WESTERN CAPE EDUCATION DEPARTMENT

INFORMATION TECHNOLOGY P1

2023

MARKS: 150

TIME: 3 hours

This question paper consists of 21 pages, 2 data pages and 2 blank pages planning pages.

INSTRUCTIONS AND INFORMATION

1. This paper is divided into FOUR sections. Candidates must answer ALL the questions from 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 the Delphi programming language.

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 the Delphi programming language for any of these routines.

8. All data structures must be defined 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/name 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 given 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 **DataFilesWC2023.exe**.

Do the following:

- Double click on the password-protected executable file.
- Click on the 'Extract' button.
- Enter the following password: **IT@Gr12!**

Once extracted, the following list of files will be available in the folder:
DataFilesWC2023

Question1:
Numbers.jpg
Question1_P.dpr
Question1_p.dproj
Question1_P.res
Question1_u.dfm
Question1_u.pas

Question 2:
DBConnection.pas
Q2_p.dpr
Q2_p.dproj
Q2_p.res
Q2_u.dfm
Q2_u.pas
Question2.mdb
Question2_BackUp.mdb

Question 3:
clsPlanets_u.pas
Question3_p.dpr
Question3_p.dproj
Question3_p.res
Question3_u.dfm
Question3_u.pas

Question 4:
Jovan.bmp
Longjump.txt
Question4_p.dpr
Question4_p.dproj
Question4_p.res
Question4_u.dfm
Question4_u.pas

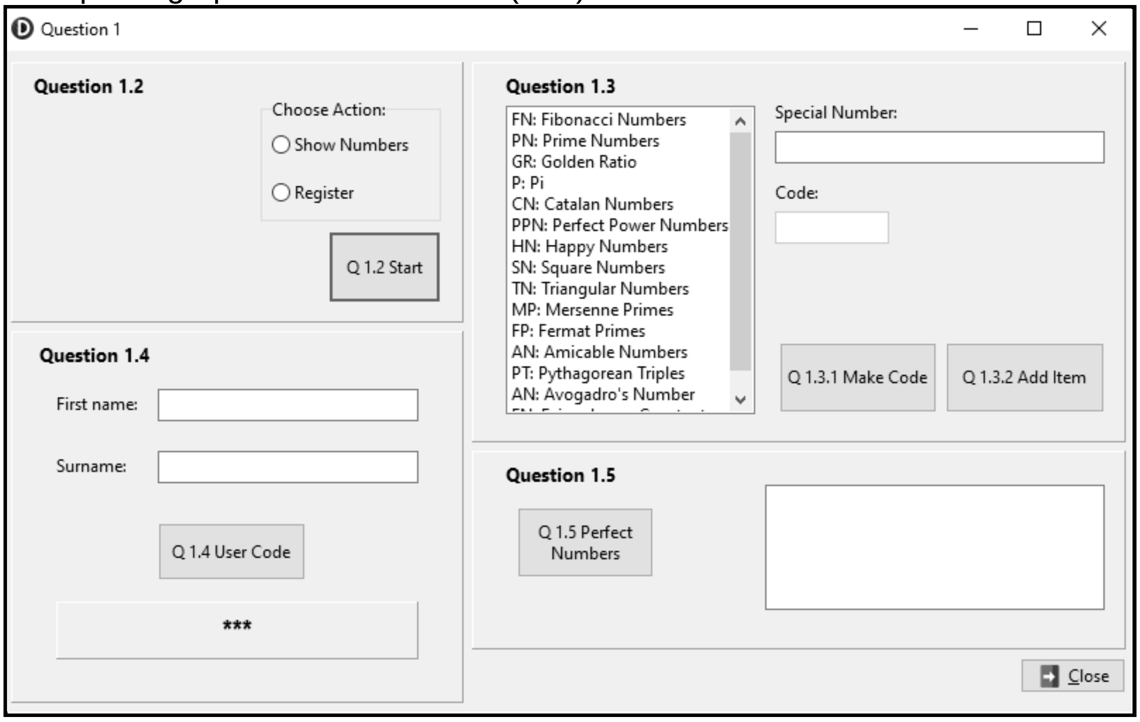
SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your name and surname as a comment in the first line of the **Question1_U.pas** file.
- Compile and execute the program. The user interface displays FOUR sections labelled QUESTION 1.2 to QUESTION 1.5. The program has no functionality currently.

Example of graphical user interface (GUI):

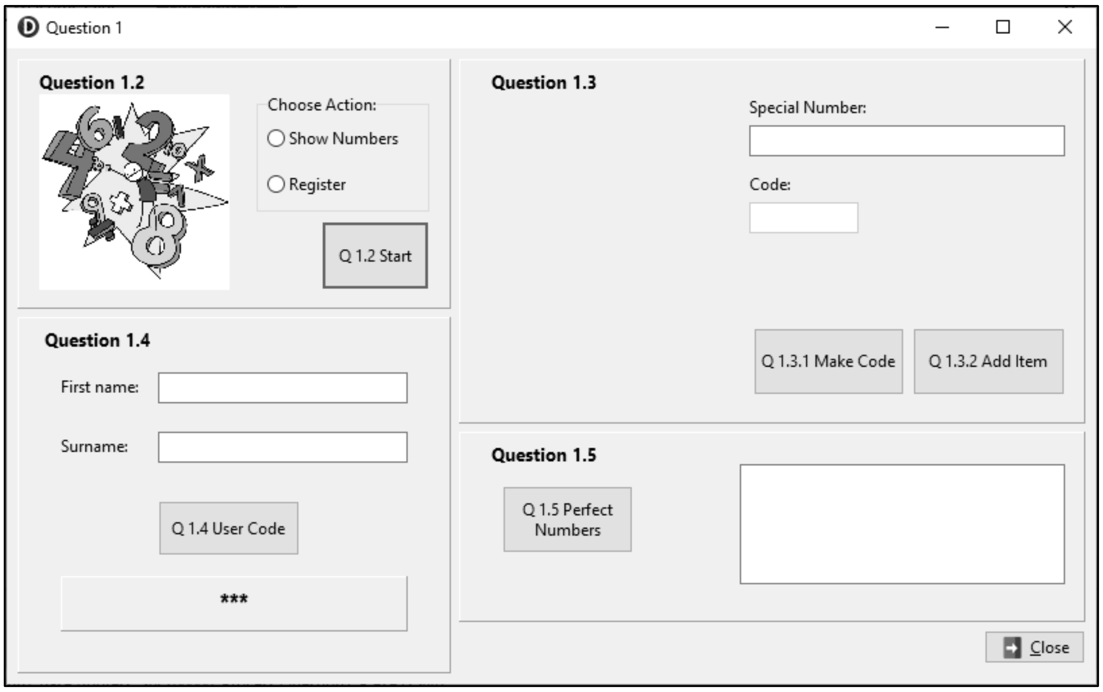


Follow the instructions below to complete the code for EACH section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.5.

There are several interesting numbers that have been given special names that can be found in mathematics, the sciences and computing.

1.1 Create an **OnShow** event for the form.
Add code to the form's **OnShow** event to do the following:
- Hide the list box **lstInteresting**.
- Disable the panel **pnlQ1_4**.
- Load the **Numbers.jpg** picture on the image component **imgQ1**. The picture has been provided in the Question 1 folder.
- Change the property so that the picture fits into the component.

