basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2021

MARKS: 150

TIME: 3 hours

This question paper consists of 22 pages and 2 data pages.

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into FOUR sections. Candidates must answer ALL the questions in ALL FOUR 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 with programming terms that are specific to Delphi programming language. The Delphi programming language must be used to answer the questions.

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 Delphi for any of these routines.

8. All data structures must be declared by you, the programmer, unless the data structures are supplied.

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. Your examination number must appear on all the printouts. 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 provided to you on the 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: Candidates must use the file **DataENGNov2021.exe**.

Do the following:

- Double click on the following password-protected executable file: **DataENGNov2021.exe**.
- Click on the 'Extract' button.
- Enter the following password: **eHikeNov2021**

Once extracted, the following list of files will be available in the folder **DataENGNov2021**:

Question 1:
Question1_P.dpr
Question1_P.dproj
Question1_P.res
Question1_U.dfm
Question1_U.pas

Question 2:
ConnectDB_U.pas
HikingDB - Copy.mdb
HikingDB.mdb
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question 3:
HikingTrail_U.pas
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas
*.txt 9 files
*.jpg 9 files

Question 4:
Question4_P.dpr
Question4_P.dproj
Question4_P.res
Question4_U.dfm
Question4_U.pas

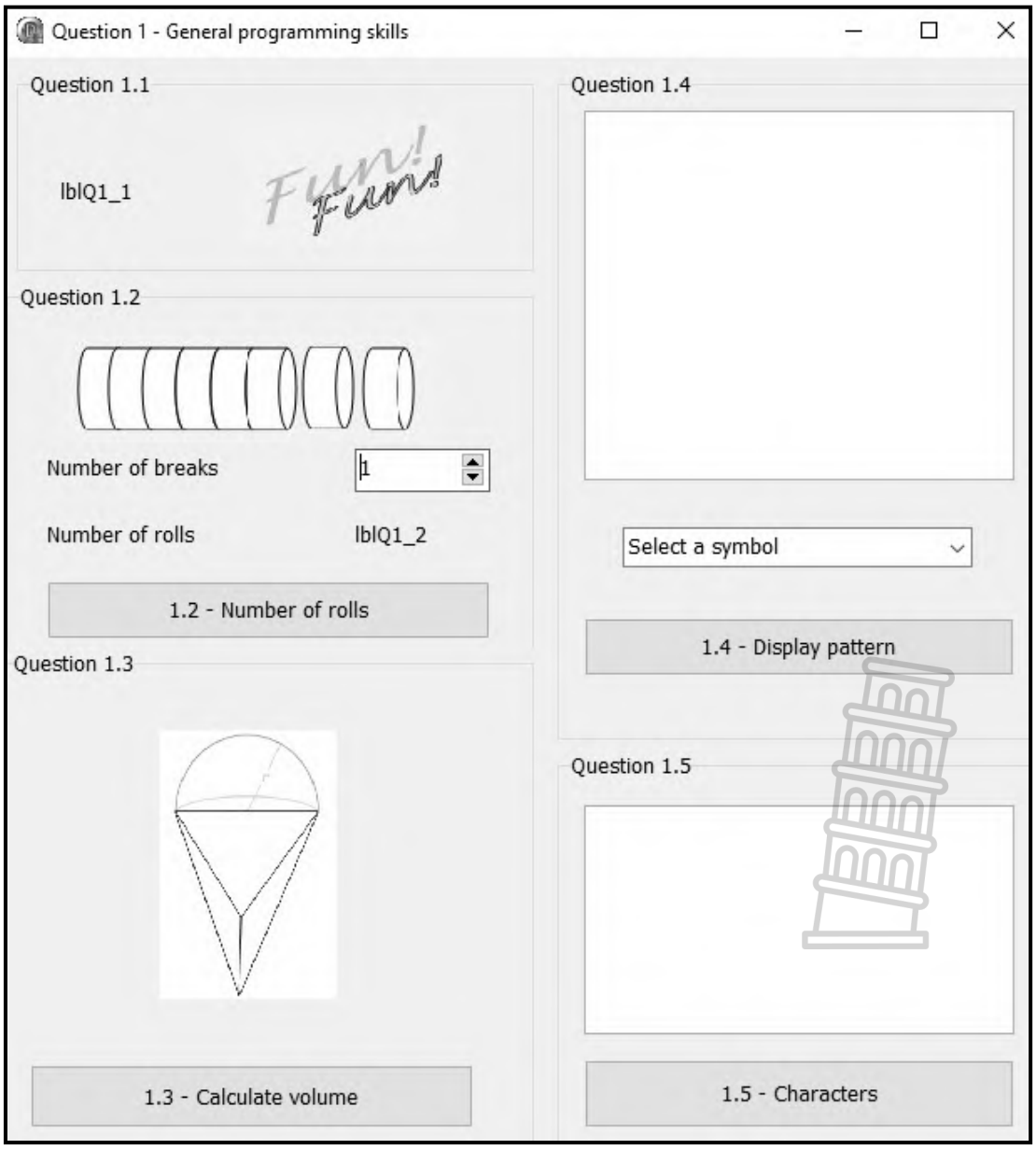
SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your examination number as a comment in the first line of the **Question1_U.pas** file.
- Compile and execute the program. The program has no functionality currently.
- Use the variables provided to answer the questions.

Example of the graphical user interface (GUI):



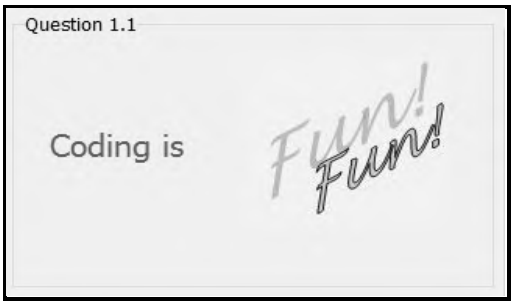
- Complete the code for each section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.5 that follow.

1.1 **FormCreate event**

Write code to change the format of the label **lblQ1_1** as follows:

- Display the text 'Coding is' when the program is executed.
- Change the colour of the text to green (clGreen).
- Change the size of the font to 16 pt.
- Change the font to 'Arial'.

Example of output:



(4)

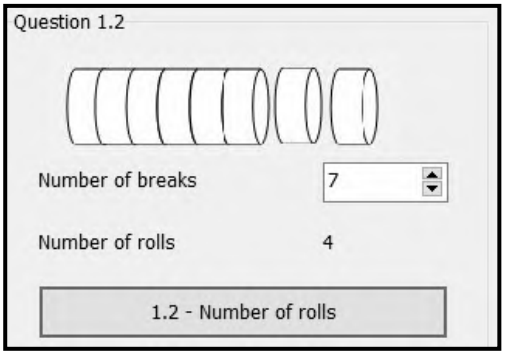
1.2 **Button [1.2 - Number of rolls]**

Energy sweets recommended for hikers to have during each break while hiking are packaged in rolls of eight sweets per roll. Hikers normally eat four sweets during each break. Use the **spnQ1_2** spin edit to select/enter the number of breaks a hiker plans on taking. The program must determine the number of sweet rolls that must be purchased for the hike.

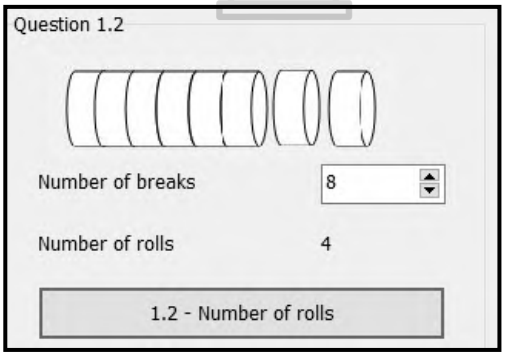
Write code to do the following:

- Create a constant SWEETS_PER_ROLL to store the value of 8.
- Declare suitable integer variables for the number of breaks, the total number of sweets and the number of rolls to purchase.
- Extract the number of breaks to be taken during a hike from the **spnQ1_2** spin edit.
- Calculate the total number of sweets that a hiker will have during the hike. Assume that hikers will consume four sweets during each break.
- Calculate the minimum number of rolls that need to be purchased to supply the hiker with enough sweets for the hike.
- Display the number of rolls on the **lblQ1_2** label.

