basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2016

MARKS: 150

TIME: 3 hours

This question paper consists of 21 pages.

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into THREE sections. Candidates must answer ALL THREE sections.

2. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.

3. This question paper is set in programming terms that are not specific to any particular programming language (Delphi/Java (using the Netbeans IDE)).

4. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements.

5. Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.

6. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper.

7. **Routines such as search, sort and selection must be developed from first principles. You may NOT use the built-in features of a programming language for any of these routines.**

8. Data structures which are not supplied must be defined by you, the programmer.

9. You must save your work regularly on the disk/CD/DVD/flash disk you have been given, or on the disk space allocated to you for this examination session.

10. Make sure that your examination number appears as a comment in every program that you code, as well as on every event indicated.

11. If required, print the programming code of all the programs/classes that you completed. You will be given half an hour printing time after the examination session.

12. At the end of this examination session you must hand in a disk/CD/DVD/flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read.

13. The files that you need to complete this question paper have been given to you on a disk/CD/DVD/flash disk or on the disk space allocated to you. The files are provided in the form of password-protected executable files.

NOTE:

- Delphi candidates must use the file **DelphiDataENGMarch2016.exe**.
- Java candidates must use the file **JavaDataENGMarch2016.exe**.

Do the following:

- Double click on the password-protected executable file.
- Click on the extract button.
- Enter the following password: **MountainP$%#**

Once extracted, the following list of files will be available in the folder **DelphiDataENGMarch2016/JavaDataENGMarch2016**:

Delphi files	**Java (Netbeans) files**
Question1:	**Question1:**
Question1P.dpr	Question1.form
Question1P.res	Question1.java
Question1U.dfm	
Question1U.pas	
Question2:	**Question2:**
Gamkaskloof Pass.jpg	Gamkaskloof Pass.jpg
MountainPassU.pas	MountainPass.java
NoMap.jpg	NoMap.jpg
Prince Alfred Pass.jpg	Prince Alfred Pass.jpg
Question2P.dpr	Question2.form
Question2P.res	Question2.java
Question2U.dfm	
Question2U.pas	
Question3:	**Question3:**
Q3Data.txt	Q3Data.txt
Question3P.dpr	Question3.form
Question3P.res	Question3.java
Question3U.dfm	
Question3U.pas	

SCENARIO

South Africa is considered to be one of the most scenic countries in the world and attract many tourists to local holiday destinations and internationally acclaimed mountain passes. Software programs are being designed to assist with the administration of tourists.

SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

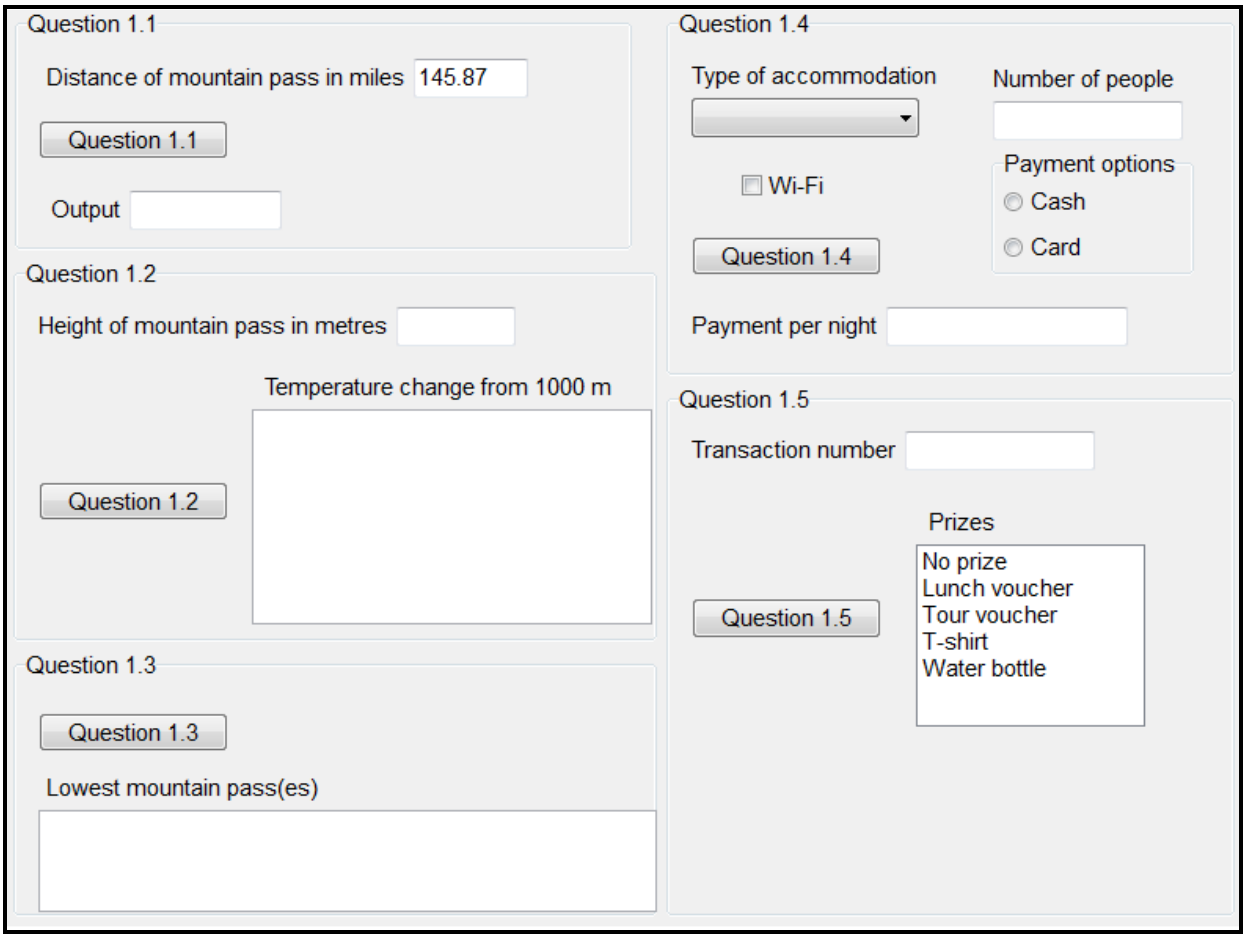
A variety of aspects concerning mountain passes and holiday destinations are dealt with in this question.