Output on GUI:

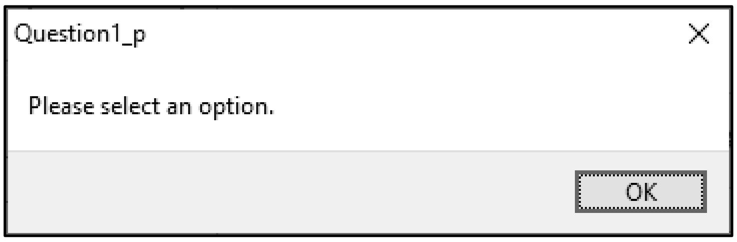


(5)

1.2 **Button [Q 1.2 Start]**

Add code to do the following:

- If the **Show Numbers** option of the radio group named **rgpQ1_2** has been selected, the **lstInteresting** component must be shown on the form.
- If the **Register** option of the radio group named **rgpQ1_2** is selected, the **pnlQ1_4** component must be enabled.
- If none of the options of **rgpQ1_2** have been chosen, an error message must be displayed as shown below.


Question1_p

Please select an option.

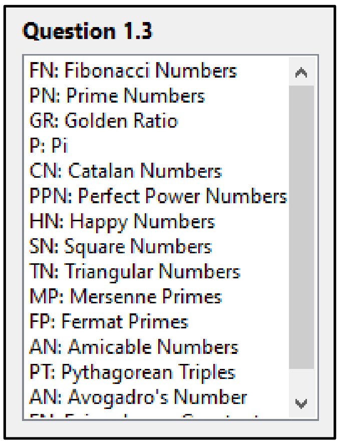
(7)

1.3 The list box **lstInteresting** given in this section contains the names of some of the interesting numbers that occur in the fields of Maths and Science.

Each number has been given a code made up of the first letters of each word in the name of the special number.

For example:

```
FN: Fibonacci Numbers
PN: Prime Numbers
GR: Golden Ratio
P: Pi
CN: Catalan Numbers
PPN: Perfect Power Numbers
...
```

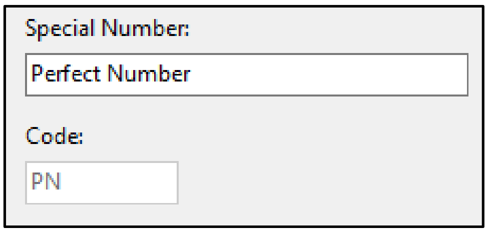

Question 1.3

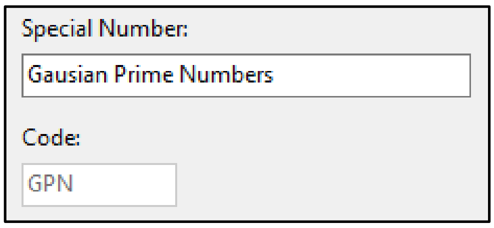
FN: Fibonacci Numbers
PN: Prime Numbers
GR: Golden Ratio
P: Pi
CN: Catalan Numbers
PPN: Perfect Power Numbers
HN: Happy Numbers
SN: Square Numbers
TN: Triangular Numbers
MP: Mersenne Primes
FP: Fermat Primes
AN: Amicable Numbers
PT: Pythagorean Triples
AN: Avogadro's Number

1.3.1 **Button [Q 1.3.1 Make Code]**

Write code to do the following:

- Store the name of a special number that has been entered in the edit box named **edtName** in a suitable global variable.
- Extract the first letter of each word in the name of the special number and use the letters to create the code as shown in the examples below.
- Ensure that the code is all in upper case letters as shown even if the names have been typed in all lower case.
- Store the code in a suitable global variable.
- Display the code you created on the edit box named **edtCode**.


Special Number: Perfect Number
Code: PN


Special Number: Gausian Prime Numbers
Code: GPN

(9)

1.3.2 **Button [Q 1.3.2 Add Item]**

Write to do the following:

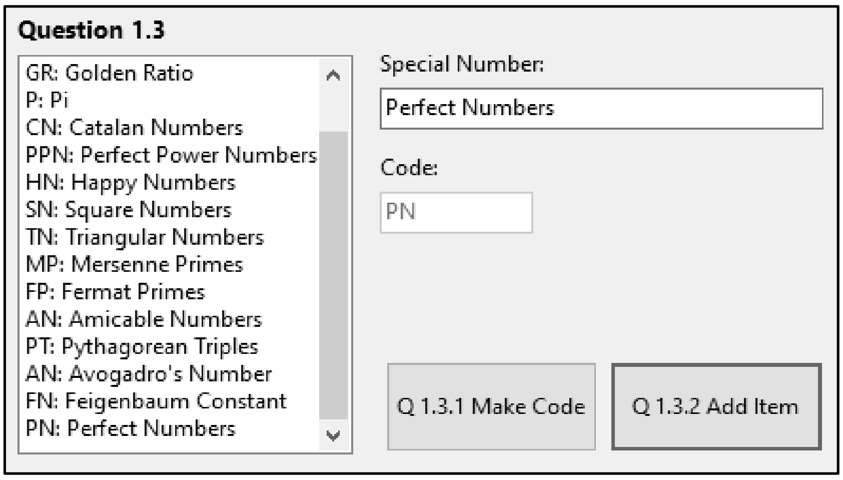
- Concatenate the code stored in the two global variables used in question 1.3.1 with a colon and the name of the number entered into the edit box **edtNumber** to form the text to be added to the list box in the format:

  `<code>: <Name of Number>`

  For example:

  `PN: Perfect Number`

Add this text to the items in the list box **lstInteresting** as shown on the next page.



Question 1.3

GR: Golden Ratio
P: Pi
CN: Catalan Numbers
PPN: Perfect Power Numbers
HN: Happy Numbers
SN: Square Numbers
TN: Triangular Numbers
MP: Mersenne Primes
FP: Fermat Primes
AN: Amicable Numbers
PT: Pythagorean Triples
AN: Avogadro's Number
FN: Feigenbaum Constant
PN: Perfect Numbers

Special Number: Perfect Numbers

Code: PN

(3)

1.4 **Button [Q 1.4 User Code]**

Write code to do the following:

- Create a registration code that is put together as follows if an item has been selected on the list box:
  - o Read the name and surname that the user has entered in the edit boxes **edtFirstname** and **edtSurname**
  - o Extract the person's initial from the first name.
  - o Generate a random integer in the range 10000 to 99999 inclusive.
  - o Add any TWO random characters from the following set of seven symbols:

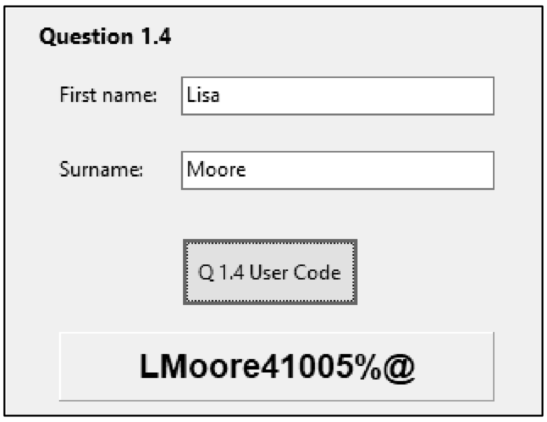
    ```
    ? # @ * % ! $
    ```

    The two symbols can be different or the same.

  - o Concatenate the registration code in the following format:
    ```
    <initial> + <surname> + <random integer>
    + <first symbol> + <second symbol>
    ```

- Display the registration code on the panel **pnlCode** as shown on the here.

