cost to total cost✓ If card number Not Valid✓ Use a dialog box to display “Invalid card number”✓ Set radio button selection to cash option✓ Show formatted total cost per night in the text box✓	**19**	
1.5	**Button - [Question 1.5]** Extract transaction number from text box as integer Set prime flag to True✓ Loop from 2✓ to transaction number / 2✓ (or ticket number -1) If transaction number is divisible by loop counter✓ Set prime flag to False✓ If prime is True✓ Generate random number in range 1 to 4✓ Else Set random number = 0✓ Set the value in the list box✓to highlight the position of the random number	**9**	
	TOTAL:	**48**	

ANNEXURE B

SECTION B

QUESTION 2: MARKING GRID – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1.1	**Constructor:** Definition with four correct parameters and data types✓ Initialise all four attributes using the parameters✓	**2**	
2.1.2	**setDangerLevel method:** Method definition with parameter✓ Assign the parameter value to the danger level attribute✓	**2**	
2.1.3	**calculateFine method:** Correct parameters for time travel and speed limit✓ Calculate the speed: Use distance attribute✓ Divide by the travel time✓ divide by 60✓ Initialise fine to 0✓ Check if speed greater than or equal to✓ speedl imit increased by 10✓ Calculate fine: 500✓+ (Speed – speed limit – 10)✓ Rounded up✓ Divide by 3✓ Multiply by 100✓ Return fine as decimal value✓	**13**	
2.1.4	**suggestedDangerLevel method:** Method definition with correct parameter for average rainfall ✓ Set level to value of danger level attribute✓ Test if Gradient > 10✓AND average rainfall >= 10✓ If dangerLevel = “Low” Change level to “Medium”✓ Else If danger Level = “Medium” Change level to “High”✓ Else level = “High”✓ Return level✓	**8**	

2.2.1	**Button – [Question2.2.1]** Read selected mountain pass from radio button and assign to correct pass name variable✓ Read distance from text box as integer value✓ Read danger level from text box as String✓ Read gradient from text box as String✓ Instantiate object using correct values✓ in correct order✓ Display message to indicate object was created✓	**7**	
2.2.2	**Button – [Question2.2.2]** Use object to get mountain pass name, distance,✓ danger level and gradient✓ Assign mountain pass name, distance, danger level and gradient✓ to correct textfields✓ Get the correct filename for the pass (name of the pass)✓ Compile filename using pass name and .jpg✓ Test if file exist✓ If exist – display correct image on component provided✓	**8**	
2.2.3	**Button – [Question2.2.3]** Get speedlimit and travelTime from text boxes✓ in the correct format✓ Get the fine amount by calling the calculateFine method of the object and✓ sending the correct arguments✓ Display the fine✓ in the correct format✓ (Currency to 2 decimal places)	**6**	

2.2.4	**Button – [Question2.2.4]** Use toString method to display the data of the mountain pass in the output area✓ Initialise sum to 0✓ Get row value of pass selected✓✓ Get the rainfall from array elements from the row✓ Add each value to sum✓ Calculate the average rainfall (/7)✓ Call the determineDangerLevel method of the object✓ If return value not the same as object's current danger level✓ Ask if the danger level should be updated✓ If answer = "Yes"✓ Call the setDangerLevel met with argument to change the danger level of the object✓ Set the danger level in the text field to the changed value✓ Display a message to indicate what the danger level was changed to in the output area✓ else if answer = "No" Display a message to indicate that the suggestion has been rejected and the current danger level that will remain in the output area ✓ else Display a message to indicate that no change was suggested and the current danger level that will remain in the output area ✓	**16**	
	TOTAL:	**62**	

ANNEXURE C

SECTION C

QUESTION 3: MARKING GRID – PROBLEM SOLVING PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	**Direct route:** Open the text file to read from file: ✓✓ Delphi: AssignFile, RESET Java: Create object to read from file Set flag to False✓ Loop through the file ✓ Read line of text ✓ Test if town of departure AND destination ✓has been found in the line of text✓but are not the same✓ then Flag = True✓ Find position of '#'✓, Find position of '*'✓ Extract the Distance from the line of text✓ If mountain pass exists (test)✓ "Pass on route" message✓ else "No pass" message✓ Display all information regarding the route✓ If flag = False Display message "Not Found" ✓ Close File✓	**18**	
3.2	**List of towns that can be visited:** Declare appropriate data structures (2 arrays)✓✓ Set counter to 0✓ (or 1 for Delphi) Get the town that was selected from the combo box Open file for reading data from file ✓ Loop through file✓ Read line of text, extract depart town✓ and destination✓ Test whether town = destination town✓ Store name of destination in names array✓ Extract distance and store in distances array✓ Increment counter✓ Test whether town = depart town✓ Store name of depart town in names array✓ Extract distance and store in distances array Increment counter✓ Two nested loops through distances array✓✓ Comparing values✓ Swapping names ✓✓ also swapping distances✓ Loop for display ✓ and displaying towns and distances from arrays neatly in columns✓	**22**	
	TOTAL	**40**	