INSTRUCTIONS:

Delphi programmers	Java programmers
• The project **Question1** is provided in the **DelphiDataENGMarch2016** folder.	• The project **Question1** is provided in the **JavaDataENGMarch2016** folder.
• Open the incomplete project file **Question1P.dpr** in the **Question1** folder.	• Open the incomplete class **Question1.java** contained in the folders **Source Packages (src)**, **Question1Package**.
• Add your examination number as a comment in the first line of the main form unit **Question1U.pas** file.	• Add your examination number as a comment in the first line of the class **Question1.java**.

Do the following:

- Compile and execute the program. The GUI displays five sections labelled QUESTION 1.1 to QUESTION 1.5. The program currently has no functionality. An example of the GUI is given on the next page:



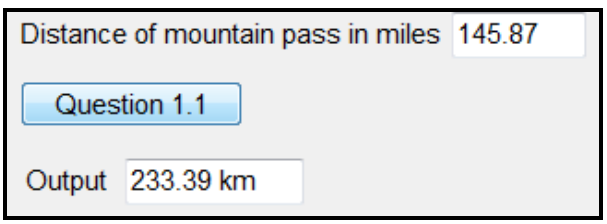
- Complete the code for each section of QUESTION 1 as described in QUESTION 1.1 to QUESTION 1.5 below.

1.1 Distances are displayed in miles on some road maps. The user is required to enter the total distance of a mountain pass in miles.

Write code to convert the distance entered in miles to kilometres. Display the converted value and unit (km) in the provided output component.

NOTE: 1 mile = 1,6 kilometres

Example of output:



(3)

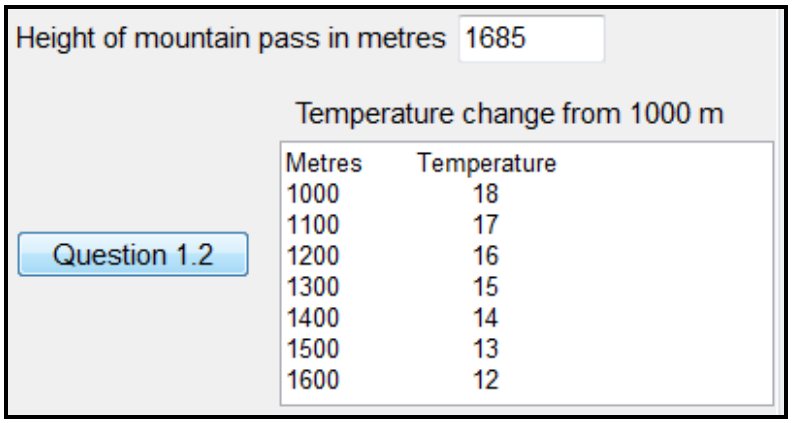
1.2 Many tourists are interested in the decrease in temperature while travelling up a mountain pass. The user is required to enter the height of the mountain pass in metres. The temperature at the 1 000 metre mark on the mountain pass is 18 °C. The temperature decreases by 1 °C for every 100 m increase in the height of the mountain pass.