NOTE: The number generated by your program as well as the symbols will probably be different because random values are being used.

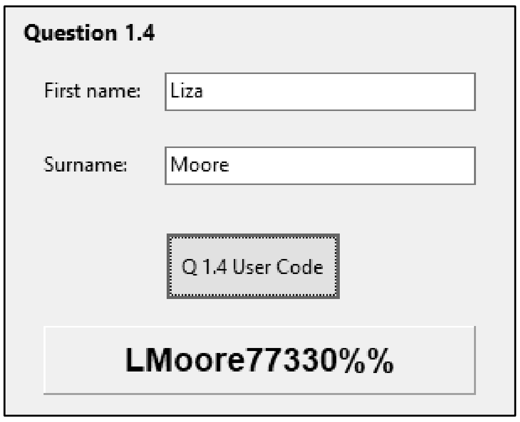


Question 1.4

First name: Lisa

Surname: Moore

LMoore41005%@



Question 1.4

First name: Liza

Surname: Moore

LMoore77330%%

(9)

1.5 **Button [Q 1.5 Perfect Numbers]**

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3 (excluding itself), and 1 + 2 + 3 = 6, so 6 is a perfect number.

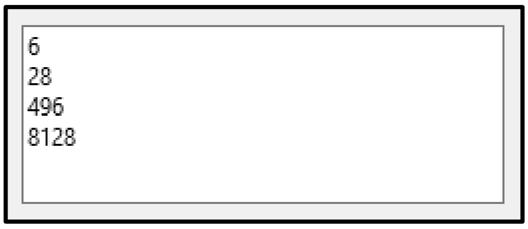
A divisor is a number that divides into another number without a remainder.

The first five perfect numbers are:

```
6, 28, 496, 8128, 33550336
```

Write code to do the following:

- Check the integer values from 1 to 10 000 and calculate which numbers are perfect numbers according to the definition given above.
- The perfect numbers must be displayed on the rich edit **redDisplay** as shown below.



```
6
28
496
8128
```

(7)

- Enter your surname and name 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

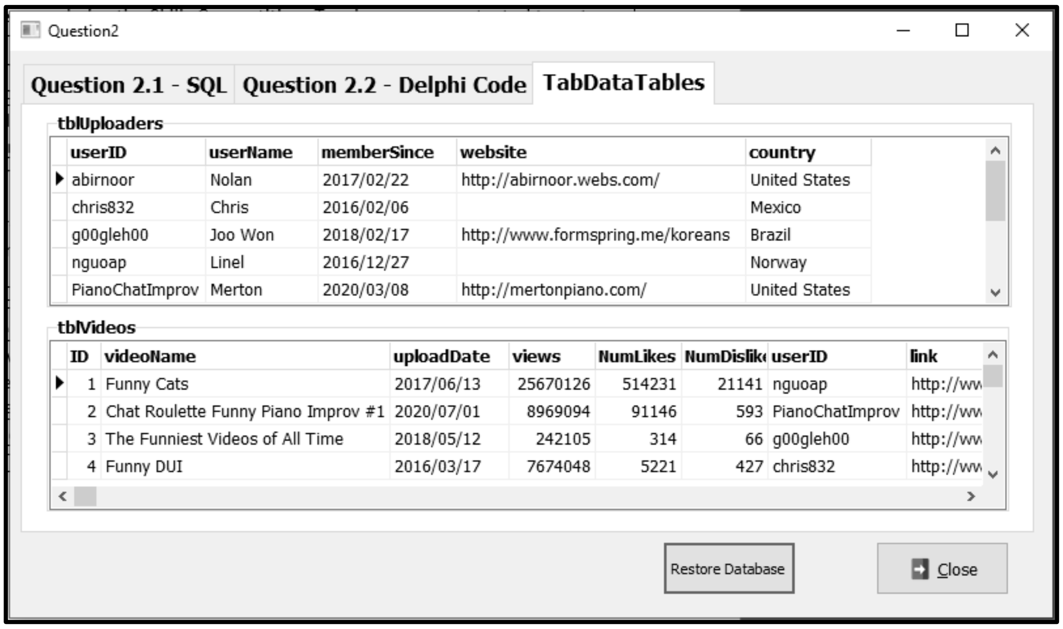
QUESTION 2: SQL AND DATABASE

Scenario
You are given a database called Question2.mdb, this is a database containing information about YouTube users and their videos back when there was still the option to give a video a dislike. This database consists out of two tables named: **tblUploaders** – contains a list of user profile information and **tblVideos** – a list of their videos they have uploaded, including statistics.

The data pages attached at the end of this question paper provide information on the design of the database and the content of the tables.

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 first four lines of data of each of the tables are displayed on the tab sheet **TableDataTables**, as shown below.



tblUploaders

userID	userName	memberSince	website	country
abirnoor	Nolan	2017/02/22	http://abirnoor.webs.com/	United States
chris832	Chris	2016/02/06		Mexico
g00gleh00	Joo Won	2018/02/17	http://www.formspring.me/koreans	Brazil
nguoap	Linel	2016/12/27		Norway
PianoChatImprov	Merton	2020/03/08	http://mertonpiano.com/	United States

tblVideos

ID	videoName	uploadDate	views	NumLikes	NumDislik	userID	link
1	Funny Cats	2017/06/13	25670126	514231	21141	nguoap	http://ww
2	Chat Roulette Funny Piano Improv #1	2020/07/01	8969094	91146	593	PianoChatImprov	http://ww
3	The Funniest Videos of All Time	2018/05/12	242105	314	66	g00gleh00	http://ww
4	Funny DUI	2016/03/17	7674048	5221	427	chris832	http://ww

- Follow the instructions below to complete the code for EACH section, as described in QUESTION 2.1 and QUESTION 2.2 that follow.
- Use SQL statements to answer QUESTION 2.1 and Delphi code to answer QUESTION 2.2.

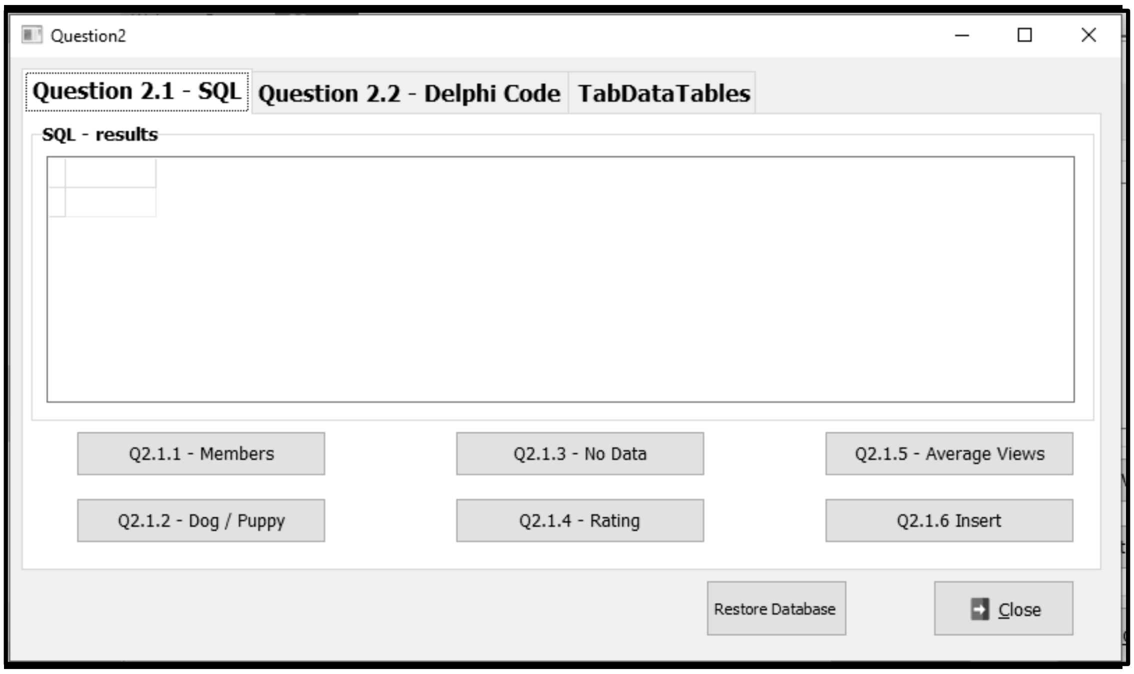
NOTE:

- The 'Restore database' button is provided to restore the data contained in the database to the original content.
- The content of the database is password protected, in other words you will not be able to gain access to the content of the database using Microsoft Access.
- Code is provided to link the GUI components to the database. Do NOT change any of the code provided.
- TWO variables are declared as global variables, as described in the table below

Variable	Data type	Description
`tblUploaders`	`TADOTable`	Refers to the table **tblUploaders**
`tblVideos`	`TADOTable`	Refers to the table **tblVideos**

2.1 **Tab sheet [Question 2.1 - SQL]**

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



NOTE:

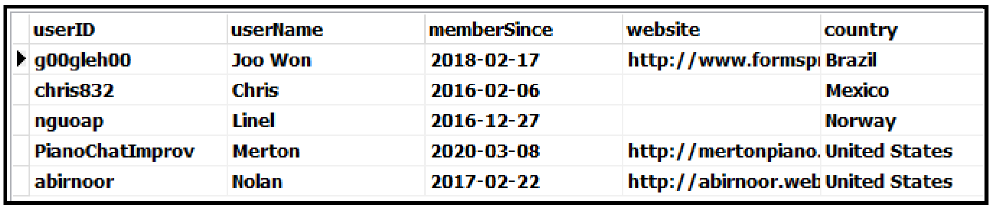
- Use ONLY SQL statements to answer QUESTION 2.1.1 to QUESTION 2.1.6.
- Code is provided to execute the SQL statements and display the results of the queries. The SQL statements assigned to the variable sSQL are incomplete.

Complete the SQL statements to perform the tasks described in QUESTION 2.1.1 to QUESTION 2.1.6 that follow.

2.1.1 **Button [Q2.1.1 - Members]**

Display the members in alphabetical order of country, then members' username.

Example of output:



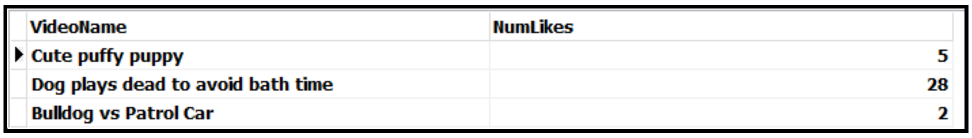
userID	userName	memberSince	website	country
g00gleh00	Joo Won	2018-02-17	http://www.formsp	Brazil
chris832	Chris	2016-02-06		Mexico
nguoap	Linel	2016-12-27		Norway
PianoChatImprov	Merton	2020-03-08	http://mertonpiano.	United States
abirnoor	Nolan	2017-02-22	http://abirnoor.wel	United States

(3)

2.1.2 **Button [Q2.1.2 - Dog / Puppy]**

Display the names of all the videos and the number of views they have received, with the word dog or puppy in the **VideoName** (You will need to have a look at the titles of the videos to construct your criteria).

Example of output:



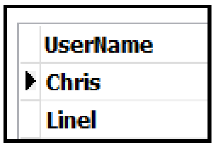
VideoName	NumLikes
Cute puffy puppy	5
Dog plays dead to avoid bath time	28
Bulldog vs Patrol Car	2

(4)

2.1.3 **Button [Q2.1.3 - No Data]**

There are a few members who did not list a website on their profile information. Display the members' username if they have no website.

Example of output:



UserName
Chris
Linel

(3)

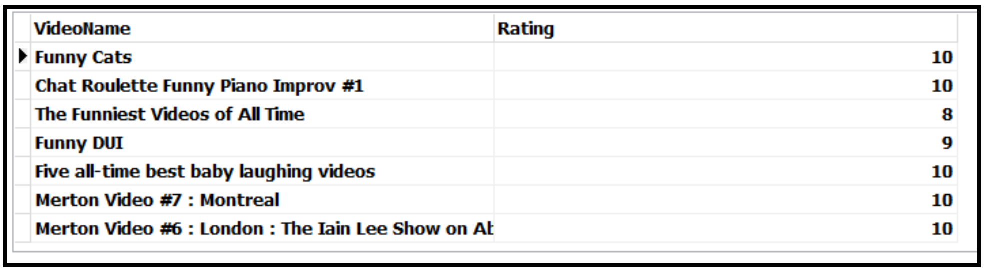
2.1.4 **Button [Q2.1.4 – Rating]**

A popularity rating can be derived from the following formula:

$$R = \frac{likes}{likes + dislikes} \times 10$$

Display the name of each YouTube video and its rating, only if the rating is greater than 7 out of 10. The rating must be rounded to the nearest integer value.

Example of output:



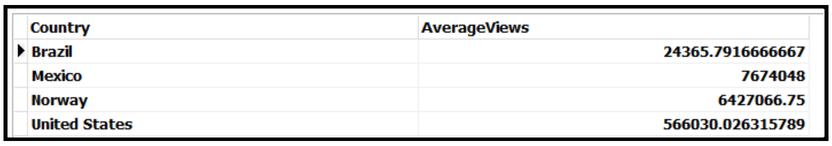
VideoName	Rating
Funny Cats	10
Chat Roulette Funny Piano Improv #1	10
The Funniest Videos of All Time	8
Funny DUI	9
Five all-time best baby laughing videos	10
Merton Video #7 : Montreal	10
Merton Video #6 : London : The Iain Lee Show on At	10

(4)

2.1.5 **Button [Q2.1.5 – Average Views]**

Calculate and display the average number of views generated by the videos uploaded from each country's users~~.~~

Example of output:



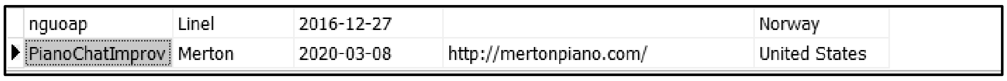
Country	AverageViews
Brazil	24365.7916666667
Mexico	7674048
Norway	6427066.75
United States	566030.026315789

(5)

2.1.6 **Button [Q2.1.6 – Insert]**

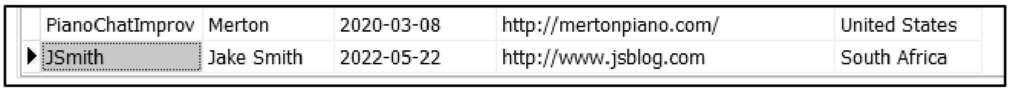
Insert your own details into the **tblUploaders** table. You joined on your sixteenth birthday.