Example of output for seven breaks:



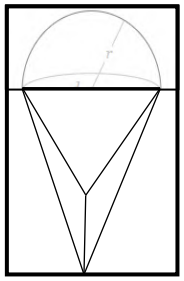
Example of output for eight breaks:



(8)

1.3 **Button [1.3 - Calculate volume]**

The figure shown below was compiled using a combination of half a sphere and a tetrahedron (a solid shape with four flat sides that are triangles). The program must calculate and display the volume of the combined figure.

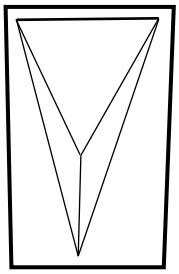


The following information is provided:

- The formula to calculate the volume of the combined figure is:

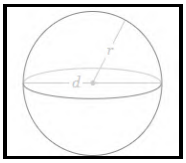
$$V = volume\ of\ tetrahedron + \frac{1}{2}(volume\ of\ sphere)$$

- The volume of the tetrahedron in the figure is given as 133 cm^3.



- The formula to calculate the volume of a sphere is:

$$V = \frac{4}{3}\pi r^3$$



Use the information provided and write code to do the following:

- Calculate the volume of the combined figure if the radius of the sphere is 3 cm.
- Display the volume in a ShowMessage box formatted to ONE decimal place.

Example of output:

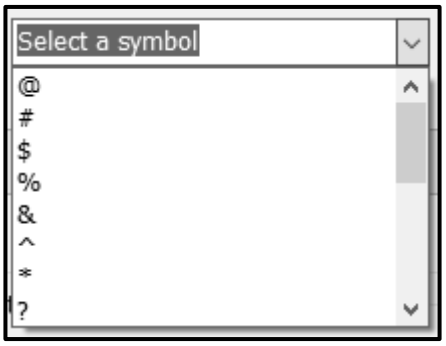
(7)

1.4 **Button [1.4 - Display pattern]**

A pattern with a symbol selected from the **cmbQ1_4** combo box must be displayed. The pattern is a square which consists of the same number of rows and columns. The number of rows and columns is obtained from the position of the symbol that has been selected from the combo box.

NOTE: The first item (@) in the combo box is situated in position one.

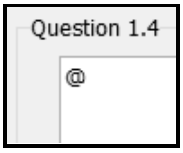
The first eight symbols in the combo box are shown below.



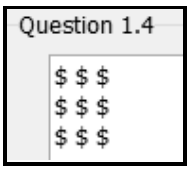
Write code to do the following:

- Clear the output area **redQ1_4**.
- Extract the symbol selected by the user from the **cmbQ1_4** combo box.
- Obtain the position of the symbol selected in the combo box to determine the number of rows and columns.
- Use nested loop structures to display the pattern (block) in the rich edit, using the selected symbol.

Example of output if the first item (@) is selected in **cmbQ1_4**:



Example of output if the third item ($) is selected in **cmbQ1_4**:



(11)

1.5 **Button [1.5 - Characters]**

A string must be compiled by randomly generating upper-case letters of the alphabet until the same letter is generated consecutively.

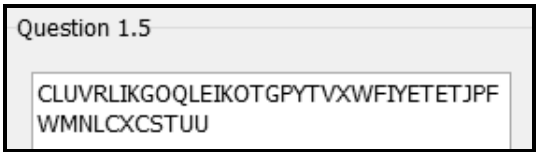
Code has been provided to clear the rich edit and to initialise an output string.

Write code to do the following:

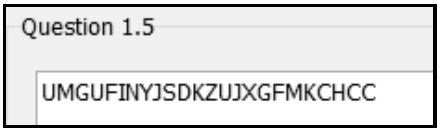
- Randomly generate a capital letter from the alphabet in the range 'A'..'Z'.
- Compile a string using the generated letters.
- Repeat the process until the generated letter is a duplicate of the previous (last) letter that was generated.
- Display the generated string in the **redQ1_5** rich edit. Each time the output string must include the final duplicate characters, as shown in the example output.

HINT: The ASCII values for the letters 'A' to 'Z' start at 65 for the letter 'A', 66 for the letter 'B', and so on up to the value of 90 for the letter 'Z'.

Example of output:



In the example above, the process was terminated because the letter 'U' was generated consecutively.



In the example above, the process was terminated because the letter 'C' was generated consecutively.

NOTE: The output your program generates may differ from the output examples, because the letters are randomly generated. The string that is generated can contain less or more characters than the strings supplied in the examples. (10)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION A: 40

SECTION B

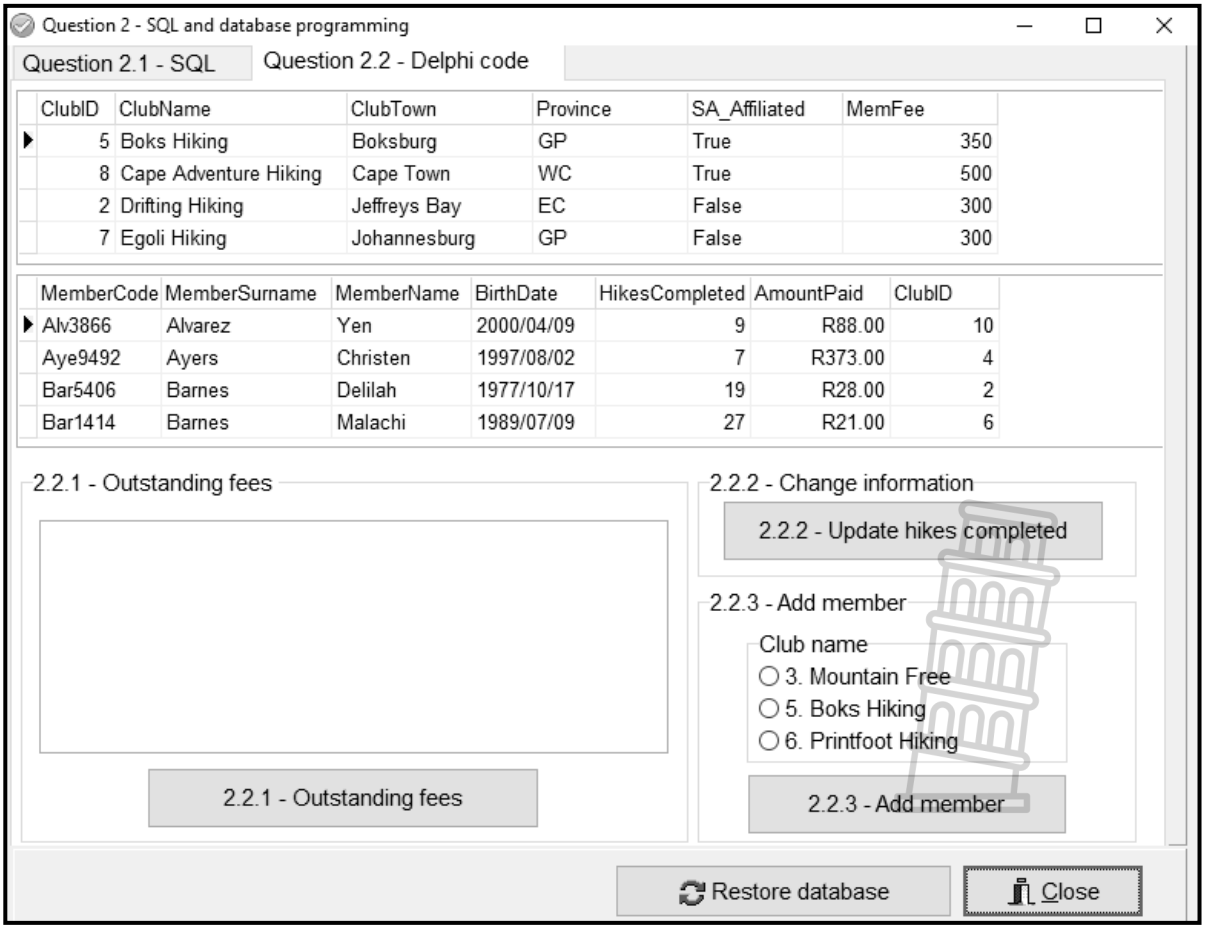
QUESTION 2: SQL AND DATABASE PROGRAMMING

The database **HikingDB** contains information of members of different hiking clubs. The database contains two tables, namely **tblClubs** and **tblMembers**.