NOTE: The height of the mountain pass that you enter must be higher than or equal to 1 000 m.

Write code to do the following:

- Obtain the height of the mountain pass from the text box.
- Validate the value entered as follows:
  - If the value is less than 1 000, display a suitable message, clear the text box and set the focus to the text box.
  - If the value is greater than or equal to 1 000, calculate and display the temperature for every 100 m travelled above 1 000 m up the mountain pass to the nearest 100 m below the top of the pass.

Example of output if the height of the pass is 1 685 m:



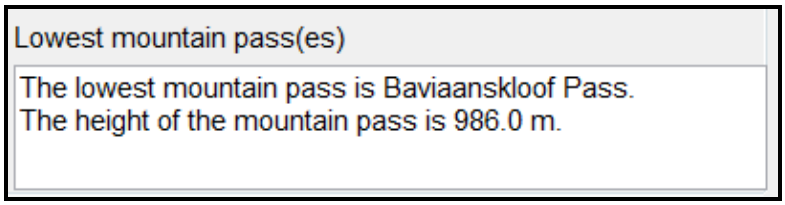
(11)

1.3 Some tourists want to travel up the lowest mountain pass. Two parallel arrays are provided. The array named **arrPassNames** contains the names of mountain passes while the array named **arrPassHeights** contains the heights of the mountain passes contained in the **arrPassNames** array.

Write code to determine the name of the mountain pass with the lowest height.

NOTE: You will be penalised for using fixed values in your code to indicate the number of elements in the arrays. Although the arrays are provided and the number of elements is known, an appropriate built-in function/method must be used to determine the number of elements in the arrays.

Example of output:



(6)

1.4 Tourists can book accommodation close to any of the mountain passes. The types of accommodation available and the price per person per night are given in the table below.

TYPE OF ACCOMMODATION	PRICE PER PERSON PER NIGHT
Hotel	R1 200
Bed-and-breakfast	R1 000
Self-catering unit	R750
Camping site	R300

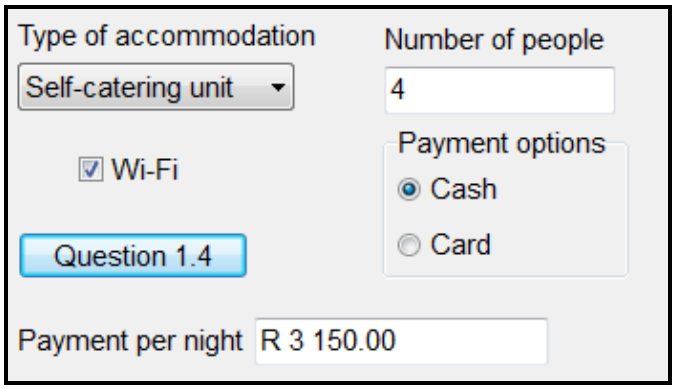
The price of Wi-Fi is R150 per night.

Write code to do the following:

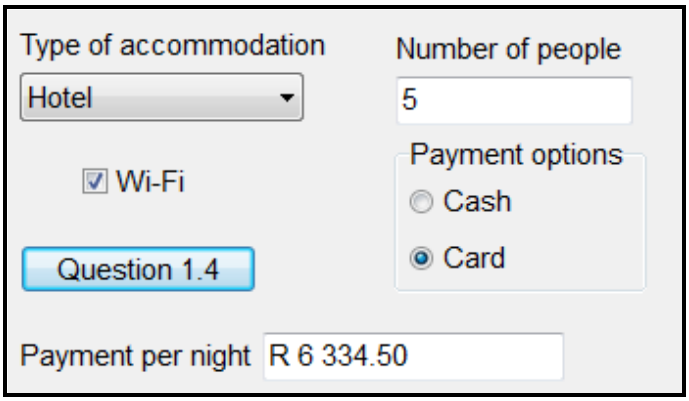
- Obtain the type of accommodation from the combo box.
- Obtain the number of people who require accommodation for this booking from the text box.
- Determine whether the check box control for access to Wi-Fi has been selected.
- Determine the cost per night according to the values entered by the user and the values from the table above.
- Payment options:
  - If the user selects the 'Cash' option, display the payment per night, in currency format with TWO decimal places, in the text box provided.
  - If the user selects the 'Card' option, do the following:
    - Use a dialog box to insert the card number.
    - Do the following TWO tests to ensure that the card number is valid:
      - Ensure that the card number consists of 9 digits.
      - Ensure that the card number consists of only digits.

- If the card number is valid:
  - Add a card charge of 3% to the cost.
  - Display the payment per night in the text box provided.
- If the card number is NOT valid:
  - Use a dialog box to display a message that the card number is invalid.
  - Set the radio button to the 'Cash' option.
  - Display the cash price as the payment per night.