Before:



nguoap	Linel	2016-12-27		Norway
PianoChatImprov	Merton	2020-03-08	http://mertonpiano.com/	United States

After:



PianoChatImprov	Merton	2020-03-08	http://mertonpiano.com/	United States
JSmith	Jake Smith	2022-05-22	http://www.jsblog.com	South Africa

(3)
[22]

2.2 Tab sheet [Question 2.2 - Delphi code]

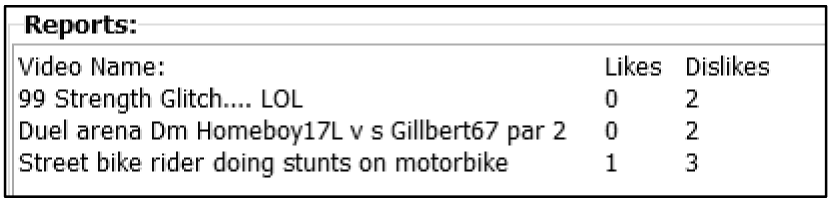
NOTE:

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

2.2.1 Button [Q2.2.1 Likes < Dislikes]

Display the names of all the videos that have more dislikes than likes as a report in the **redOutput**.

Example of output:



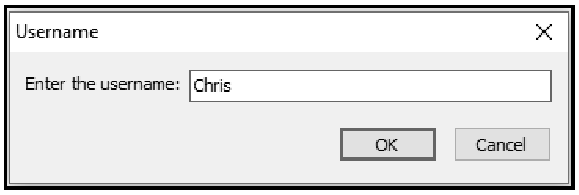
```
Reports:
Video Name:                                  Likes  Dislikes
99 Strength Glitch.... LOL                   0      2
Duel arena Dm Homeboy17L v s Gillbert67 par 2 0     2
Street bike rider doing stunts on motorbike  1      3
```

(5)

2.2.2 Button [Q2.2.2 Edit Website]

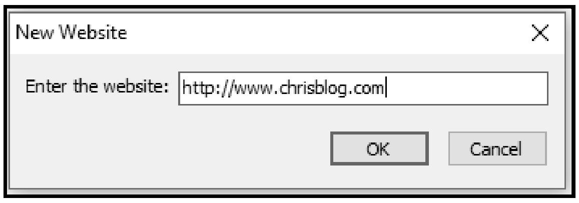
After seeing the result of the empty fields, you are given an instruction to write code to change the website field of the Uploaders table.

- Make use of an inputbox to receive the username of the record to change.
- Use another inputbox to change the website address.



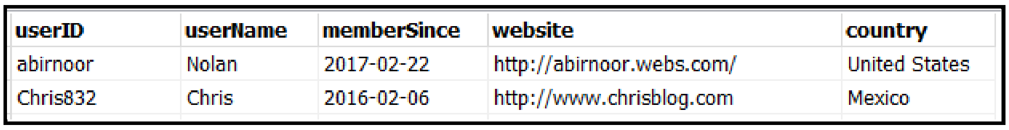
Username

Enter the username: Chris



New Website

Enter the website: http://www.chrisblog.com

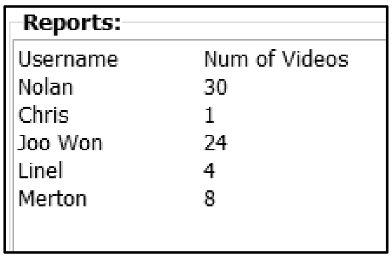


userID	userName	memberSince	website	country
abirnoor	Nolan	2017-02-22	http://abirnoor.webs.com/	United States
Chris832	Chris	2016-02-06	http://www.chrisblog.com	Mexico

(4)

2.2.3 **Button [Q2.2.3 Num of Videos]**

Write code that will display a list in the rich edit **redOutput**. This must be a list of all YouTube members' **UserNames** with the number of videos that they have uploaded.



```
Reports:
Username        Num of Videos
Nolan           30
Chris           1
Joo Won         24
Linel           4
Merton          8
```

(9)

[18]

- Enter your surname and name 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

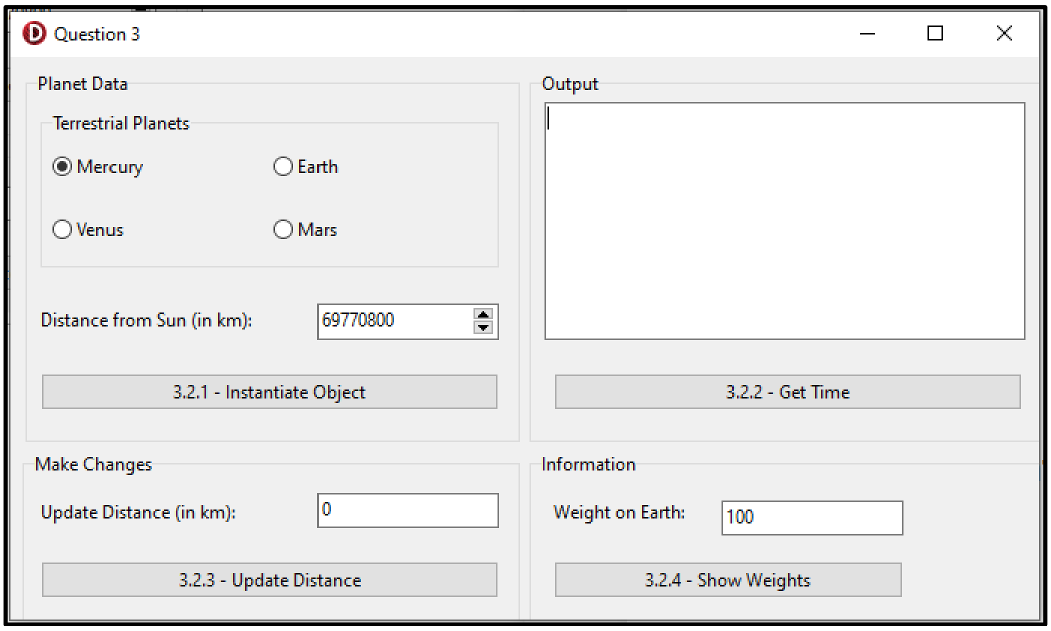
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

SCENARIO:
A South African student is interested in learning about terrestrial planets in our solar system. Rather than having to provide code for each planet, the learner decided to create a Planet object that can be used to calculate various aspects of planets within our solar system.

Do the following:

- Open the incomplete program in the **Question3** folder.
- Open the incomplete object class **clsPlanets_u.pas**.
- Compile and execute the program.

The following user interface is displayed with limited functionality:



3.1 The provided incomplete object class (**TPlanet**) contains the declaration of three attributes which describe a **planet** object.

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

Attribute	**Description**
fPlanetName	The name of the Planet
fDistance	The distance between the planet and the Sun.
fGravity	The gravity of the planet measured in meters per second.

Complete the code in the object class as described in QUESTION 3.1.1 to QUESTION 3.1.6 below.

3.1.1 Write code for a private method **setGravity** that stores the planet's gravity using the name of the planet from the object's attribute. The data is as follows:

Planet	**Gravity**
Mercury	3.7
Venus	8.9
Earth	9.8
Mars	3.7

(4)

3.1.2 Write code for a **constructor** method that will receive the planet's name and distance from the sun. Use the private method **setGravity** to set the planet's Gravity. (4)

3.1.3 Write the code for a mutator method called **setDistance** that receives a value for the planet's distance as parameter. (3)

3.1.4 Write an accessor method **getGravity** that will return the gravity of the planet. (2)

3.1.5 Write code for an auxiliary method called **getTime** that uses the distance between the planet and the sun to calculate and return the time in seconds, rounded up the nearest integer.