SUMMARY OF LEARNER'S MARKS:

CENTRE NUMBER:		EXAMINATION NUMBER:		
	SECTION A	SECTION B	SECTION C	
	QUESTION 1	QUESTION 2	QUESTION 3	GRAND TOTAL
MAX. MARKS	48	62	40	150
LEARNER'S MARKS				

ANNEXURE D: SOLUTION FOR QUESTION 1: JAVA

```
package Question1Package;
import java.text.DecimalFormat;
import javax.swing.JOptionPane;

public class Question1_Memo extends javax.swing.JFrame {
//=====================================================================
//Supplied code
//=====================================================================
  String arrPassNames[] = {"Barkly Pass", "Baviaanskloof Pass",
      "Katberg Pass", "Baster Voetslaan Pass", "Felton Pass", "Chapmans
           Peak Pass"};
  double arrPassHeights[] = {1560, 986, 2341.5, 1268, 987, 1258.9};

  DecimalFormat df = new DecimalFormat("0.00");

  public Question1_Memo() {
        initComponents();
        this.setLocationRelativeTo(this);
    }
//=====================================================================
// Question  1.1
//=====================================================================
private void btnQuestion1_1ActionPerformed(java.awt.event.ActionEvent evt) {
        String distanceValue = txfInfo.getText();
        txfDistance.setText(df.format(Double.parseDouble(distanceValue)
        * 1.60)km");
=====================================================================
// Question 1.2
=====================================================================
private void btnQuestion1_2ActionPerformed(java.awt.event.ActionEvent evt) {
        int temperature = 18;
        double height = 1000;
        double heightTopOfMountain =
                Double.parseDouble(txfHeight.getText());

       if(heightTopOfMountain <=1000){
           JOptionPane.showMessageDialog(null, "Enter a valid height");
           txfHeight.setText("");
       }
          else
       {
        txaOutput.setText(String.format("%-20s%-
        10s%n","Metres","Temperature"));
        do{
             txaOutput.append(String.format("%-28s%-
             10s%n",height,temperature));
               height += 100;
               temperature -= 1;
        }
        while (height <= heightTopOfMountain);
       }


//=====================================================================
// Question 1.3
//=====================================================================
private void btnQuestion1_3ActionPerformed(java.awt.event.ActionEvent evt) {
        double lowest = arrPassHeights[0];
```

```
String name = arrPassNames[0];
        for (int i = 1; i < arrPassHeights.length; i++) {
            if (arrPassHeights[i] < lowest) {
                lowest = arrPassHeights[i];
                name = arrPassNames[i];
            }
        }
        txaLowestHeight.setText("");
        txaLowestHeight.append("The lowest mountain pass is " + name +
            ".\n The height of the pass is " + lowest + " m.");

        }
  }
//=========================================================================
// Question 1.4
//=========================================================================
private void btnQuestion1_4ActionPerformed(java.awt.event.ActionEvent evt) {
        String accommType = (String) (cmbAccommType.getSelectedItem());
        int numPersons = Integer.parseInt(txfNumPersons.getText());
        double cost = 0;
        switch (accommType) {
            case "Hotel":
                cost = 1200;
                break;
            case "B&B":
                cost = 1000;
                break;
            case "Self-catering unit":
                cost = 750;
                break;
            case "Camping site":
                cost = 300;
                break;
        }
        cost *= numPersons;
        if (chbWiFi.isSelected()) {
            cost += 150;
        }
        boolean flag = false;

        if (rbnCash.isSelected()) {
            txfPayment.setText("R " + df.format(cost));
        } else {

         String cardNumber = JOptionPane.showInputDialog("Enter card
         number");
            if (cardNumber.length() == 9) {
                flag = true;
                for (int i = 0; i < 9; i++) {
                    if (!Character.isDigit(cardNumber.charAt(i))) {
                        flag = false;
                    }
                }
            }
            if (!flag) {
           JOptionPane.showMessageDialog(null, "Invalid card number");
                rbnCash.setSelected(true);
                 txfPayment.setText("R " + df.format(cost));
            } else {
                  cost = cost * 1.03;
                txfPayment.setText("R " + df.format(cost));
      }
  }
```

```
//=====================================================================
// Question 1.5
//=====================================================================

private void btnQuestion1_5ActionPerformed(java.awt.event.ActionEvent evt) {
        int transactionNumber =
                  Integer.parseInt(txfNumber.getText());
        Boolean prime = true;
        for (int i = 2; i <= transactionNumber / 2; i++) {
            if (transactionNumber % i == 0) {
                prime = false;
            }
        }

        int randNum = 0;
        if (prime) {
            randNum = (int)(Math.random() * 4)+1;
        }
       lstPrizes.setSelectedIndex(randNum);
    }
```