Example of the payment per night for the data shown in the screenshot below:



Example of the payment per night for the data as shown in the screenshot below. The valid card number 789674556 has been entered in the dialog box.



(19)

1.5 Toll fees, which are used to maintain the mountain passes, have to be paid on mountain pass routes. The company maintaining the mountain passes is running a promotion to attract tourists to travel along the mountain passes. Each toll gate receipt has a unique transaction number. The company uses these numbers to give away prizes. The list box provided contains the prizes.

The user is required to enter the transaction number that appears on a toll gate receipt.

The following criteria are used to determine prize winners:

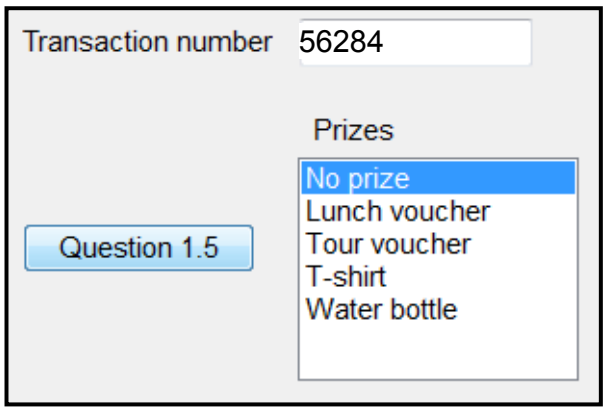
- If the transaction number is a prime number the person will receive a prize which is randomly selected from the prizes in the given list box.
- If the transaction number is a NOT a prime number the user will NOT receive a prize.

Use the algorithm provided below to code the solution.

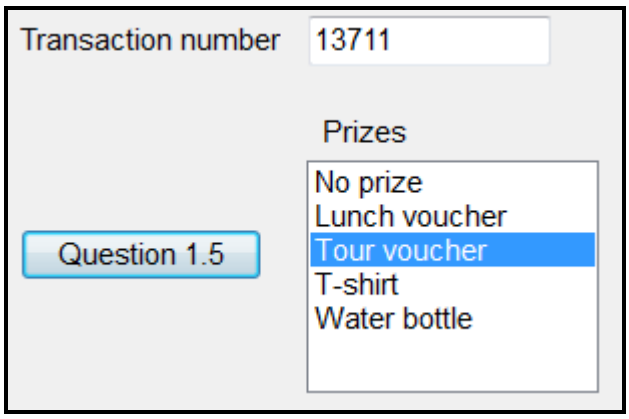
Algorithm:

1. Obtain the transaction number from the text box.
2. Determine whether the transaction number is a prime number.
   (A prime number is a number with only TWO factors, namely 1 and the number itself.)
3. If the transaction number is a prime number, do the following:
   - 3.1 Generate a random number in the range 1 to 4 (including 1 and 4). The random number indicates the index of the prize in the list box, which is the prize that will be allocated to the winner.
   - 3.2 Choose the randomly generated index in the list box in order to highlight the prize.
4. If the transaction number is a NOT prime number, select the 'No prize' option in the list box.

Example of output for transaction number 56284, which is NOT a prime number:



Example of output for transaction number 13711, which is a prime number, and 2 as the randomly generated number:



NOTE: Due to the nature of the random function/method the output generated by your project/program may differ from the example above. (9)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

TOTAL SECTION A: 48

SECTION B

QUESTION 2: OBJECT-ORIENTATED PROGRAMMING

A group of mountain passes has been selected to determine specific statistics, such as the number of travellers, fines received by travellers and details about the danger levels of the mountain passes.

INSTRUCTIONS:

Delphi programmers	Java programmers
• The project **Question2**, provided in the **DelphiDataENGMarch2016** folder, contains: o A main form unit **Question2U.pas** o An incomplete unit file **MountainPassU.pas**	• The project **Question2**, provided in the **JavaDataENGMarch2016** folder, contains: o A GUI class file **Question2.java** o An incomplete object class file **MountainPass.java**
• Open the incomplete project file **Question2P.dpr** and the class unit file **MountainPassU.pas** in the **Question2** folder.	• Open the incomplete classes **Question2.java** and **MountainPass.java** in the folders **Source Packages (src)**, **Question2Package**.
• Add your examination number as a comment in the first line of both files **Question2U.pas** and **MountainPassU.pas**.	• Add your examination number as a comment in the first line of both classes **Question2.java** and **MountainPass.java**.

Do the following:

- Complete the code for each section of QUESTION 2 as described in QUESTION 2.1 and QUESTION 2.2 below.