Formula:

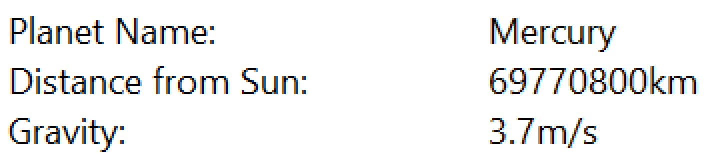
$$Time\ =\ distance\ /\ speed\ of\ light$$

Use the fact that the speed of light is 300 000 km per second. (3)

3.1.6 Write code to complete the **toString** method to return a string with all the attributes of the object in the following format:

```
Planet Name:          <fPlanetName>
Distance from Sun:    <fDistance> km
Gravity:              <fGravity> m/s
```

Example output when the program is complete:



```
Planet Name:               Mercury
Distance from Sun:         69770800km
Gravity:                   3.7m/s
```

(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 **objPlanet**.

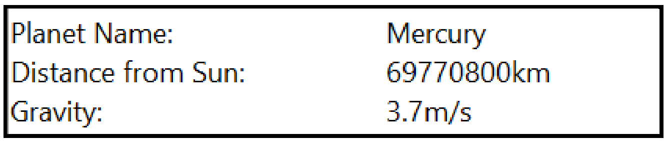
Write code to perform the tasks described in QUESTION 3.2.1 to QUESTION 3.2.6 below.

3.2.1 **Button [3.2.1 - Instantiate object]**

Write code to do the following:

- Extract the name of the planet from the radio group **rgpPlanet** and the distance from the sun from the spin edit component **spnDistance**.
- Use the information to instantiate a new Planet object.
- Call the **toString** method to display the object parameter values in **redOut**.

Example output:



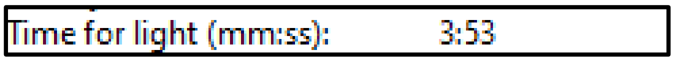
```
Planet Name:          Mercury
Distance from Sun:    69770800km
Gravity:              3.7m/s
```

(5)

3.2.2 **Button [3.2.2 – Get Time]**

Call the **getTime** method to obtain the number of seconds that it takes for light from the sun to reach the planet. Display the time, in minutes and seconds, in the rich edit component **redOut**.

Example output:



```
Time for light (mm:ss):     3:53
```

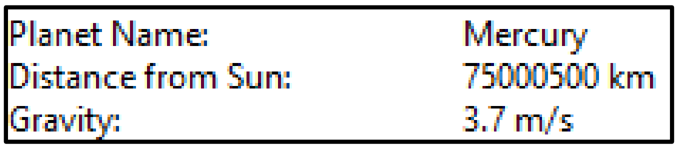
(3)

3.2.3 **Button [3.2.3 – Update Distance]**

Write code to do the following:

- Update the distance between the Sun and the planet using the edit box component **edtUpdateDistance** and relevant method.
- Call the **toString** method to show the updated object parameters in the rich edit component **redOut**.

Example output for a distance of 75 000 500 km:



```
Planet Name:          Mercury
Distance from Sun:    75000500 km
Gravity:              3.7 m/s
```

(3)

3.2.4 **Button [3.2.4 – Show Weights]**

Two global arrays have been declared for you – **arrNames** and **arrDistance**. The arrays are parallel and contain the names and corresponding distances required to instantiate a planet object in our solar system. The gravity of earth has been provided as a constant **EARTHGRAVITY**.

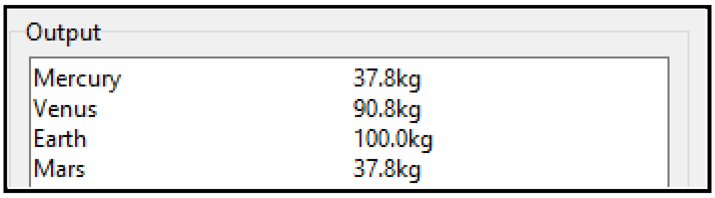
Write code to do the following:

- Obtain the weight of a person living on Earth from the edit box component **edtWeight**.
- For each entry of the array, instantiate a planet object to obtain the gravity for the planet.
- Use the gravity to determine your Earth weight on each of the other planets.
- Output the name of each planet and the respective weights (rounded to one decimal) to the rich edit **redOut**.

The following algorithm is used to determine the weight on other planets:

```
rWeightEarth ← Earth Weight
rPlanetGravity ← Gravity of planet
rPlanetWeight  ← rWeightEarth / EARTHGRAVITY * rPlanetGravity
```

Example output with an Earth weight of 100kg:



```
Mercury          37.8kg
Venus            90.8kg
Earth            100.0kg
Mars             37.8kg
```

(9)

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

TOTAL SECTION C: 40

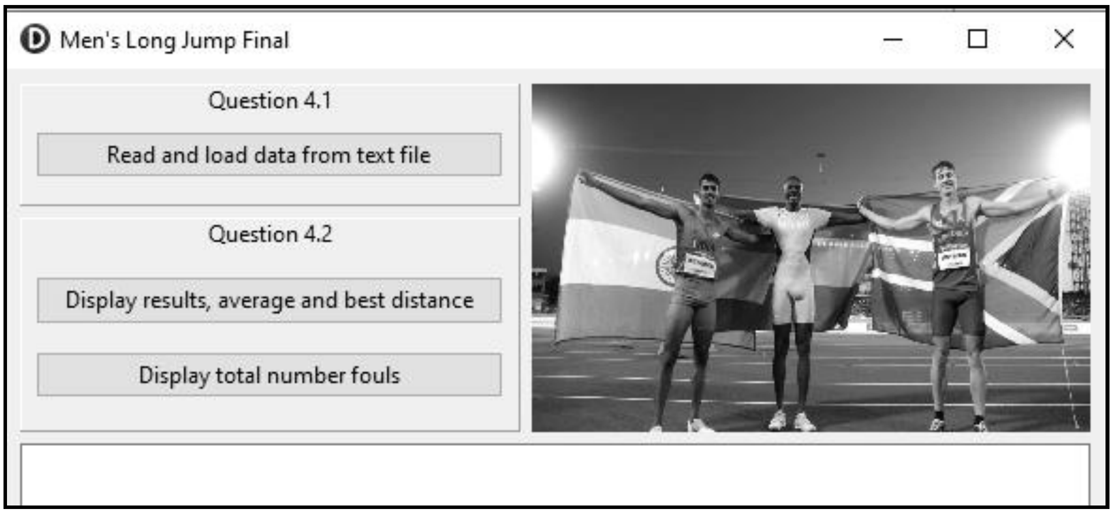
SECTION D

QUESTION 4: PROBLEM-SOLVING PROGRAMMING

A number of our excellent athletes took part in the Commonwealth Games which was hosted at Birmingham, England. One of South Africa's athletes, Jovan van Vuuren achieved a bronze medal in the men's long jump event.

As we anticipate the Olympic Games 2024 which is to be hosted by France in Paris, we would like to analyse the men's long jump results of the Commonwealth Games 2022.