The data pages attached at the end of the question paper provide information on the design of the database and its contents.

Do the following:

- Open the incomplete project file called **Question2_P.dpr** in the **Question 2** folder.
- Enter your examination number as a comment in the first line of the **Question2_U.pas** unit file.
- Compile and execute the program. The program has no functionality currently. The contents of the tables are displayed as shown below on the selection of tab sheet **Question 2.2 - Delphi code**.



- Follow the instructions that follow to complete the code for each section as described in QUESTION 2.1 and QUESTION 2.2.

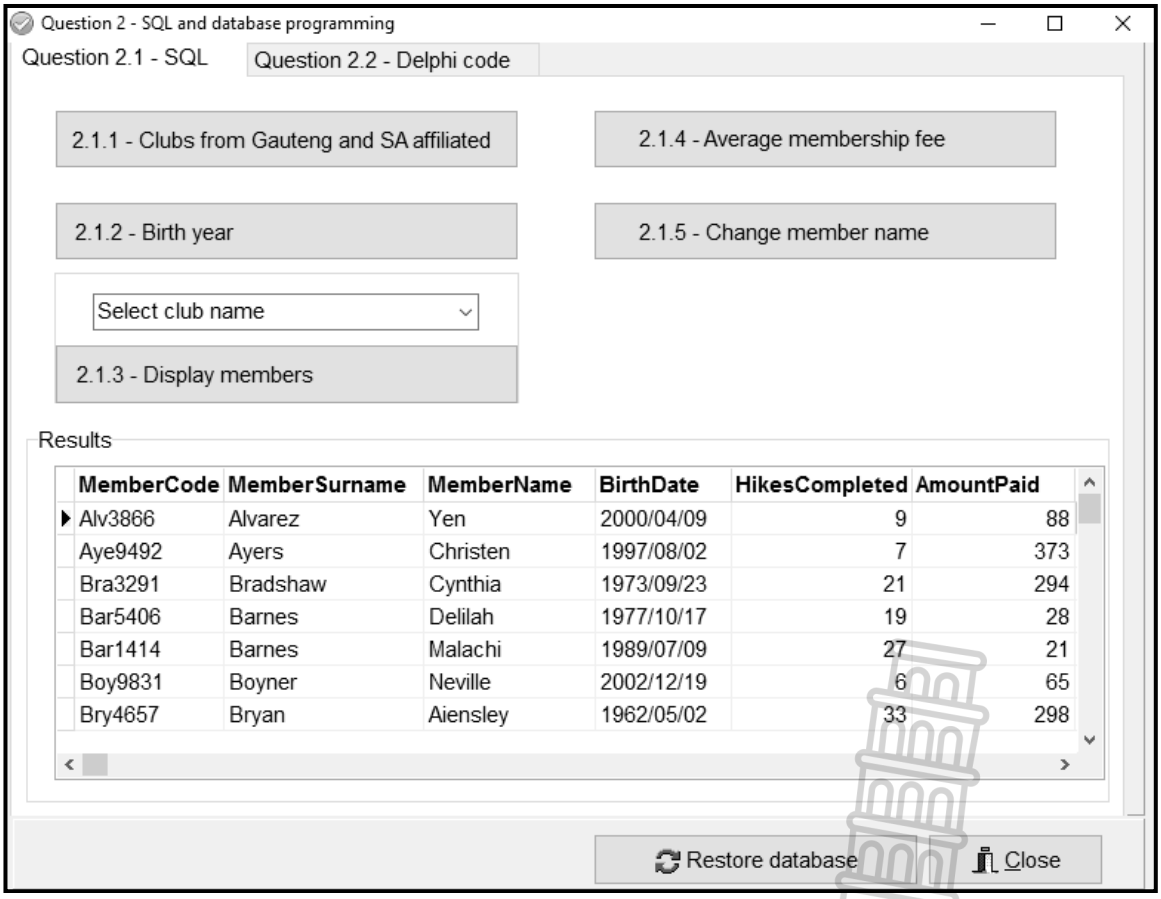
NOTE:

- The 'Restore database' button is provided to restore the data contained in the database to the original content.
- Code is provided to link the GUI components to the database. Do NOT change any of the code provided.
- TWO variables are declared as public variables as described in the table below.

Variable	Data type	Description
`tblClubs`	`TADOTable`	Refers to the table tblClubs
`tblMembers`	`TADOTable`	Refers to the table tblMembers

2.1 Tab sheet [Question 2.1 - SQL]

Example of graphical user interface (GUI) for QUESTION 2.1:



NOTE: Code to execute the SQL statements and display the results of the queries is provided. The SQL statements assigned to the variables **sSQL1**, **sSQL2**, **sSQL3**, **sSQL4** and **sSQL5** are incomplete.

Use ONLY SQL code to complete the SQL statements for QUESTION 2.1.1 to QUESTION 2.1.5 that follow.

2.1.1 **Button [2.1.1 - Clubs from Gauteng and SA affiliated]**

Display the names and the towns of all clubs in Gauteng (GP) that are SA affiliated.

Example of output:

ClubName	ClubTown
Boks Hiking	Boksburg
Printfoot Hiking	Pretoria

(3)

2.1.2 **Button [2.1.2 - Birth year]**

Display the name, surname and date of birth of all members who were born in 2002.

Example of output:

MemberName	MemberSurname	BirthDate
Neville	Boyner	2002/12/19
Rahim	Heath	2002/09/25
Jin	Lucas	2002/08/03
Carl	Monroe	2002/09/20
Hyatt	Whitney	2002/06/13

NOTE: The date format may differ from the example in the output. (3)

2.1.3 **Button [2.1.3 - Display members]**

Code has been provided to select a **ClubName** from the **cmbQ2_1_3** combo box.

Display the surname and name of members from the club selected in the combo box.

Example of output if Kamma Hiking is selected:

MemberSurname	MemberName
Cohran	Raymond
Kelly	Blossom
Monroe	Carl

(4)

2.1.4 **Button [2.1.4 - Average membership fee]**

Display the province and the average membership fee per province where the average membership fee is more than R400.

The average membership fee per province must be a calculated field with the heading **AvgFee** and it must be displayed as currency.

Example of output:

Province	AvgFee
KZN	R475.00
WC	R425.00

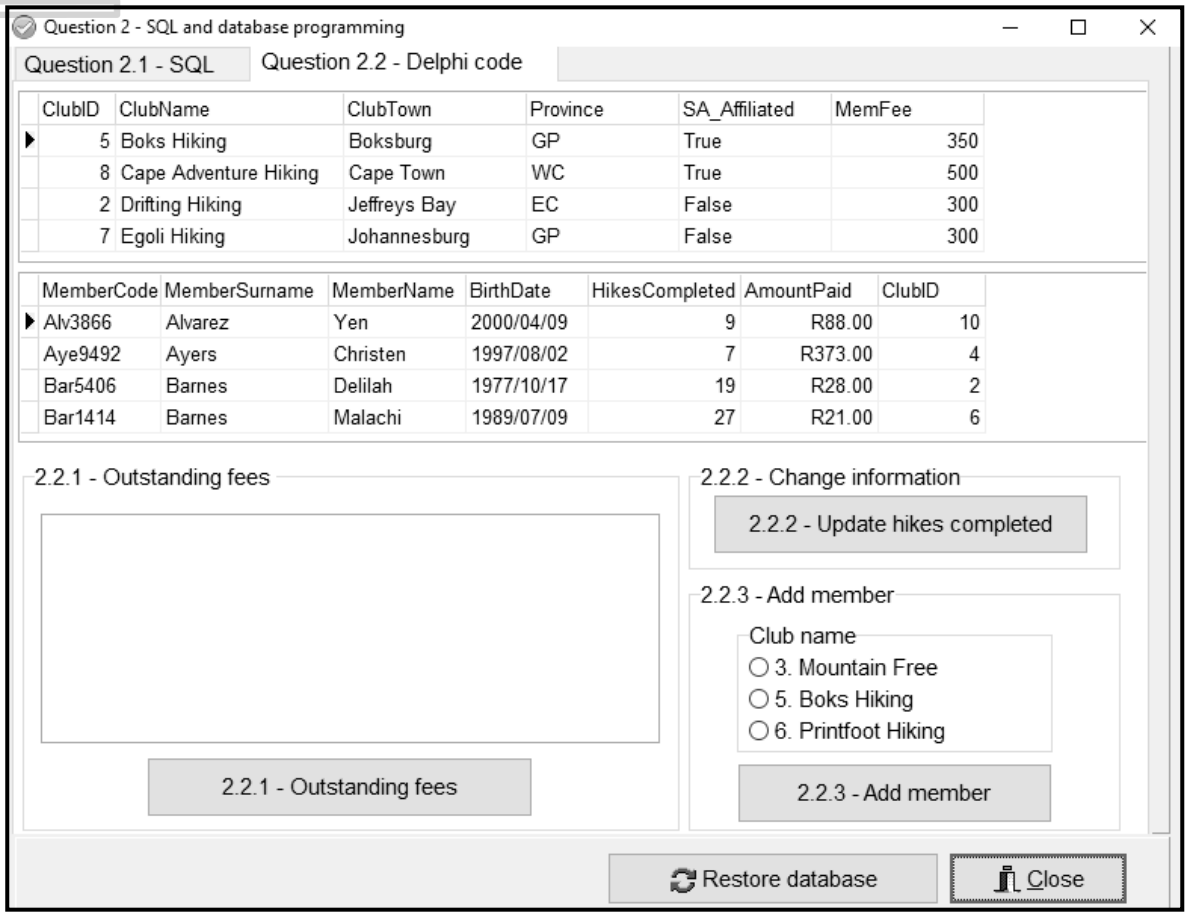
(6)

2.1.5 **Button [2.1.5 - Change member name]**

Write code to change all member names spelled as 'Aiensley' to 'Ainsley'. (3)

2.2 **Tab sheet [Question 2.2 - Delphi code]**

Example of graphical user interface (GUI) for QUESTION 2.2:



NOTE:

- Use ONLY Delphi programming code to answer QUESTION 2.2.1 to QUESTION 2.2.3.
- NO marks will be awarded for SQL statements in QUESTION 2.2.

2.2.1 **Button [2.2.1 - Outstanding fees]**

Write code to display the club name and annual membership fee as part of a heading and the surname, amount paid and outstanding fees for each club member in the **redQ2_2_1** rich edit.

Example of output for the first two clubs:

```
Boks Hiking, annual fee = R350.00
=====================================
Surname         Paid          Outstanding
Oliver          R219.00       R131.00
Pearce          R70.00        R280.00

Cape Adventure Hiking, annual fee = R500.00
=====================================
Surname         Paid          Outstanding
Dunn            R33.00        R467.00
English         R481.00       R19.00
Guzman          R334.00       R166.00
Heath           R298.00       R202.00
Hubbard         R326.00       R174.00
Ward            R235.00       R265.00
```

(12)

2.2.2 **Button [2.2.2 - Update hikes completed]**

Select a member from the **tblMembers** table in the dbgrid (dbgrdMany) who completed another hike.

Write code to increment the **HikesCompleted** field, by one for the member selected in the dbgrid. (4)

2.2.3 **Button [2.2.3 - Add member]**

Only the following clubs allow new member to join:

Mountain Free
Boks Hiking
Printfoot Hiking

The IDs and names of these clubs are listed in the **rgpQ2_2_3** component in the following format: <ClubID>**.** <ClubName> with the caption 'Clubname'.

Code has been provided to assign the surname, name, date of birth and membership code of a new member to variables.

NOTE: To add a new member, the user must first select a club name from the **rgpQ2_2_3** radio group.

Write code to do the following:

- Extract the **ClubID** from the club selected in the **rgpQ2_2_3** radio group.
- Assign the provided variables to the corresponding member fields and the **ClubID** to the **ClubID** member field.
- Add the required statement(s) to ensure that the new member's details are saved in the database table.

NOTE: You can add the member ONCE only. If you want to test your code by adding the member again, first restore the database to its original content before testing the code again. (5)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION B: 40

SECTION C

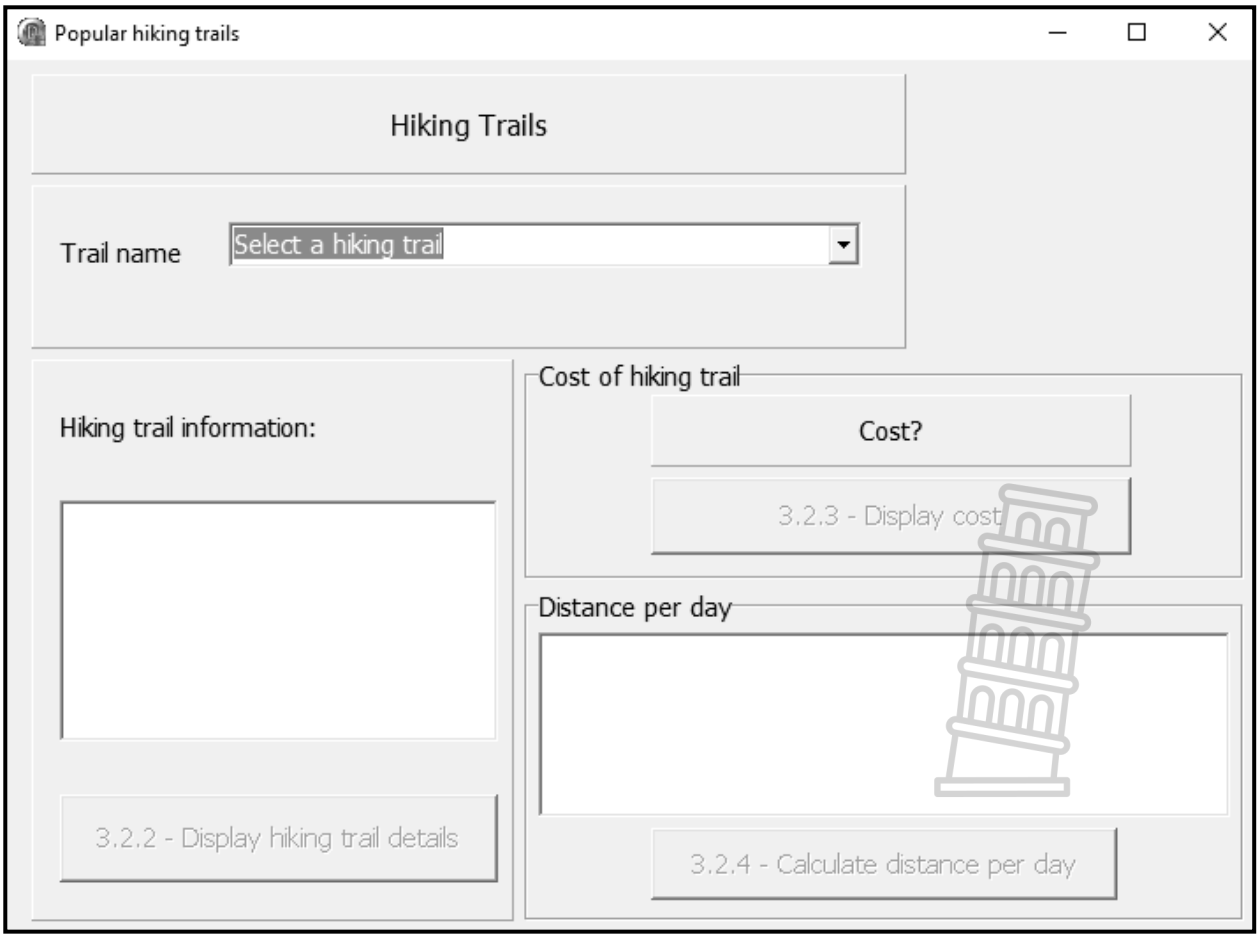
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

Information on some of the popular hiking trails in South Africa are available in the format of text files. The incomplete program provided must use the information in the text files to determine whether prospective hikers are fit enough to attempt a specific hiking trail. The program also needs to calculate the cost to book the specific hiking trial.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Open the incomplete object class **HikingTrail_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question3_U.pas** file and the **HikingTrail_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

Example of graphical user interface (GUI):



- Complete the code as specified in QUESTION 3.1 and QUESTION 3.2.