2.1 An object class called **TMountainPass/MountainPass** with some methods has been provided.

Complete the code in the given class (**TMountainPass/MountainPass**) as described in QUESTION 2.1.1 to QUESTION 2.1.6.

The given attributes of the class **TMountainPass/MountainPass** are described in the table below.

NAMES OF ATTRIBUTES		DESCRIPTION
Delphi	**Java**	
fName	name	The name of the mountain pass
fDistance	distance	Distance in kilometres
fDangerLevel	dangerLevel	Low/Medium/High
fGradient	gradient	The gradient of the steepest part of the pass in degrees, for example 25 degrees

2.1.1 Write code for a constructor method that receives the name of a mountain pass, the distance of the pass in kilometres, the danger level and the gradient of the pass as parameters.

Use the parameters to initialise the attributes of the object class. (2)

2.1.2 Write a mutator method named **setDangerLevel** that updates the **fDangerLevel/dangerLevel** attribute to the danger level received as a parameter. (2)

2.1.3 Write a method named **calculateFine** that receives the time it took the tourist to travel up the mountain pass **in minutes** and the speed limit applicable to this mountain pass in **kilometres per hour** as parameters. The method must calculate the average speed that was travelled and determine whether the speed limit was exceeded. Return a value of zero if the speed limit was NOT exceeded. If the speed limit was exceeded, calculate and return the fine to be paid.

The fine is determined as follows:

- If the average speed exceeds the speed limit by less than 10 km/h, the fine is zero.
- If the average speed exceeds the speed limit by exactly 10 km/hour, then the basic fine is R500,00.
- If the average speed exceeds the speed limit by more than 10 km/h, an additional R100 is added to the basic fine of R500,00 for every 3 km/h or part thereof that the speed limit was exceeded by more than 10 km/h.

NOTE: Average speed in km/h = $\frac{\text{distance in kilometres}}{\text{time } \textbf{in hours}}$

Examples on how to calculate fines are given on the next page.

Example 1:

Speed limit = 80 km/h
Average speed travelled = 96 km/h
Calculating the fine: R500,00 (for exceeding the speed limit by 10 km/h) + (2 x R100,00 for the additional 6 km/h by which the average speed travelled exceeded the 10 km/h range)
Fine = R700,00

Example 2:

Speed limit = 60 km/h
Average speed travelled = 83 km/h
Fine = R1 000,00 (13)

2.1.4 Write a method named **suggestedDangerLevel** that receives the average rainfall on the pass over the previous seven days as a parameter. The method returns a suggested danger level (Low/Medium/High) using the following criteria:

- If the average rainfall is equal to or more than 10 mm and the maximum gradient of the pass exceeds 10 degrees, it is suggested that the present danger level be increased to the next danger level.

  Example: If the danger level is currently 'Low' it will be suggested that the danger level must be raised to 'Medium'.

- If an increase is not necessary or the level is already 'High', the present danger level must be returned as the suggested level. (8)

2.2 **Data and GUI:**

The following data is supplied:

- The road maps of some of the mountain passes have been saved as .jpg files. The format of the file name for the maps is **<Name of the mountain pass>.jpg**.

- A two-dimensional array named **arrRain** contains the rainfall that was measured over a period of seven days for the three mountain passes listed in the GUI.

The following GUI allows the user to select one of three mountain passes (on the next page):

Complete the code for the buttons in the main form unit (Delphi)/GUI class (Java) as described in QUESTION 2.2.1 to QUESTION 2.2.4 below.

2.2.1 **Button [Question 2.2.1]**

Write code to do the following:

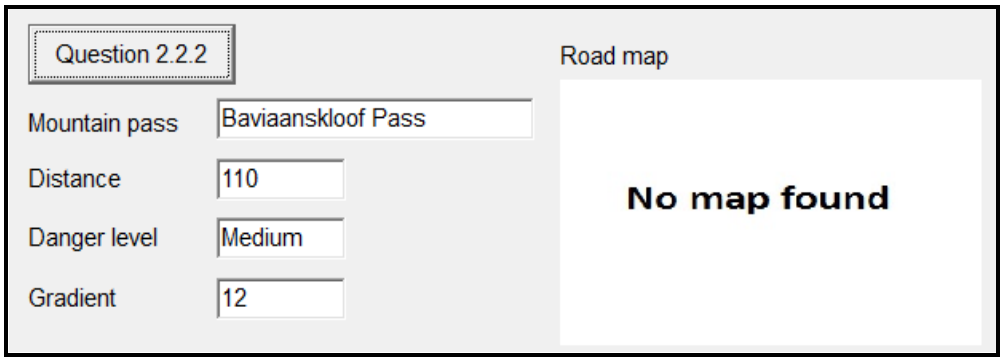
- Determine the selected mountain pass from the provided radio buttons.
- Request the user to enter the distance, danger level and gradient of the selected mountain pass from the keyboard, using dialog boxes.
- Instantiate the **mountain pass** object. A variable for the object has been globally declared.
- Display a message to indicate that the object has been instantiated. (7)