- Compile and execute the incomplete program in the **Question 4** folder.
- The program has no functionality currently.
- The following user interface is displayed:


Men's Long Jump Final
Question 4.1
Read and load data from text file
Question 4.2
Display results, average and best distance
Display total number fouls

NOTE: Question 4 contains **QUESTION 4.1** and **QUESTION 4.2**. You are required to answer both questions.

The program contains the code shown below for the declaration of three global parallel arrays called **arrAthletes**, **arrCountries** and **arrDistances**.

- **arrAthletes** contains the surnames and names (in that order) of the eight male finalists who took part in the men's long jump.
- **arrCountries** contains the countries that these eight athletes represented.
- **arrDistances** contains the recorded distances in metres (correct to two decimal digits) that each finalist achieved in the men long jump event.
- The 3 parallel arrays (**arrAthletes**, **arrCountries** and **arrDistances**) are declared globally with a maximum size of 8 elements as follows:

```
var
    arrAthletes: array [1 .. 8] of string;
    arrCountries: array [1 .. 8] of string;
    arrDistances: array [1 .. 8] of real;
```

NOTE: **arrAuxAthletes, arrAuxCountries** and **arrAuxDistances** are three populated constant arrays that have been commented and must ONLY be used to answer **QUESTION 4.1** if your code to populate the arrays **arrAthletes**, **arrCountries** and **arrDistances** was NOT successful.

A text file called **Longjump.txt** is provided and contains a number of lines of data containing the full names of the athletes, the countries and the longest distances obtained for each athlete. The data is saved in the text file in the following format:

```
<Surname and name>#<Country>#<Distance in metres>
```

Example of the data of the 8 final long jumpers in the text file **Longjump.txt**:

```
Nairn LaQuan#Bahamas#8.09
Sreeshankar Murali#India#8.08
Van Vuuren Jovan#South Africa#8.06
Thompson Shawn-D#Jamaica#8.05
Yahiya Muhammed#India#7.97
Frayne Henry#Australia#7.94
James Tristan#Dominica#7.85
Otuonye Ifeanyichukwu#Turks and Caicos Islands#7.80
```

Explanation of the first three lines of data in the **Longjump.txt** text file:

- LaQuan Nairn from Bahamas achieved a best distance of 8.09 metres.
- Murali SreesHankar from India achieved a best distance of 8.08 metres.
- Jovan van Vuuren from South Africa achieved a best distance of 8.06 metres.

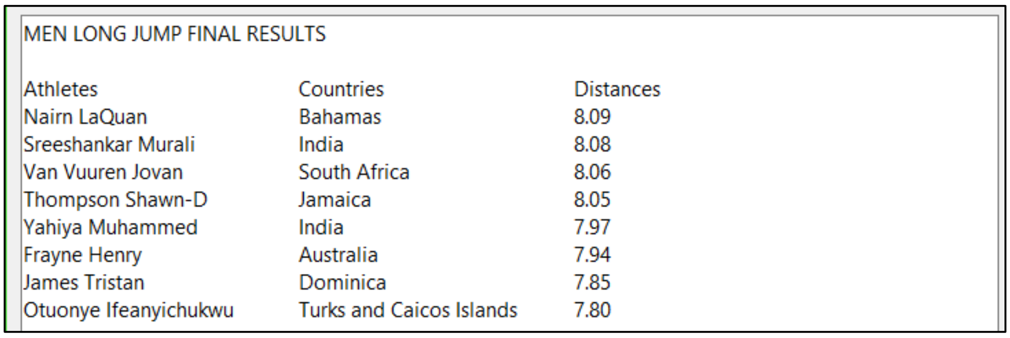
NOTE:
- A method called **Display** has been provided that receives as parameter a string that contains the title/heading of the output that is to be displayed, e.g. 'MEN LONG JUMP FINAL RESULTS.

Complete the code for **QUESTION 4.1.**

4.1 **Button [Read and load data from text file]**

Write code to do the following:
- Do a test to determine if the text file **Longjump.txt** can be accessed.
  - If not, display an appropriate message to indicate that the file cannot be accessed and leave the procedure.
  - If the text file can be accessed, use the text file **Longjump.txt** to populate the arrays **arrAthletes**, **arrCountries** and **arrDistances** with the correct data.
- Write code to call the **Display** method that displays the title/headings, column headings and the content of the three arrays **arrAthletes, arrCountries** and **arrDistances** (in neat columns) in **redQ4** has been provided**.**

Example output:



```
MEN LONG JUMP FINAL RESULTS

Athletes                Countries                  Distances
Nairn LaQuan            Bahamas                    8.09
Sreeshankar Murali      India                      8.08
Van Vuuren Jovan        South Africa               8.06
Thompson Shawn-D        Jamaica                    8.05
Yahiya Muhammed         India                      7.97
Frayne Henry            Australia                  7.94
James Tristan           Dominica                   7.85
Otuonye Ifeanyichukwu   Turks and Caicos Islands   7.80
```

(14)

The 8 best long jump athletes qualified to take part in the final which gives the athletes the opportunity to jump 6 more jumps each. The results of these 6 final jumps for the 8 athletes are provided in a two-dimensional array called **arrJumps**.

The two-dimensional array **arrJumps** is declared globally as follows:

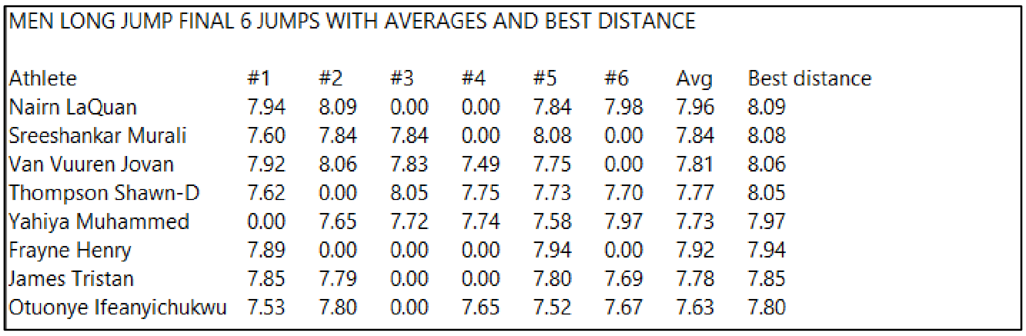
```
arrJumps: array [1 .. 8, 1 .. 6] of real = ((7.94, 8.09, 0, 0, 7.84, 7.98),
     (7.60, 7.84, 7.84, 0, 8.08, 0), (7.92, 8.06, 7.83, 7.49, 7.75, 0),
     (7.62, 0, 8.05, 7.75, 7.73, 7.70), (0, 7.65, 7.72, 7.74, 7.58, 7.97),
     (7.89, 0, 0, 0, 7.94, 0), (7.85, 7.79, 0, 0, 7.80, 7.69),
     (7.53, 7.80, 0, 7.65, 7.52, 7.67));
```

4.2.1 **Button [Display results, average and best distance]**

Display the information about each athlete as follows:

- Display each athlete's name and surname. (Column 1)
- Display the six final distances jumped for each athlete. (Columns 2 to 7)
- Calculate the average distance jumped for all the non-zero distances for every athlete. (Column 8)
- Crossing the line (the take-off board) in long jump results in a foul jump and does not count. The distance for a foul jump is indicated as 0.00 (zero). It is required that you write code to count how many jumps in total for all 8 finalists resulted in foul jumps. You have been provided with a global variable **iFouls** that has been declared and initialised. (NOTE: The result will only be displayed in **QUESTION 4.2.2**).
- Display the best distance jumped for each athlete (column 9)