ANNEXURE E: SOLUTION FOR QUESTION 2: JAVA

```
package Question2_Package;

// Solution for Object class

//=====================================================================
//Supplied code
//=====================================================================

    private String name;
    private int distance;
    private String dangerLevel;
    private int gradient;

    public String getName() {
        return name;
    }

    public int getDistance() {
        return distance;
    }

    public String getDangerLevel() {
        return dangerLevel;
    }

    public int getGradient() {
        return gradient;
    }

//=====================================================================
// Question 2.1.1
//=====================================================================

 public MountainPass(String name, int distance, String dangerLevel,
                     int gradient) {
        this.name = name;
        this.distance = distance;
        this.dangerLevel = dangerLevel;
        this.gradient = gradient;
    }
//=====================================================================
// Question 2.1.2
//=====================================================================
    public void setDangerLevel(String dangerLevel) {
        this.dangerLevel = dangerLevel;
      }
//=====================================================================
// Question 2.1.3
//=====================================================================
public double calculateFine(int travelTime, int speedlimit) {
        double speed = distance / (travelTime / 60.0);
        double fine = 0;

         if (speed >= (speedlimit + 10)){
        fine = 500 + (Math.ceil((speed - speedlimit - 10) / 3.0)) * 100;
        }
        return fine;
    }
```

```
//============================================================================
// Question 2.1.4
//============================================================================
  public String suggestedDangerLevel(double avgRain) {
       String level = dangerLevel;
       if (gradient > 10 && avgRain >= 10) {
          if (dangerLevel.equalsIgnoreCase("Low")) {
         level = "Medium";}
         else if(dangerLevel.equalsIgnoreCase("Medium")){
         level = "High";
         }
         else{
             level = "High";
         }
       }
       return level;
    }

//============================================================================
// Supplied toString method
//============================================================================

public String toString() {
       return name+" with a maximum gradient of "+
         gradient +" degrees has a danger level rating of "+
         dangerLevel +".\nThe distance of the pass is "+ distance+"
                        km.";
}
```

GUI CLASS: QUESTION2_SOLUTION

```
package Question2_Package;

import java.io.File;
import java.io.FileNotFoundException;
import javax.swing.JOptionPane;

public class Question2_Memo extends javax.swing.JFrame {
//=====================================================================
//Supplied code
//=====================================================================
int[][] arrRain = {{0, 23, 13, 1, 2, 0, 14},
                   {33, 3, 11, 35, 3, 0, 21},
                   {50, 0, 0, 1, 20, 0, 2}};
MountainPass mountainPassObj;
String name = "";

public Question2_Memo() {
        initComponents();
        setLocationRelativeTo(this);
        pnlDisplay.setVisible(false);
    }
=====================================================================
// Question 2.2.1
=====================================================================

private void btnQuestion2_1_1ActionPerformed(java.awt.event.ActionEvent evt) {
        if (rbtPass1.isSelected()) {
            name = rbtPass1.getText();
        }

        if (rbtPass2.isSelected()) {
            name = rbtPass2.getText();
        }
        if (rbtPass3.isSelected()) {
            name = rbtPass3.getText();
        }
        int distance =
            Integer.parseInt(JOptionPane.showInputDialog("Enter the
                        distance for the mountain pass.", "110"));
        String dangerLevel = JOptionPane.showInputDialog("Enter the
            danger level of the pass (Low/Medium/High)", "Medium");
        int gradient =
            Integer.parseInt(JOptionPane.showInputDialog("Enter the
                  maximum gradientT in degrees", "12"));
        mountainPassObj = new MountainPass(name, distance, dangerLevel,
                  gradient);
        JOptionPane.showMessageDialog(rootPane, "The object for " + name
                  + " has been instantiated.");
        pnlDisplay.setVisible(true);    }
```

```
//=========================================================================
// Question 2.2.2
//=========================================================================
private void btnQuestion2_2_2ActionPerformed(java.awt.event.ActionEvent evt) {
txfMountainPass.setText(mountainPassObj.getName());
        txfDistance.setText(mountainPassObj.getDistance() + "");
        txfDangerLevel.setText(mountainPassObj.getDangerLevel());
        txfGradient.setText("" + mountainPassObj.getGradient());

        String fName = mountainPassObj.getName() + ".jpg";
        File mapFile = new File(fName);
        if (mapFile.exists()) {
            lblMap.setIcon(new
            javax.swing.ImageIcon(getClass().getResource(fName)));
        }

//=========================================================================
// Question 2.2.3
//=========================================================================

private void btnQuestion2_2_3ActionPerformed(java.awt.event.ActionEvent evt) {
        int speedLimit = Integer.parseInt(txfSpeedLimit.getText());
        int travelTime = Integer.parseInt(txfTravelTime.getText());
        double fine = mountainPassObj.calculateFine(travelTime,
                      speedLimit);
        txfFine.setText(String.format(" R%.2f", fine));
//=========================================================================
// Question 2.2.4
//=========================================================================
private void btnQuestion2_2_4ActionPerformed(java.awt.event.ActionEvent evt) {
        int sum = 0;
        txaOutput.setText("");
        txaOutput.append(mountainPassObj.toString());

        int row = 0;
        if (rbtPass1.isSelected()) {
            row = 0;
        } else if (rbtPass2.isSelected()) {
            row = 1;
        } else {
            row = 2;
        }

        for (int i = 0; i < 7; i++) {
            sum += arrRain[row][i];
        }
        double averageRainfall = (double) sum / 7.0;
        String dangerLevel =
        mountainPassObj.suggestedDangerLevel(averageRainfall);

        if
            (!dangerLevel.equalsIgnoreCase(mountainPassObj.getDangerLevel
                  ())) {
            String answer = JOptionPane.showInputDialog("Must danger
                  level be changed to " + dangerLevel);
            if (answer.equalsIgnoreCase("Yes")) {
                mountainPassObj.setDangerLevel(dangerLevel);
                txfDangerLevel.setText(dangerLevel);
                txaOutput.append("\n\nPass danger level rating changed
                  to  " + mountainPassObj.getDangerLevel().toUpperCase());
            } else {
                txaOutput.append("\n\nSuggestion rejected. Pass danger
                  level rating kept at " +
                        (mountainPassObj.getDangerLevel().toUpperCase()));
```

```
            }
        } else {
            txaOutput.append("\n\nNo suggested change. Danger level
            remains " +
                  (mountainPassObj.getDangerLevel().toUpperCase()));
        }

    private void btnClosActionPerformed(java.awt.event.ActionEvent evt) {
        // Supplied code
        System.exit(0);
    }

}
```