2.2.2 **Button [Question 2.2.2]**

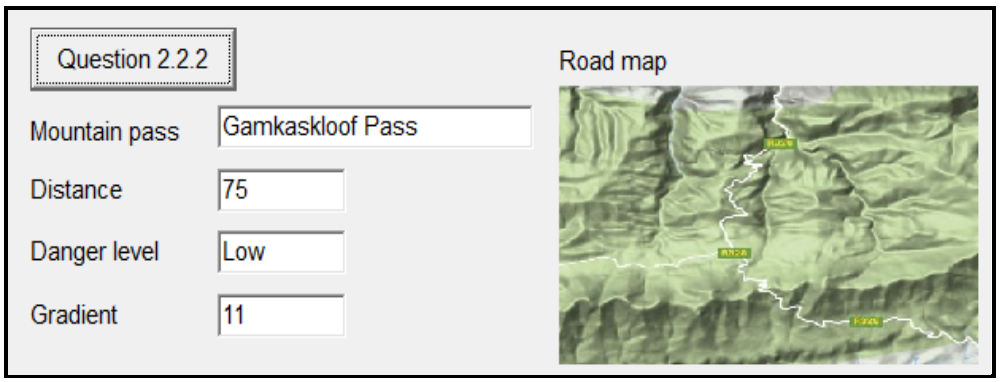
Write code to do the following:

- Use the accessor methods to display the information about the mountain pass that was entered by the user in the relevant text boxes.
- Test whether the road map file for the instantiated object exists. If the map file exists, replace the **<No map found>** image with the content of the map file in the component provided. Otherwise the **<No map found>** image must remain in the display area.

Example of output and the correct image if the user selected Baviaanskloof Pass (see next page):



Example of output and the correct image if the user selected Gamkaskloof Pass:



(8)

2.2.3 **Button [Question 2.2.3]**

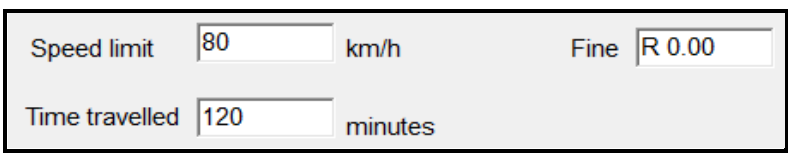
The user is required to enter the speed limit (in km/h) and the time travelled (in minutes) in the text boxes provided.

Write code to do the following:

- Obtain the speed limit and the time travelled from the text boxes. Use these values as arguments and call the **calculateFine** method that must calculate and return the amount the traveller has to pay as a fine.

- Display the amount that should be paid as a fine, formatted as currency, in the output area.

Example of output for travelling over the Prince Alfred Pass with a distance of 88 km and a speed limit of 80 km/h. The time travelled was 120 minutes.



Example of output for travelling over the Gamkaskloof Pass with a distance of 75 km and a speed limit of 40 km/h. The time travelled was 88 minutes.

(6)

2.2.4 **Button [Question 2.2.4]**

A two-dimensional array called **arrRain** contains the rainfall (in millimetres) measured on each of the past 7 days in each of the mountain passes the user could choose from. The row order in the array corresponds to the order of the passes in the supplied radio buttons on the GUI.

Baviaanskloof:	**0, 23, 13, 1, 2, 0, 14**
Gamkaskloof:	**33, 3, 11, 35, 3, 0, 21**
Prince Alfred:	**50, 0, 0, 1, 20, 0, 2**

Write code to do the following:

- Use the **toString** method to display information about the **mountain pass** object in the output area.
- Use the **arrRain** array and calculate the average rainfall for the **mountain pass** object.
- Use the **suggestedDangerLevel** method to obtain the suggested danger level for the pass based on the calculated average rainfall.
- Only if the suggested danger level differs from the present danger level, a dialog box must be used to ask the user whether the danger level should be changed to the suggested level or not. If the user chooses to change the danger level it must be set to the suggested level using the **setDangerLevel** method.
- Display a message in the output area, indicating whether:
    - No change was suggested
    - The suggested change was rejected
    - The danger level was changed

Use the following data to test your program (on the next page):

	TEST DATA 1	TEST DATA 2	TEST DATA 3
Mountain pass	Baviaanskloof	Gamkaskloof	Prince Alfred
Distance in km	110	75	88
Danger level	Medium	Low	Medium
Gradient	12	11	15

NOTE: The message must indicate the danger level in upper case, as shown in the examples below.