Example of output:



```
MEN LONG JUMP FINAL 6 JUMPS WITH AVERAGES AND BEST DISTANCE

Athlete                #1    #2    #3    #4    #5    #6    Avg   Best distance
Nairn LaQuan           7.94  8.09  0.00  0.00  7.84  7.98  7.96  8.09
Sreeshankar Murali     7.60  7.84  7.84  0.00  8.08  0.00  7.84  8.08
Van Vuuren Jovan       7.92  8.06  7.83  7.49  7.75  0.00  7.81  8.06
Thompson Shawn-D       7.62  0.00  8.05  7.75  7.73  7.70  7.77  8.05
Yahiya Muhammed        0.00  7.65  7.72  7.74  7.58  7.97  7.73  7.97
Frayne Henry           7.89  0.00  0.00  0.00  7.94  0.00  7.92  7.94
James Tristan          7.85  7.79  0.00  0.00  7.80  7.69  7.78  7.85
Otuonye Ifeanyichukwu  7.53  7.80  0.00  7.65  7.52  7.67  7.63  7.80
```

(13)

4.2.2 **Button [Write the total number of foul jumps to file]**

- Display the total number of foul jumps that you calculated in **QUESTION 4.2.1** by making use of an output dialogue as shown below:

```
The total number of foul jumps = 14
```

(3)

- Enter your name and surname as a comment in the first line of the program file.
- Save your program.
- Make a printout of the code if required.

TOTAL SECTION D: 30
GRAND TOTAL: 150

INFORMATION TECHNOLOGY P1

DATABASE INFORMATION QUESTION 2:

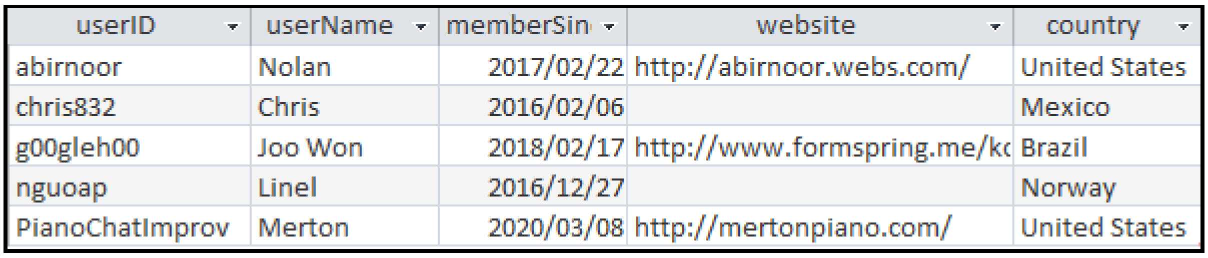
The design of the database tables is as follows:

Table: **tblUploaders**

The table contains the information of users that upload videos to YouTube.

Field name	Data type	Description
userID (PK)	Short Text	Unique ID for the YouTube user
userName	Short Text	Username for the YouTube user
MemberSince	Date/Time	Date the user jpoined YouTube
Website	Short Text	URL of the website
Country	Short Text	Country of origin of the YouTube user

Example of the records in the **tblUploaders** table:



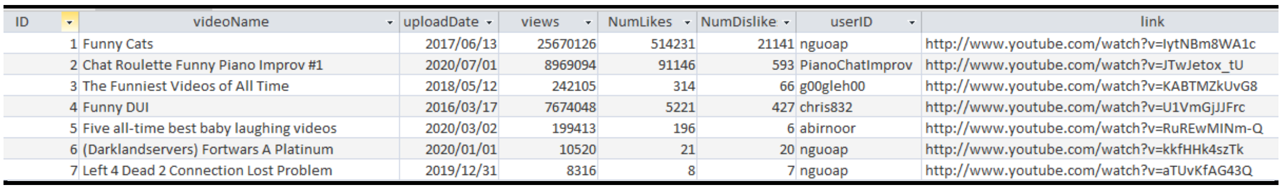
userID	userName	memberSin	website	country
abirnoor	Nolan	2017/02/22	http://abirnoor.webs.com/	United States
chris832	Chris	2016/02/06		Mexico
g00gleh00	Joo Won	2018/02/17	http://www.formspring.me/kc	Brazil
nguoap	Linel	2016/12/27		Norway
PianoChatImprov	Merton	2020/03/08	http://mertonpiano.com/	United States

Table: **tblVideos**

The table contains the information on the different transactions when customers purchase goods at the listed stores.

Field name	Data type	Description
ID (PK)	AutoNumber	Unique number for each video
VideoName	Short Text	Name of the video that was uploaded
UploadDate	Date/Time	Date the video was uploaded
Views	Number	Number of views the video received
NumLikes	Number	Number of likes the video received
NumDislikes	Number	Number of dislikes the video received
UserID (FK)	Short Text	User that uploaded the video
Link	Short Text	Link to the video

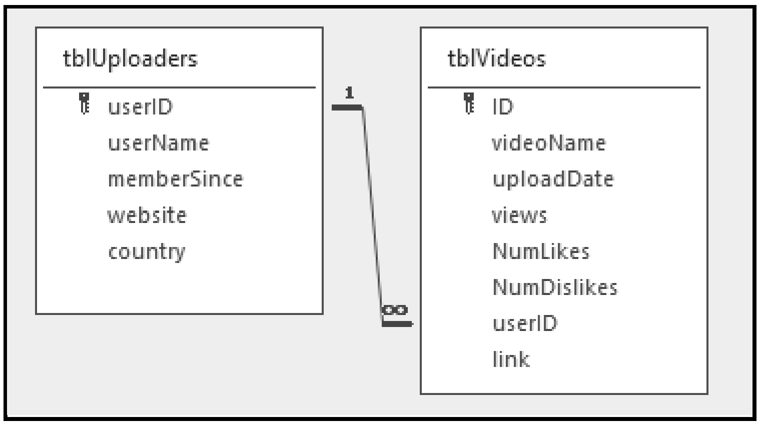
Example of the first seven records in the **tblVideos** table:



ID	videoName	uploadDate	views	NumLikes	NumDislike	userID	link
1	Funny Cats	2017/06/13	25670126	514231	21141	nguoap	http://www.youtube.com/watch?v=IytNBm8WA1c
2	Chat Roulette Funny Piano Improv #1	2020/07/01	8969094	91146	593	PianoChatImprov	http://www.youtube.com/watch?v=JTwJetox_tU
3	The Funniest Videos of All Time	2018/05/12	242105	314	66	g00gleh00	http://www.youtube.com/watch?v=KABTMZkUvG8
4	Funny DUI	2016/03/17	7674048	5221	427	chris832	http://www.youtube.com/watch?v=U1VmGjJJFrc
5	Five all-time best baby laughing videos	2020/03/02	199413	196	6	abirnoor	http://www.youtube.com/watch?v=RuREwMINm-Q
6	(Darklandservers) Fortwars A Platinum	2020/01/01	10520	21	20	nguoap	http://www.youtube.com/watch?v=kkfHHk4szTk
7	Left 4 Dead 2 Connection Lost Problem	2019/12/31	8316	8	7	nguoap	http://www.youtube.com/watch?v=aTUvKfAG43Q

NOTE: Connection code has been provided.

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



INFORMATION TECHNOLOGY P1 – PLANNING PAGE 1

INFORMATION TECHNOLOGY P1 – PLANNING PAGE 2