NOTE: For this question, you are NOT allowed to include any additional attributes or user-defined methods unless explicitly stated in the question.

3.1 The incomplete object class (**THikingTrail**) provided contains code for the declaration of five attributes that describe a **HikingTrail** object.

The attributes for a **HikingTrail** object have been declared as follows:

Attribute	Description
fTrailName	The name of the hiking trail
fTerrainType	The type of terrain, e.g. Rocky
fNumberOfDays	The number of days it takes to complete the trail
fDistance	The total distance of the hiking trail in kilometres
fCostPP	The cost to book the hiking trail per person

A **constructor** that assigns parameter values to the attributes has been provided.

Complete the code in the object class as described in QUESTION 3.1.1 to QUESTION 3.1.5 below.

3.1.1 Write an accessor method called **getNumberOfDays** for the fNumberOfDays attribute. (2)

3.1.2 Write a method **calcDistPerDay** to calculate and return the average number of kilometres that must be completed per day to complete the trail within the required number of days, rounded off to the nearest integer. (3)

3.1.3 Write a method called **determineLevel** to determine and return the difficulty level of the trail as a string. The difficulty level can be advanced, moderate or easy.

Write code to do the following:

- Call the **calcDistPerDay** method to calculate and return the distance that must be completed per day.
- Use the distance per day value to determine the difficulty level as follows:
  - 'Advanced' if the distance per day is more than 15 km and the terrain is 'Rocky' or 'Sandy'.
  - 'Moderate' if the distance per day is from 10 km to 15 km.
  - 'Easy' if the difficulty level is neither 'Advanced' nor 'Moderate'. (10)

3.1.4 Write a method called **calcTotalCost** to receive the number of hikers in a group as a parameter and return the total cost for the group, based on the fCostPP attribute value. (4)

3.1.5 Write a **toString** method to return all the attributes of the object in the following format:

```
<Name of hiking trail>: <Type of terrain>
<distance> km in <number of days> days
Cost per person: <Cost> formatted to currency
```

Example:

```
Amatola Hiking Trail: Rocky
100 km in 6 days
Cost per person: R1500.00
```

(4)

3.2 An incomplete program has been supplied in the **Question 3** folder. The program contains code for the object class to be accessible and declares an object variable called **objHikingTrail**.

Write code to perform the tasks described in QUESTION 3.2.1 to QUESTION 3.2.4.

3.2.1 **Combo box - cmbQ3_2_1**

Information on hiking trails are provided in different text files.

The name of the hiking trail, as provided in the combo box **cmbQ3_2_1**, is the same as the name of the text file. The extension of all text files is '.txt'.

Each text file contains four lines of information in the following format:

```
Type of terrain
Total distance in km
Total number of days
Cost of booking the trail for one person
```

For example, the content of the text file **Amatola Hiking Trail** is:

```
Rocky
100
6
1500
```

Code has been provided to extract the selected hiking trail from the combo box **cmbQ3_2_1** and to display an image of the hiking trail in the image component.

Each time the name of the hiking trail is selected in the combo box, a new object must be instantiated.

Write code to do the following:

- Open the correct text file to read from using the name of the selected hiking trail.
- Read the terrain, distance, number of days and the cost per person from the text file.
- Use this information and the name of the hiking trail to instantiate a new hiking trail object.
- Display a message using a dialogue box to indicate that the object has been instantiated successfully.

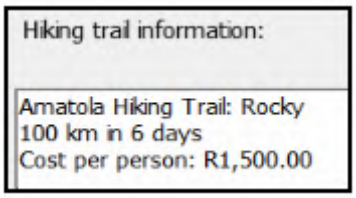
Example of output:

(8)

3.2.2 **Button [3.2.2 - Display hiking trail details]**

Write code to use the **toString** method to display the object's information in the **redQ3_2_2** component.

Example of output if the Amatola Hiking Trail is selected:



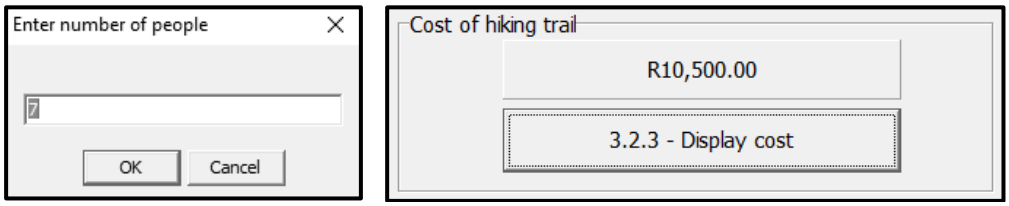
(2)

3.2.3 **Button [3.2.3 - Display cost]**

Write code to do the following:

- Use an input box to enter the number of hikers in the group.
- Use the **calculateCost** method to assign the cost for the group to the provided cost variable.

Example of output if the Amatola Hiking Trail is selected and the size of the group is 7:



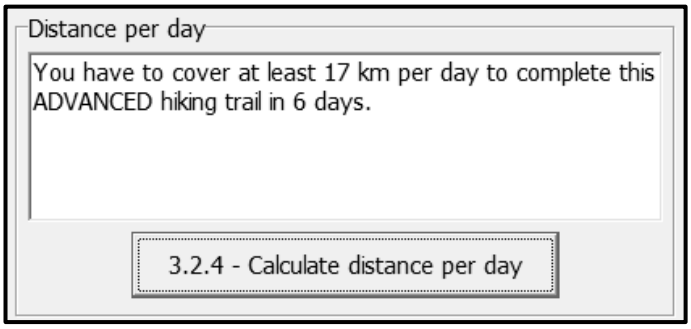
(4)

3.2.4 **Button [3.2.4 – Calculate distance per day]**

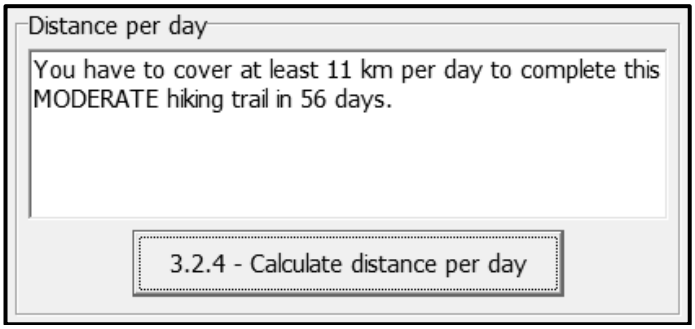
Write code to use the object's methods and display the difficulty level and recommended distance per day in the following format for the selected hiking trail in the **redQ3_2_4** rich edit:

```
You have to cover at least <distance per day> km
per day to complete this <difficulty level> hiking
trail in <number of days> days.
```

Example of output if the Amatola Hiking Trail is selected:



Example of output if the Rim of Africa hiking trail is selected:



(3)

- Enter your examination number as a comment in the first line of the object class and the form class.
- Save your program.
- Print the code of the object class and the form class if required.