Example of output for the Baviaanskloof Pass:

```
Baviaanskloof Pass with a maximum gradient of 12 degrees has a
danger level rating of Medium.
The distance of the pass is 110 km.

No change suggested. Danger level rating remains MEDIUM.
```

Example of output for the Gamkaskloof Pass if suggestion is rejected:

```
Gamkaskloof Pass with a maximum gradient of 11 degrees has a
danger level rating of Low.
The distance of the pass is 75 km.

Suggestion rejected. Danger level rating kept at LOW.
```

Example of output for the Prince Alfred Pass if suggestion is accepted:

```
Prince Alfred Pass with a maximum gradient of 15 degrees has a
danger level rating of Medium.
The distance of the pass is 88 km.

Danger level rating changed to HIGH.
```

(16)

- Enter your examination number as a comment in the first line of both files containing your code.
- Save your program.
- Print the code of both files that contain your code if printouts are required.

TOTAL SECTION B: 62

SECTION C

QUESTION 3: PROBLEM-SOLVING PROGRAMMING

SCENARIO

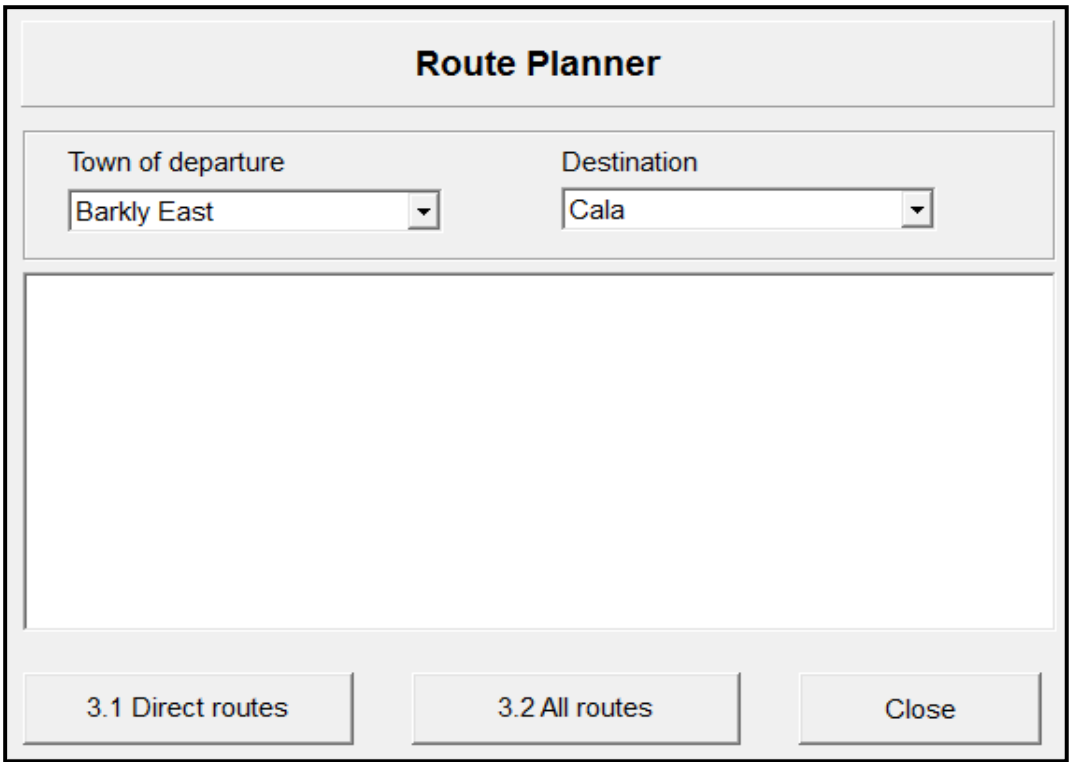
Your dad/guardian is planning a holiday in the Eastern Cape where you can drive through various areas. He wants to determine the possible routes between different towns and whether there are mountain passes on these routes. You need to design the necessary software to assist him in this.

INSTRUCTIONS:

Delphi programmers	Java programmers
• The project **Question3** is provided in the **DelphiDataENGMarch2016** folder. • Open the incomplete project file **Question3P.dpr** in the **Question3** folder. • Enter your examination number as a comment in the first line of the main form unit **Question3U.pas**.	• The project **Question3** is provided in the **JavaDataENGMarch2016** folder. • Open the incomplete class **Question3.java** contained in the **Source Packages (src)**, **Question3Package**. • Enter your examination number as a comment in the first line of the class **Question3.java**.

Supplied GUI:

The supplied GUI contains components for input, an output area and three buttons.



DATA:

The given text file **Q3Data.txt** contains an unknown number of lines of text (which does not exceed 50 entries).