ANNEXURE F: SOLUTION FOR QUESTION 3: JAVA

```
package Question3Package;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;
import javax.swing.JOptionPane;

public class Question3_Memo extends javax.swing.JFrame {

    public Question3_Memo() {
        initComponents();
        setLocationRelativeTo(this);
    }

=========================================================================
//Question 3.1
=========================================================================

private void btnDirectRoutesActionPerformed(java.awt.event.ActionEvent evt) {
        String startTown = "" + cmbStart.getSelectedItem();
        String destTown = "" + cmbDestination.getSelectedItem();
        boolean found = false;
        try {
            int cnt = 0;
            Scanner scFile = new Scanner(new FileReader("Q3Data.txt"));
            while (scFile.hasNext() && !found) {
                String line = scFile.nextLine();
                String[] temp = line.split(";|#|\\*");
                int p1 = line.indexOf(startTown);
                int p2 = line.indexOf(destTown);
                int distance = Integer.parseInt(temp[2]);
                String pass = "Route includes a mountain pass.";
                if (p1 >= 0 && p2 >= 0 && p1 != p2) {
                    found = true;
                    if (temp[3].equals("No")) {
                        pass = "The route does not include a mountain
                              pass.";
                    }
                    txaOutput.setText("The distance from " + startTown +
                        " to " + destTown + " is " + distance + " km.\n" +
                              pass);
                }
            }
            scFile.close();
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(rootPane, e);
        }
        if (!found) {
            txaOutput.setText("No possible route provided between " +
                  startTown + " and " + destTown + ".");
        }
    }
```

```
=========================================================================
//Question 3.2
=========================================================================
private void btnAllRoutesActionPerformed(java.awt.event.ActionEvent evt) {
        String startTown = "" + cmbStart.getSelectedItem();
        txaOutput.setText("Towns to visit when staying in: " +
                  startTown);
        // Determine all destinations
        int numTowns = 0;
        try {
            int cnt = 0;
            Scanner sc = new Scanner(new FileReader("Q3Data.txt"));
            while (sc.hasNext()) {
                String line = sc.nextLine();
                String[] temp = line.split(";|#|\\*");
                if (startTown.equals(temp[0])) {
                    visitTowns[numTowns] = temp[1];
                    visitDistance[numTowns] = Integer.parseInt(temp[2]);
                    numTowns++;
                }
                  if (startTown.equals(temp[1])) {
                    visitTowns[numTowns] = temp[0];
                    visitDistance[numTowns] = Integer.parseInt(temp[2]);
                    numTowns++;
                 }
            }
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(rootPane, e);
        }
        sortTowns(numTowns);
        txaOutput.setText("Towns that can be directly reached from " +
            startTown
                  + "\n");
        for (int cnt = 0; cnt < numTowns; cnt++) {
            txaOutput.append(visitTowns[cnt] + "\t" + visitDistance[cnt]
            + "km\n");
        }
    }

    public void sortTowns(int numTowns)
    {
        for (int outside = 0; outside < numTowns - 1; outside++) {
           for (int inside = outside + 1; inside < numTowns; inside++) {
               if (visitDistance[outside] > visitDistance[inside]) {
                    int dist = visitDistance[outside];
                    visitDistance[outside] = visitDistance[inside];
                    visitDistance[inside] = dist;
                    String town = visitTowns[outside];
                    visitTowns[outside] = visitTowns[inside];
                    visitTowns[inside] = town;
                }
            }
        }
    }
```