TOTAL SECTION C: 40

SECTION D

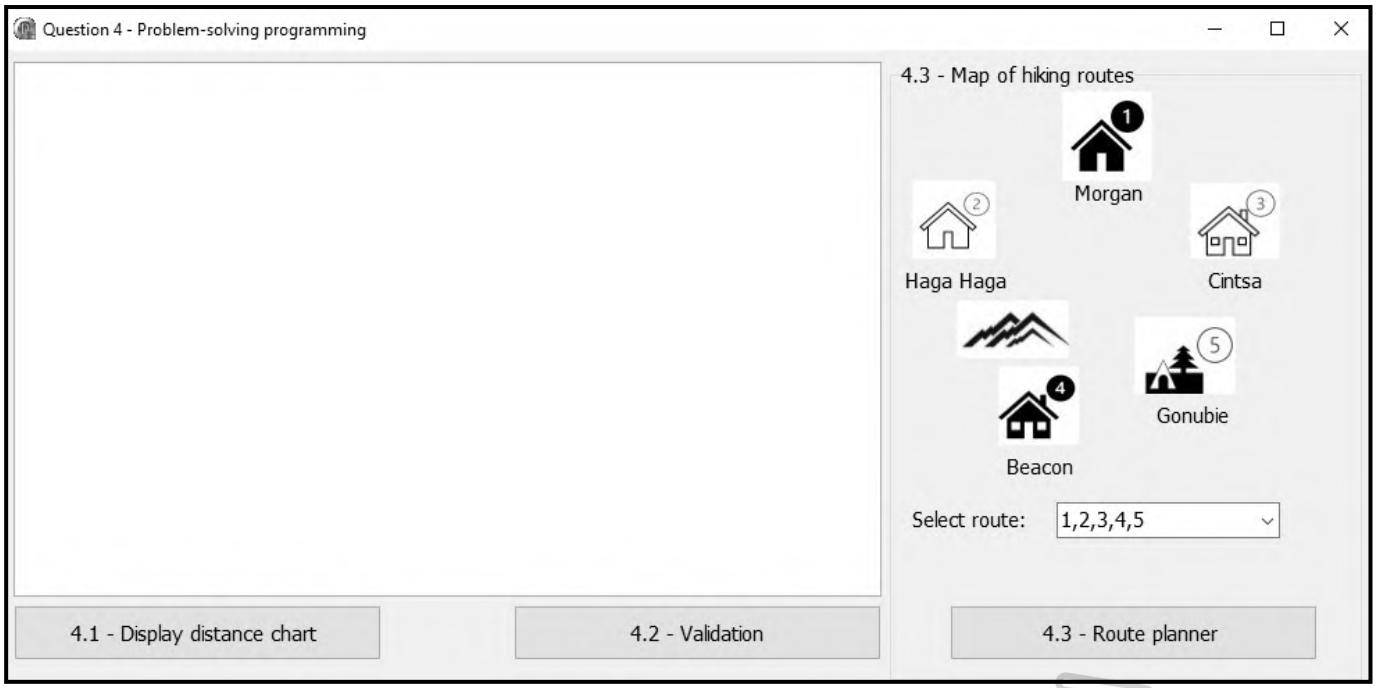
QUESTION 4: PROBLEM-SOLVING PROGRAMMING

Hikers use a distance chart to plan their hiking trips. Checkpoints refer to rest areas or overnight locations on a hiking trail. A distance chart consists of the distances between different checkpoints on a hiking trail.

Do the following:

- Open the incomplete program in the **Question 4** folder.
- Enter your examination number as a comment in the first line of the **Question4_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

Example of the graphical user interface (GUI):



The following have been provided in the program:

- The names of checkpoints are stored in a one-dimensional array named **arrNames**:

```
arrNames: array[1..5] of String =
          ('Morgan', 'Haga Haga', 'Cintsa', 'Beacon', 'Gonubie');
```

- The distances between the checkpoints on the trail are stored in a two-dimensional array named **arrDistances** in number of kilometres:

```
arrDistances: array[1..5,1..5] of real =
              ((0, 6.2, 9, 13.2, 16.2),
              (6.2, 0, 13.37, 8.74, 12.86),
              (9.57, 13.37, 0, 9.63, 24.63),
              (13.2, 8.74, 9.63, 0, 15.2),
              (16.2, 12.86, 24.63, 15, 0));
```

- Code has been provided to display the headings for QUESTION 4.1 and QUESTION 4.2.

- Write code to perform the tasks described in QUESTION 4.1 to QUESTION 4.3.

4.1 **Button [4.1 - Display distance chart]**

Write code to display the names of the checkpoints from the array, **arrNames**, and the contents of the two dimensional array, **arrDistances**, in neat columns in the rich edit **redQ4** provided.

NOTE: The intersection between two of the same checkpoints has a value of 0.

Example of output:

NOTE: Tabs and blank lines in the example output are included for readability and are NOT required as part of the output of the program.

	Morgan	Haga Haga	Cintsa	Beacon	Gonubie
Morgan	0	6.2	9	13.2	16.2
Haga Haga	6.2	0	13.37	8.74	12.86
Cintsa	9.57	13.37	0	9.63	24.63
Beacon	13.2	8.74	9.63	0	15.2
Gonubie	16.2	12.86	24.63	15	0

(5)

4.2 **Button [4.2 - Validation]**

Some of the values in the distance chart in the array **arrDistances** were rounded down to an integer value. These inaccurate integer values need to be replaced by the more accurate decimal value in the corresponding row and column.

Write code to do the following:

- Check if the corresponding row and column in the **arrDistances** array are exactly the same.

  For example:

  The value at arrDistance[1,3] must be the same as the value at arrDistance[3,1], the value at arrDistance[4,2] must be the same as the value at arrDistance[2,4] and so on.

- For each inaccurate value that is identified, replace the inaccurate integer value with the accurate corresponding decimal value.

- Display the position of the inaccurate value(s) that were identified as a row and column number followed by the accurate decimal value that will replace the inaccurate integer value in the following format:

  [row value, column value] with <accurate value>

Example of output:

```
Replace distance at:
[1,3] with 9.57
[5,4] with 15.2
```

Example of the updated two-dimensional array:

	Morgan	Haga Haga	Cintsa	Beacon	Gonubie
Morgan	0	6.2	9.57	13.2	16.2
Haga Haga	6.2	0	13.37	8.74	12.86
Cintsa	9.57	13.37	0	9.63	24.63
Beacon	13.2	8.74	9.63	0	15.2
Gonubie	16.2	12.86	24.63	15.2	0

(8)

4.3 Button [4.3 - Route planner]

Hikers can select one of different routes listed in the **cmbRoutes** combo box to do a hiking trail.

The hiker must select a possible route in the combo box.

If the hiker selects the route '4,3,1,2,5', it means that the hiking trail starts at the Beacon checkpoint. The route then continues towards the Cintsa, Morgan and Haga Haga checkpoints with Gonubie being the last checkpoint.

An itinerary needs to be compiled and displayed for the route selected.

Write code to do the following:

- Extract the route from the **cmbRoutes** combo box and display the selected route in the **redQ4** rich edit.
- Use the information in the **arrNames** array to display the names of each pair of checkpoints and the array **arrDistances** to display the distance between each pair of checkpoints.
- Determine and display the approximate time (in minutes) it will take to complete the hike between the pairs of checkpoints. Use the following information:
  - It takes 20 minutes to walk a distance of 1 km.
  - It takes double this time to walk the same distance on the rocky route between Haga Haga (checkpoint 2) and Beacon (checkpoint 4).
- If the total time spent hiking for one day is more than 8 hours (480 minutes), the hiker must book accommodation to stay overnight at the last checkpoint for that day. The name(s) of the overnight checkpoint(s) must be displayed.
- Calculate and display the total distance of the hike in kilometres.

Example of output if the selected route is 4,3,1,2,5:

```
Route: 4,3,1,2,5

Beacon to Cintsa: 9.63 (192.6 minutes)
Cintsa to Morgan: 9.57 (191.4 minutes)
Morgan to Haga Haga: 6.2 (124 minutes)
Book at Haga Haga.

Haga Haga to Gonubie: 12.86 (257.2 minutes)

Total distance: 38.3 km
```

Example of output if the selected route is 2,4,3,5,1:

```
Route: 2,4,3,5,1

Haga Haga to Beacon: 8.74 (349.6 minutes)
Beacon to Cintsa: 9.63 (192.6 minutes)
Book at Cintsa.

Cintsa to Gonubie: 24.63 (492.6 minutes)
Book at Gonubie.

Gonubie to Morgan: 16.2 (324 minutes)

Total distance: 59.2 km
```

(17)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION D: 30
GRAND TOTAL: 150

INFORMATION TECHNOLOGY P1 (2)

DATABASE INFORMATION QUESTION 2:

The design of the database tables is as follows:

Table: **tblClubs**

The table contains the information of hiking clubs in different provinces.