Each line consists of the name of a town of departure, the name of a neighbouring town that can be directly reached from the town of departure, the distance between the two towns and 'Yes' or 'No', indicating whether a mountain pass is included in the route or not. Different characters are used to separate the data contained in one line.

Format of each line of text in the text file:

```
<town1>;<town2>#<distance between the two towns>*<mountain pass>
```

Example of data in the text file:

```
Ugie;Maclear#25*No
Elliot;Cala#28*Yes
Queenstown;Whittlesea#32*Yes
Whittlesea;Cathcart#46*No
Cathcart;Hogsback#49*Yes
Elliot;Ugie#51*Yes
Queenstown;Cathcart#59*No
:
```

The data in the first two lines above can be interpreted as follows:

- The two neighbouring towns are Ugie and Maclear. The distance between the two towns is 25 km and there is no mountain pass along the route.
- The two neighbouring towns are Elliot and Cala. The distance between the two towns is 28 km and there is a mountain pass along the route.

NOTE:

- You are NOT allowed to modify the supplied data manually. Code must be used to manipulate the supplied data according to the requirements.
- The use of good programming techniques and modular design must be applied to the design and coding of your solution.
- You may use appropriate data structures as you see fit.

3.1 **Direct routes:**

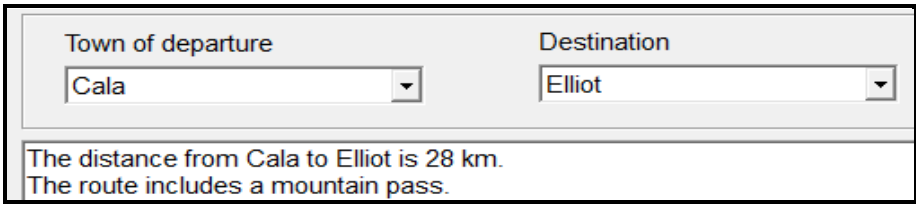
The GUI allows the user to select the town of departure and the destination town from the combo boxes provided.

Use the **Q3Data.txt** text file to determine whether there is a direct route between the two selected towns, in other words, whether they are neighbouring towns or not.

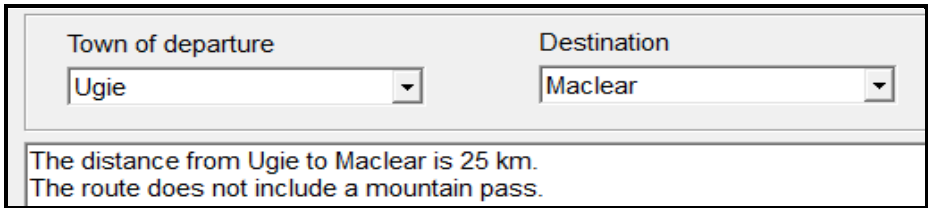
If a direct route exists, display the distance between the two towns and a note to indicate whether there is a mountain pass along the route or not.

NOTE: Direct routes are determined irrespective of the order in which the two neighbouring towns' names appear in the lines of text.

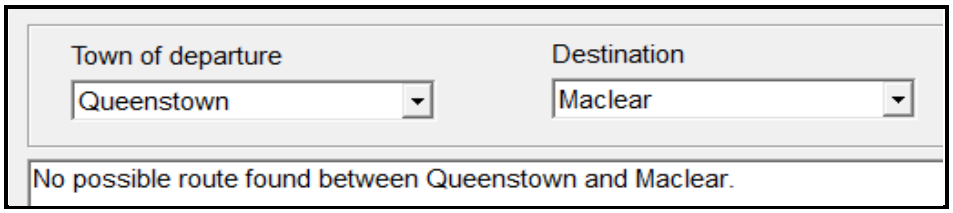
Example of output if a direct route exists between the two selected towns and there is a mountain pass along the route:
(Data in the file: **`Elliot;Cala#28*Yes`**)



Example of output if a direct route exists between the two selected towns and there is no mountain pass along the route:
(Data in the file: **`Ugie;Maclear#25*No`**)



Example of output if there is no direct route between the two selected towns:



(18)

3.2 **List of neighbouring towns:**

The user is required to select only a town of departure from the given combo box. Display all neighbouring town(s) that can be directly reached from the selected town of departure. Also display the distance(s) between the towns.

Use the data supplied in the **Q3Data.txt** text file to find the required towns and distances.

Display the information in column format, sorted in ascending order according to the distances between the towns. A suitable heading must be displayed to indicate the town of departure.

Example of output if Whittlesea is chosen as the town of departure:

Towns that can be directly reached from Whittlesea:	
Queenstown	32 km
Cathcart	46 km

(22)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

TOTAL SECTION C: 40
GRAND TOTAL: 150