ANNEXURE G: SOLUTION FOR QUESTION 1: DELPHI

```
unit Question1U;

// A solution for Question 1

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

type
  TfrmQuestionONE = class(TForm)
    bmbClose: TBitBtn;
    gpbQuest1_1: TGroupBox;
    gpbQuest1_4: TGroupBox;
    gpbQuest1_2: TGroupBox;
    gpbQuest1_3: TGroupBox;
    gpbQuest1_5: TGroupBox;
    lblGPS: TLabel;
    lblDistance: TLabel;
    edtInfo: TEdit;
    btnQuest1_1: TButton;
    edtHeight: TEdit;
    x: TLabel;
    lblTemperature: TLabel;
    btnQuest1_2: TButton;
    btnQuest1_3: TButton;
    redQ1_3: TRichEdit;
    lblHeight: TLabel;
    lblAccommodation: TLabel;
    cboType: TComboBox;
    rbgPayment: TRadioGroup;
    lblPaymentRequired: TLabel;
    edtPayment: TEdit;
    chbWiFi: TCheckBox;
    lblTicketNumber: TLabel;
    lblPrize: TLabel;
    btnQuest1_5: TButton;
    btnQuest1_4: TButton;
    edtNumber: TEdit;
    lblNumPersons: TLabel;
    edtNumPersons: TEdit;
    redOut: TRichEdit;
    lstPrizes: TListBox;
    edtDistance: TEdit;
    procedure btnQuest1_1Click(Sender: TObject);
    procedure btnQuest1_2Click(Sender: TObject);
    procedure btnQuest1_3Click(Sender: TObject);
    procedure btnQuest1_4Click(Sender: TObject);
    procedure btnQuest1_5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestionONE: TfrmQuestionONE;

implementation

{$R *.dfm}
```



```
{$R+}

uses Math;

const
  arrPassHeights: array [1..6] of Real =
            (1560, 986, 2341.5, 1268, 986,1258.9);
  arrPassNames: array [1..6] of String = ('Barkly Pass',
    'Baviaanskloof Pass', 'Katberg Pass', 'Baster Voetslaan Pass',
    'Felton Pass', 'Chapmans Peak Pass');

// ================================================================
// Question 1.1
// ================================================================
procedure TfrmQuestionONE.btnQuest1_1Click(Sender: TObject);
var
  sDistance : String;
  rDistance : Real;
begin
  sDistance := edtInfo.text;
  rDistance := StrToFloat(sDistance) * 1.60;
  edtDistance.Text:= FloatToStrF(rDistance, ffFixed, 8,2)+' km';
end;
// ================================================================
// Question 1.2
// ================================================================
procedure TfrmQuestionONE.btnQuest1_2Click(Sender: TObject);
var
  rHeight : Real;
  iTemperature, iHeight : Integer;
begin
  iHeight := 1000;
  iTemperature := 18;
  rHeight := StrToFloat(edtHeight.Text);
  if rHeight >= 1000 then
    begin
     redOut.Lines.Add('Metres' + #9 + '     Temperature');
     while iHeight < rHeight do
       begin
         redOut.Lines.Add(IntToStr(iHeight) + #9#9 +
            IntToStr(iTemperature));
         Inc(iHeight, 100);
         Dec(iTemperature);
       end;
    end
   else
    begin
      showmessage('Value must be at least 1000');
      edtHeight.clear;
      edtHeight.setFocus;
    end;
end;


// ================================================================
// Question 1.3
// ================================================================
procedure TfrmQuestionONE.btnQuest1_3Click(Sender: TObject);
var
  iLoop : Integer;
  rLow  : Real;
  sName : String;
begin
  rLow := arrPassHeights[1];
```

```
  sName := arrPassNames[1];
  for iLoop := 2 to length(arrPassNames) do
    if arrPassHeights[iLoop] < rLow then
     begin
      rLow := arrPassHeights[iLoop];
      sName := arrPassNames[iLoop];
     end;
  redQ1_3.Lines.Add('The lowest mountain pass is ' + sName);
  redQ1_3.Lines.Add('The height of the mountain pass is ' +
            FloatToStr(rLow) + ' m');
end;

// ================================================================
// Question 1.4
// ================================================================
procedure TfrmQuestionONE.btnQuest1_4Click(Sender: TObject);
var
  iCost, iNumPersons, A: Integer;
  rTotal : Real;
  sCardNumber : String;
  bValid : Boolean;
  iCardNumber, iError : Integer;
begin
  iNumPersons := StrToInt(edtNumPersons.Text);
  case cboType.ItemIndex of
    0: iCost := 1200;
    1: iCost := 1000;
    2: iCost := 750;
    3: iCost := 300;
  end;

  if chbWiFi.Checked then
    rTotal := iCost * iNumPersons + 150
  else
    rTotal := iCost * iNumPersons;

  if rbgPayment.ItemIndex = 1 then
  begin
   sCardNumber := inputbox('Card Number', 'Enter your card number', '');

   bValid := length(sCardNumber) = 9;
   Val(sCardNumber, iCardNumber, iError);
    //Alternative validation
    //    For A := 1 to length(sCardNumber) do
    //     if NOT(sCardNumber[A] in ['0'..'9']) then
    //       bValid := false;
   if (iError = 0) AND bValid then
     begin
        rTotal := rTotal * 1.03;
     end
   else
     begin
        MessageDlg('The card number is invalid', mtError, [mbOk], 0);
        rbgPayment.ItemIndex := 0;
     end;
  end;//if

  edtPayment.Text := FloatToStrF( rTotal, ffCurrency, 7, 2);
end;
```

```
// ====================================================================
// Question 1.5
// ====================================================================
procedure TfrmQuestionONE.btnQuest1_5Click(Sender: TObject);
var
  iTransactionNum : Integer;
  iNumber, iLoop, iRandom : Integer;
  sPrime : Boolean;
begin
  iTransactionNum := StrToInt(edtNumber.Text);

  sPrime := true;

  for iLoop := 2 to iTransactionNum div 2 do
    if iTransactionNum mod iLoop = 0 then
      sPrime := false;

  if (sPrime) then
      iRandom := random(4) + 1
    else iRandom := 0;
  lstPrizes.ItemIndex := iRandom;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 2: DELPHI

OBJECT CLASS: MOUNTAINPASS