Field name	Data type	Description
ClubID (PK)	Number	A unique number assigned to the club
ClubName	Text (20)	The name of the hiking club
ClubTown	Text (25)	The town where the club is situated
Province	Text (20)	The province where the club is situated
SA_Affiliated	Boolean	Boolean field indicating whether the hiking club is affiliated with the SA hiking club or not
MemFee	Currency	Annual membership fee

Example of the first ten records in the **tblClubs** table:

tblClubs

ClubID	ClubName	ClubTown	Province	SA_Affiliated	MemFee
1	Wild Boast Hiking	Cradock	EC	☑	R500.00
2	Drifting Hiking	Jeffreys Bay	EC	☐	R300.00
3	Mountain Free	Bergville	KZN	☑	R450.00
4	Velo Wildlife Hiking	Winterton	KZN	☑	R500.00
5	Boks Hiking	Boksburg	GP	☑	R350.00
6	Printfoot Hiking	Pretoria	GP	☑	R500.00
7	Egoli Hiking	Johannesburg	GP	☐	R300.00
8	Cape Adventure Hiking	Cape Town	WC	☑	R500.00
9	Richway Ramblers	Cape Town	WC	☑	R450.00
10	Vent Int Hiking	Cape Town	WC	☑	R300.00
11	Walk-a-Hike	Cape Town	WC	☑	R400.00
12	Garden Trails	Knysna	WC	☐	R400.00
13	Kamma Hiking	Tsitstikamma	WC	☐	R500.00

Table: **tblMembers**

This table contains the information of members in different hiking clubs.

Field name	Data type	Description
MemberCode (PK)	Text (7)	A unique code assigned to each member
MemberSurname	Text (20)	The surname of the member
MemberName	Text (20)	The name of the member
BirthDate	Date/Time	The birthday of the member
HikesCompleted	Number	The total number of hikes completed by the member
AmountPaid	Currency	The amount paid by the member towards the membership fee
ClubID (FK)	Number	A number that connects the member to the club where the member is registered

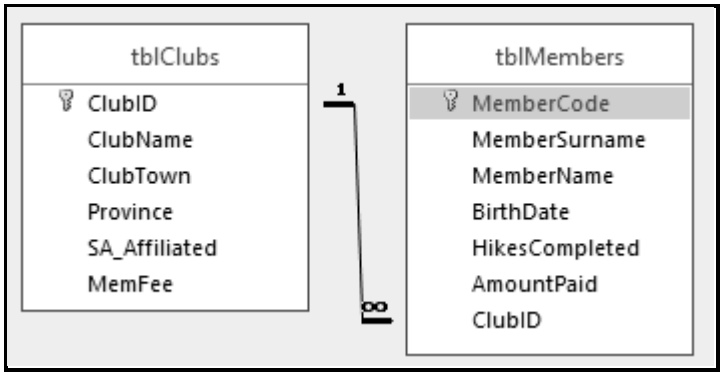
Example of the first ten records in the **tblMembers** table:

tblMembers

MemberCode	MemberSurname	MemberName	BirthDate	HikesCompleted	AmountPaid	ClubID
Alv3866	Alvarez	Yen	2000/04/09	9	R88.00	10
Aye9492	Ayers	Christen	1997/08/02	7	R373.00	4
Bar1414	Barnes	Malachi	1989/07/09	27	R21.00	6
Bar5406	Barnes	Delilah	1977/10/17	19	R28.00	2
Boy9831	Boyner	Neville	2002/12/19	6	R65.00	4
Bra3291	Bradshaw	Cynthia	1973/09/23	21	R294.00	3
Bry4657	Bryan	Aiensley	1962/05/02	33	R298.00	1
Bus2297	Bush	Cameron	2001/08/26	11	R417.00	1
Cal5273	Calderon	Aidan	1974/11/11	31	R294.00	2
Coh5809	Cohran	Raymond	1989/06/02	47	R220.00	13

NOTE: Connection code has been provided.

The following one-to-many relationship with referential integrity exists between the two tables in the database:



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2021

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 23 pages.

GENERAL INFORMATION:

- These marking guidelines are to be used as the basis for the marking session. They were prepared for use by markers. All markers are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' work.

- Note that learners who provide an alternate correct solution to that given as example of a solution in the marking guidelines will be given full credit for the relevant solution, unless the specific instructions in the paper was not followed or the requirements of the question was not met

- **Annexures A, B, C** and **D** (pages 3 to 10) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 11 to 22) contain examples of solutions for Questions 1 to 4 in programming code.

- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 10) should be made for each learner and completed during the marking session.

ANNEXURE A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
1.1	**FormCreate event** Set caption of lblQ1_1 to 'Coding is ' ✓ Set font colour of lblQ1_1 to green ✓ Set font size of lblQ1_1 to 16 ✓ Set font name to 'Arial' ✓	**4**	
1.2	**Button - [1.2 – Number of rolls]** Create constant SWEETS_PER_ROLL = 8 Declare integer variables ✓ Extract number of breaks from spin edit ✓ Calculate total number of sweets (breaks x 4) ✓ Calculate total number of rolls (sweets✓/SWEETS _PER_ROLL✓) //OR (sweets/8) Round up using the Ceil function ✓ Display number of rolls of sweets on lblQ1_2 ✓ converted to String ✓	**8**	
1.3	**Button – [1.3 – Calculate volume]** rRadius := 3; rVolume := rTetraVolume ✓ + 4/3 * PI ✓ * power (rRadius,3) ✓/2 ✓ Display Using a ShowMessage with “Volume” label✓ Value of volume ✓ Formatted to one decimal place ✓ **Note:** In the formula: Accept the value of 133 instead of rTetraVolume Accept the value of 3 instead of rRadius Instead of PI, accept 22/7 or 3.14 Instead of 4/3, accept 1.33 Accept any correct alternative to power(rRadius,3) where the answer = 27 Accept * 0.5 instead of /2	**7**	

1.4	**Button - [1.4 – Display pattern]** Clear output area ✓ Extract symbol from combo box ✓ iSize = position of symbol in combo box ✓ //(index+1) Loop✓ rows from 0 ✓ to iSize ✓ // Loop from 1 to iSize Initialise output String ✓ (Or adding #13) Nested Loop ✓columns from 0 to iSize ✓ Add symbol to output line ✓ Display output line ✓ **Note:** The start value of the loop will depend on whether 1 was added to iSize (either 0 or 1) A correct solution that does not use a nested loop must be penalised by 1 mark The mark allocated to the nested loop range must be the same at the outer loop	**11**	
1.5	**Button - [1.5 – Characters]** Randomly generate an acceptable value ✓ Use the value to extract the corresponding character ✓ Loop ✓ Add random character ✓ to output String ✓ Set as previous character ✓ Generate new current character ✓ Until current character ✓ = previous character ✓ Display output String ✓ **Note:** ASCII table in the range 65 to 91 String, array or case statement e.g. 1 to 26	**10**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – SQL AND DATABASE PROGRAMMING

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

2.1.2	**Button [2.1.2 – Birth year]**
	`SELECT MemberName, MemberSurname, BirthDate` ✓
	`FROM tblMembers` ✓
	`WHERE YEAR (BirthDate) = 2002` ✓
	
	`Alternatives for year:`
	`WHERE YEAR (BirthDate) = "2002"`
	`BETWEEN #2002/01/01/# and #2002/12/31#`
	`LEFT(BirthDate,4)`
	`MID(BirthDate,1,4)`
	`LIKE "%2002%" OR LIKE "2002%"`
		3	

2.1.3	**Button [2.1.3 – Display members]**
	`SELECT MemberSurname, MemberName` ✓
	`FROM tblClubs,tblMembers` ✓
	`WHERE tblClubs.ClubId = tblMembers.ClubId` ✓
	`AND tblClubs.ClubName = "' + sClubName +'"` ✓
	
	Also accept join
	`SELECT MemberSurname, MemberName`
	`FROM tblClubs INNER JOIN tblMembers`
	`ON tblClubs.ClubId = tblMembers.ClubId`
	`WHERE tblClubs.ClubName = "' + sClubName +'"`
	
	QuotedStr(sClubName)
	
	Accept the use of aliases
		4	

2.1.4	**Button [2.1.4 - Average membership fee]**
	`SELECT Province,`
	`FORMAT(AVG(MemFee)` ✓`,"CURRENCY")` ✓
	`AS AvgFee` ✓
	`FROM tblClubs`
	`GROUP BY Province` ✓
	`HAVING` ✓ `AVG(MemFee) > 400` ✓
	
	Also accept:
	`FORMAT(AVG(MemFee), "R0.00")`
		6	

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

QUESTION 2: MARKING GRID (CONT.)