```
unit MountainPassU;
// Solution Question 2 object file
interface

type
  TMountainPass = class(TObject)
  private
    fName        : String;
    fDistance    : Integer;
    fDangerLevel : String;
    fGradient    : Integer;

  public
    function GetName: String;
    function GetDistance: Integer;
    function GetDangerLevel: String;
    function GetGradient: Integer;

    constructor Create(sName: String; iDistance: Integer; sDangerLevel:
                        String;Gradient: Integer);
    procedure SetDangerLevel(sDanger: String);
    function CaldulateFine(iTravelTime: Integer; rSpeedLimit: Real):
                                                                  Real;
    function SuggestedDangerLevel(iAvgRain: Integer): String;
    function toString: String;
  end;

implementation

uses SysUtils;

{ TMountainPass }
//=========================================================================
// Supplied code
//=========================================================================
function TMountainPass.GetDistance: Integer;
begin
  Result := fDistance;
end;
function TMountainPass.GetGradient: Integer;
begin
  Result := fGradient;
end;

function TMountainPass.GetName: String;
begin
  Result := fName;
end;

function TMountainPass.GetDangerLevel: String;
begin
  Result := fDangerLevel;
end;

//=========================================================================
//Question 2.1.1
//=========================================================================
constructor TMountainPass.Create(sName: String; iDistance: Integer;
     sDangerLevel: String; iGradient: Integer);
```

```
begin
  fName     := sName;
  fDistance := iDistance;
  fDangerLevel  := sDangerLevel;
  fGradient := iGradient;
end;
//============================================================================
//Question 2.1.2
//============================================================================
procedure TMountainPass.SetDangerLevel(sDanger: String);
begin
  fDangerLevel := sDanger;
end;
//============================================================================
//Question 2.1.3
//============================================================================
function TMountainPass.CalculateFine(iTravelTime: Integer;
                                           iSpeedLimit: Integer): Real;
var
  rSpeed : Real;
begin
  rSpeed := fDistance / (iTravelTime / 60);
  Result := 0; // init
  if (rSpeed >= (iSpeedLimit + 10)) then
   Result := 500 + (Round((rSpeed - iSpeedLimit - 10) / 3 + 0.5) * 100);
end;
//============================================================================
//Question 2.1.4
//============================================================================
function TMountainPass.suggestedDangerLevel(iAvgRain: Integer): String;
begin
    Result := fDangerLevel;
    if (fGradient > 10) AND (iAvgRain >= 10) then
         if (fDangerLevel = 'Low') then
          result := 'Medium'
         else
          if (fDangerLevel = 'Medium') then
            result := 'High'
        else
           result := 'High';
end;
//============================================================================
//Supplied toString method
//============================================================================
function TMountainPass.toString: string;
begin
    Result := fName + ' with a maximum gradient of '
         + IntToStr(fGradient) + ' degrees has a danger level
                  rating of '+ fDangerLevel +'. ' + #13
         + 'The distance of the pass is ' + IntToStr(fDistance) + ' km.'
end;
end.
```

MAIN FORM UNIT: QUESTION2_U.PAS

```
unit Question2U;

//A solution for Question 2

interface

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

type
  TQuestionTWO = class(TForm)
    pnlCloseBtn: TPanel;
    bmbClose: TBitBtn;
    pnlDisplay: TPanel;
    lblMountainPass: TLabel;
    lblDistance: TLabel;
    lblSurface: TLabel;
    lblGradient: TLabel;
    Label5: TLabel;
    edtMountainPass: TEdit;
    edtDistance: TEdit;
    edtDangerLevel: TEdit;
    edtGradient: TEdit;
    imgMap: TImage;
    btnQuest2_2_3: TButton;
    btnQuest2_2_4: TButton;
    redQ2: TRichEdit;
    pnlInput: TPanel;
    rgpMPass: TRadioGroup;
    btnQuest221: TBitBtn;
    btnQuest2_2_2: TButton;
    pnlSpeed: TPanel;
    lblSpeedLimit: TLabel;
    lblSpeed: TLabel;
    Label1: TLabel;
    lblMinutes: TLabel;
    edtSpeedLimit: TEdit;
    edtTime: TEdit;
    Label2: TLabel;
    edtFine: TEdit;
    procedure btnQuest2_2_3Click(Sender: TObject);
    procedure btnQuest2_2_4Click(Sender: TObject);
    procedure btnQuest2_2_2Click(Sender: TObject);
    procedure btnQuest221Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  QuestionTWO: TQuestionTWO;
   MountainPass : TMountainPass;
   sFileName:string;


const
  arrRain : array[1..3,1..7] of Integer =
            ((0,23,13,1,2,0,14),
             (33,3,11,35,3,0,21),
```

```
            (50,0,0,1,20,0,2));

implementation

{$R *.dfm}
{$R+}

//=====================================================================
//Question 2.2.1
//=====================================================================
procedure TQuestionTWO.btnQuest221Click(Sender: TObject);
var
 sDangerL, sName : String;
 iDistance, iGradient : Integer;

begin
  sName := rgpMPass.Items[rgpMPass.ItemIndex];

  iDistance := StrToInt(InputBox('Question 2',
                    'Enter the distance for the mountain pass in metres',
'110'));
  sDangerL := InputBox('Question 2',
                    'Enter the danger level for the pass
                       (Low/Medium/High)', 'Medium');
  iGradient := StrToInt(InputBox('Question 2',
                    'Enter the maximum gradient for the in degrees',
                       '12'));

  MountainPass := TMountainPass.Create(sName, iDistance, sDangerL,
            iGradient);
  MessageDlg('The object for '+ sName + ' has been instantiated.',
            mtInformation,
                  [mbOk], 0);
  pnlDisplay.Show;
end;
//=====================================================================
//Question 2.2.2
//=====================================================================

procedure TQuestionTWO.btnQuest2_2_2Click(Sender: TObject);
begin
  edtMountainPass.Text := MountainPass.GetName;
  edtDistance.Text     := IntToStr(MountainPass.GetDistance);

  edtDangerLevel.Text := MountainPass.getDangerLevel;
  edtGradient.Text    := IntToStr(MountainPass.GetGradient);

  sFileName := MountainPass.GetName + '.jpg';
  if FileExists(sFileName) then
     imgMap.Picture.LoadFromFile(sFileName)
  else
     imgMap.Picture.LoadFromFile('NoMap.jpg');
  pnlSpeed.Show;
end;




//=====================================================================
//Question 2.2.3
//=====================================================================
procedure TQuestionTWO.btnQuest2_2_3Click(Sender: TObject);
var
  iSpeedLimit, iTravelTime : Integer;
```

```
  rFine : Real;
begin
  pnlSpeed.Show;
  iSpeedLimit := StrToInt(edtSpeedLimit.Text);
  iTravelTime := StrToInt(edtTime.Text);

  rFine := MountainPass.DetermineFine(iTravelTime, iSpeedLimit);

  edtFine.Text := FloatToStrF(rFine, ffCurrency, 8,2);
end;

//=========================================================================
//Question 2.2.4
//=========================================================================
procedure TQuestionTWO.btnQuest2_2_4Click(Sender: TObject);
var
   indx,iTotal,iPass, iNumMonths : Integer;
   iAvgRainfall : Integer;
   sDangerLevel : String;
begin

   redQ2.Clear;
   redQ2.Lines.Add(MountainPass.toString);
   redQ2.Lines.Add(' ');

   iPass:= rgpMPass.ItemIndex + 1;

   iTotal := 0;
   for indx := 1 to 7 do
       inc(iTotal, arrRain[iPass,indx]);
   iAvgRainfall := Trunc(iTotal / 7);

   sDangerLevel := MountainPass.SuggestedDangerLevel(iAvgRainfall);

   if sDangerLevel <> MountainPass.getDangerLevel then
      begin
        if MessageDlg('Must danger level be changed to ' +
            sDangerLevel,mtWarning,[mbYES,mbNO],0) = mrYES  then
           begin
             MountainPass.SetDangerLevel(sDangerLevel);
             edtDangerLevel.Text := sDangerLevel;
             redQ2.Lines.Add('Pass danger level rating changed to '
                         + UpperCase(MountainPass.GetDangerLevel));
           end
         else redQ2.Lines.Add('Suggestion rejected. Pass danger level
           rating kept at '+ UpperCase(MountainPass.GetDangerLevel));
       end
      else  redQ2.Lines.Add('No suggested change. Danger level remains '
                         + UpperCase(MountainPass.GetDangerLevel));
end;
end.
```

ANNEXURE I: SOLUTION FOR QUESTION 3: DELPHI