2.2	**Database Manipulation**		
2.2.1	**Button [2.2.1 – Outstanding fees]** Go to the first record in the tblClubs ✓ Use a loop to step through the tblClubs ✓ Display the club name and annual membership fee for each club ✓ Go to the first record of the tblMembers table ✓ Use a nested loop✓ to step through tblMembers ✓ If the ClubID field in tblClubs ✓ = ClubID field in tblMembers ✓ Calculate the outstanding fee ✓ //(MemFee - AmountPaid) Display the surname, amount paid and the outstanding fee in the richedit ✓ Move to the next record in the tblMembers ✓ End loop (Members table) Move to the next record in the tblClubs ✓ End loop (tblClubs) **Note:** Also accept the repeat..until loop instead of the While loop with the correct conditions	**12**	
2.2.2	**Button [2.2.2 – Update hikes completed]** Edit mode ✓ Add 1 ✓ to HikesCompleted field ✓ Post ✓	**4**	
2.2.3	**Button [2.2.3 – Add member]** Retrieve the club Id from the radiogroup ✓ Insert mode ✓ Assign values to correct fields ✓ Assign retrieved club id to the ClubID field ✓ Post ✓ Accept tblMembers.Append	**5**	
	Subtotal:	**21**	
	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**getNumberOfDays function** Function heading with integer value as return data type ✓ Result = fNumberOfDays ✓	**2**	
3.1.2	**calcDistPerDay** Function heading with integer as the return type✓ Result = round✓ (fDistance/fNumberOfDays) ✓	**3**	

3.1.3	**determineLevel function**
	
	distance per day = calcDistPerDay ✓
	If distance per day > 15 ✓ AND
	(terrain type = 'Rocky' ✓ OR terrain type = 'Sandy') ✓
	Result = 'Advanced'✓
	Else ✓
	If distance per day is from 10 ✓ to 15 ✓ //inclusive
	Result = 'Moderate' ✓
	Else
	Result = 'Easy'✓
	
	Also Accept:
	If distance per day > 15 //1 mark
	AND NOT(terrain type = 'Flat') // 2 marks
		10	

3.1.4	**calcTotalCost function** Function heading with real value as return data type ✓ and integer parameter Result = ✓ fCost * parameter ✓ The answer in the Result is the same data type as the data type in the heading ✓ Also accept currency, string or integer as the return type provided it is used correctly	**4**	
3.1.5	**toString method** Function heading with String return data type ✓ Return trailName, terrainType, distance, number of days, cost ✓ Converted to correct format ✓ Correct text and line breaks ✓	**4**	
	Subtotal: Object class	**23**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Combobox - cmbHikingTrails** Assignfile with extracted file name + '.txt' ✓ Reset file ✓ Read 4 lines from file ✓ *Instantiate the objHikingTrail object:* objHikingTrail✓:= THikingTrail.Create ✓ Use five arguments ✓ (`sTrailName,sTerrainType,iNumDays,iDistance,rCost`) with correct data types and in correct order ✓ Display message using a showMessage dialogue box ✓	**8**	
3.2.2	**Button [3.2.2 – Display hiking trail details]** Use the objHikingTrail.toString method ✓ to display hiking trail information in rich edit component ✓	**2**	
3.2.3	**Button [3.2.3 – Display cost]** Input number of members in group using an input box✓ Cost = ✓ objHikingTrail.calcTotalCost ✓ (group size✓) Accept calculateCost as the function name	**4**	
3.2.4	**Button [3.2.4 – Calculate distance per day]** Display number of km using IntToStr(objHikingTrail.**calcDistPerDay)** ✓ Display difficulty level using objHikingTrail.**determineLevel** ✓ Display number of days using IntToStr(objHikingTrail.**getNumberOfDays)** ✓ Accept the difficulty level displayed in small or capital letters	**3**	
	Subtotal Form class:	**17**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
SECTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1	**Button - [4.1 – Display distance chart]** Loop from 1 to 5 ✓ Add name to output String ✓ Nested Loop from 1 to 5 ✓ Add distance to output String ✓ Display output String ✓ in rich edit **Note:** Two sets of brackets may be used instead of index values separated by a comma e.g. arrDistances[iR][iC]	**5**	
4.2	**Button - [4.2 – Validation]** Loop from 1 to 5 ✓ Loop from 1 to 5 ✓ Test if distance at [iRow,iCol] <> distance at [iCol,iRow] ✓ Test if distance at [iRow,iCol] < distance at [iCol,iRow] ✓ Distance at [iRow,iCol] =distance at [iCol,iRow] ✓ Build String with row and col index ✓ and distance ✓ Display output String ✓ Accept the length of the array instead of 5 used in loops Alternative test: // Test if distance at [iRow,iCol] < distance at [iCol,iRow] (2 marks)	**8**	

4.3	**Button - [4.3 – Route planner]** Extract route from combo box and initialise total distance ✓ Loop 4 times ✓ (extract all possible combinations) Extract the first check point✓ (row index) Extract the second check point✓ (column index) Delete first 2 characters ✓ (logic to start at next index) Read distance from 2D ✓ using row and column Update total distance ✓ (add distance from 2D) Calculate time (time X distance) ✓ Test if hike between points at 2 and 4 OR 4 and 2 ✓ multiply time with 2 ✓ Update time per day ✓ Display names of the two checkpoints ✓ Display distance and time✓ (in any format) Test if time per day > 480 ✓ Display name of checkpoint to book ✓ Reset time per day to 0 ✓ Display total distance ✓	**17**	
	TOTAL SECTION D: **GRAND TOTAL:**	**30** **150**	

SUMMARY OF LEARNER'S MARKS:

CENTER NUMBER:		**LEARNER'S EXAMINATION NUMBER:**			
	SECTION A	**SECTION B**	**SECTION C**	**SECTION D**	
	QUESTION 1	**QUESTION 2**	**QUESTION 3**	**QUESTION 4**	**GRAND TOTAL**
MAX. MARKS	**40**	**40**	**40**	**30**	**150**
LEARNER'S MARKS					

ANNEXURE E: SOLUTION FOR QUESTION 1

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

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

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

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

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

//Alternative solution

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

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

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

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

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

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

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

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

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

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

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

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

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

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

  //Question 2.2.3
```

```
iClubID := rgpQ2_2_3.ItemIndex + 1;
  tblMembers.Insert;

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

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

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

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

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

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

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
unit HikingTrail_U;

interface

uses SysUtils;

type

  THikingTrail = class(TObject)

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

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

implementation

{ THikingTrail }

// Provided code

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

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

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

   end.
```

Main Form Unit

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

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

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

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

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

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

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

  // Question 4.1

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

  // Question 4.2

  // Question 4.2
  for iRow := 1 to 5 do
  begin
    for iCol := 1 to 5 do
    begin
      if arrDistances[iRow, iCol] < arrDistances[iCol, iRow] then
        begin
          arrDistances[iRow, iCol] := arrDistances[iCol, iRow];
          redQ4.Lines.Add('[' + IntToStr(iRow) + ',' + IntToStr(iCol)
            + '] with ' + FloatToStr(arrDistances[iCol, iRow]));
        end;
      end;
    end;
end;


//================================================================
```

// Question 4.3 17 marks

```
// ========================================================================
procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  iCnt, iRow, iCol: integer;
  rTotalDistance, rTimePerDay, rTimeMin: real;
  sRoute: String;

begin
  // Provided code
  redQ4.Clear;

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

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

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

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

    rTimePerDay := rTimePerDay + rTimeMin;

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

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

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


end.
```