```
unit Question3UMemo;

//A solution for Question 3

interface

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

type
  TfrmQuestionThree = class(TForm)
    pnlHeading: TPanel;
    cboStartTown: TComboBox;
    cboDestinationTown: TComboBox;
    grpTowns: TGroupBox;
    lblStart: TLabel;
    lbldestination: TLabel;
    bmbClose: TBitBtn;
    redQ3: TRichEdit;
    btnDirectRoute: TButton;
    btnAllRoutes: TButton;
    procedure btnDirectRouteClick(Sender: TObject);
    procedure btnAllRoutesClick(Sender: TObject);
    function extractDistance(line: string): integer;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestionThree: TfrmQuestionThree;

implementation

{$R *.dfm}
{$R+}


//=============================================================================
//Question 3.1
//=============================================================================
procedure TfrmQuestionThree.btnDirectRouteClick(Sender: TObject);
var
   TFile             : TextFile;
   sLine, sStartTown, sDestTown, sDistance, sPass   : String;
   bFound             : Boolean;
   p1, p2             : integer;
begin
  //Display direct route information between towns
  sStartTown := cboStartTown.Text;
  sDestTown  := cboDestinationTown.Text;

  redQ3.Lines.Clear;
  AssignFile(TFile, 'Q3Data.txt');
  Reset(TFile);
  bFound := False;
  While NOT EOF(TFile) AND NOT bFound do
   begin
```

```
      Readln(TFile, sLine);
      p1 := pos(sStartTown, sLine);
      p2 := pos(sDestTown, sLine);
      if (p1 >0) AND (p2 > 0) AND (p1 <> p2) then
       begin
         bFound := True;

         sDistance := IntToStr(extractDistance(sLine));
         if pos('*Yes', sLine) > 0 then
            sPass := 'The route includes a mountain pass.'
         else
            sPass := 'The route does not include a mountain pass.';

      redQ3.Lines.Add('The distance from ' + sStartTown + ' to ' +
             sDestTown + ' is ' + sDistance + ' km. ' + #13+sPass);
       end; //if
   end; //while
   if NOT bFound then
    begin
      redQ3.Lines.Clear;
      redQ3.Lines.Add('No possible route provided between ' + sStartTown +
                      ' and ' + sDestTown + '.');
    end;
  CloseFile(TFile);
end;

//=====================================================================
//Question 3.2
//=====================================================================
procedure TfrmQuestionThree.btnAllRoutesClick(Sender: TObject);
var
   arrDestinations : Array[1..50] of String;
   arrDistance     : Array[1..50] of Integer;
   iCount, A, B, iTemp, iPos : Integer;
   TFile           : TextFile;
   sStartTown, sDestinationTown, sDistance,
   sLine, sTemp    : String;
begin
  sStartTown := cboStartTown.Items[cboStartTown.ItemIndex];
  redQ3.Lines.Clear;
  redQ3.Paragraph.TabCount := 1;
  redQ3.Paragraph.Tab[0]   := 150;
  redQ3.Lines.Add('Towns that can be directly reached from ' + sStartTown);

//Determine ALL destinations
  AssignFile(TFile, 'Q3Data.txt');
  Reset(TFile);
  iCount := 0;
  While NOT EOF(TFile) AND (iCount < 50) do
   begin
      Readln(TFile, sLine);
      iPos:= pos(sStartTown, sLine);
      if iPos > 0 then
       begin
         inc(iCount, 1);
         Delete(sLine, iPos, length(sStartTown));
         Delete(sLine, pos(';',sline),1);
         arrDestinations[iCount] := Copy(sLine, 1, pos('#', sLine)-1);

         arrDistance[iCount] := extractDistance(sLine);
       end; //if
   end; //while
  CloseFile(TFile);
```

```
  //Sort towns according to distance
  for A := iCount downto 2 do
    for B := 1 to (A-1) do
      if arrDistance[B] > arrDistance[B+1] then
       begin
         iTemp             := arrDistance[B];
         arrDistance[B]    := arrDistance[B+1];
         arrDistance[B+1]  := iTemp;

         sTemp                := arrDestinations[B];
         arrDestinations[B]   := arrDestinations[B+1];
         arrDestinations[B+1] := sTemp;
       end; //sort

  //Display the list of towns
  for A := 1 to iCount do
    redQ3.Lines.Add(arrDestinations[A] + #9 + IntToStr(arrDistance[A]) + '
            km');
end;

function TfrmQuestionThree.extractDistance(line: string): integer;
 var
   iPosHash1, iPosStar: Integer;
 begin
     iPosHash1 := pos('#', line);
     iPosStar := pos('*', line);
     result := StrToInt(copy(line, iPosHash1+1, iPosStar - iPosHash1-1));
 end;

end.
```