Province of the
EASTERN CAPE
EDUCATION

NATIONAL SENIOR CERTIFICATE

GRADE 12

SEPTEMBER 2021

INFORMATION TECHNOLOGY P1

MARKS: **150**

TIME: **3 hours**

This question paper consists of 19 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.

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 locate, 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 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 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. 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. Make sure 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.

    Do the following:

    - Double click on the password-protected executable file.
    - Click on the extract button.
    - Enter the following password: **gR12#21sEpt**

    Once extracted, the following list of files will be available in the folder **DataSept2021**:

    Question 1:

    Question1_u.pas
    Question1_u.dfm
    Question1_p.dpr
    Question1_p.res
    rain.jpg

    Question 2:

    dbConnection_u.pas
    Birds.mdb
    BirdsBackup.mdb
    Question2_u.pas
    Question2_u.dfm
    Question2_p.dpr
    Question2_p.res

    Question 3:

    Question3ClassDefinition.pas
    Question3_u.pas
    Question3_u.dfm
    Question3_p.dpr
    Question3_p.res

    Question 4:

    water.txt
    Question4_u.pas
    Question4_u.dfm
    Question4_p.dpr
    Question4_p.res

QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your full name as a comment in the first line of the **Question1_u.pas** file.
- Compile and execute the program. The program has no functionality currently.
- Follow the instructions below to complete the code for each section of QUESTION 1, as described in QUESTION 1.1, QUESTION 1.2, QUESTION 1.3 and QUESTION 1.4.

Your school has created a club to provide information and conduct research on water supply and rainfall issues. Complete the program for the club by using the instructions below.

1.1 Button [1.1 Display]

Write code to do the following:

Load the image named **rain.jpg** into the image component named **imgRain.**
Enable the panel named **pnlBath**.
Change the background colour of the panel, named **pnlRain,** to aqua. (3)

1.2 Button [1.2 Process]

Rainfall calculator.
How many 150 litre baths can be filled after rainfall has occurred over a certain area in your garden or on your roof?

Write code to do the following:

The user will enter the height and width in the spinedits named **sedWidth** and **sedHeight** and choose the number of millimetres from the combobox named **cmbRain.**

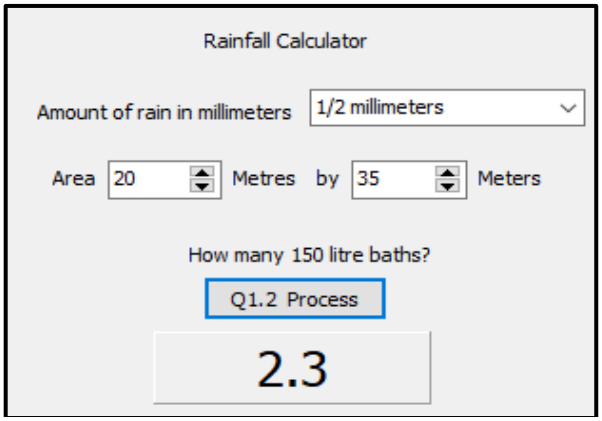
The number of 150 litre baths will be equal to the number of square meters multiplied by the amount of water in millimetres and divided by 150. Round the answer to 1 decimal place.

Example: An area of 20 m by 35 m receives 1/2 millimetre (0.5 mm) of rain.
20 x 35 x 0.5 /150 = 2.3 baths (150 litres each)

Use the following data to test your solution:

Rainfall in mm	Height in Metres	Width in Metres	Number of 150 litre baths
15	9	10	9
1/10	20	20	0.3
1/2	20	35	2.3
30	9	12	22

Example of output:



(11)

1.3 **Button [1.3 Find rainfall type]**

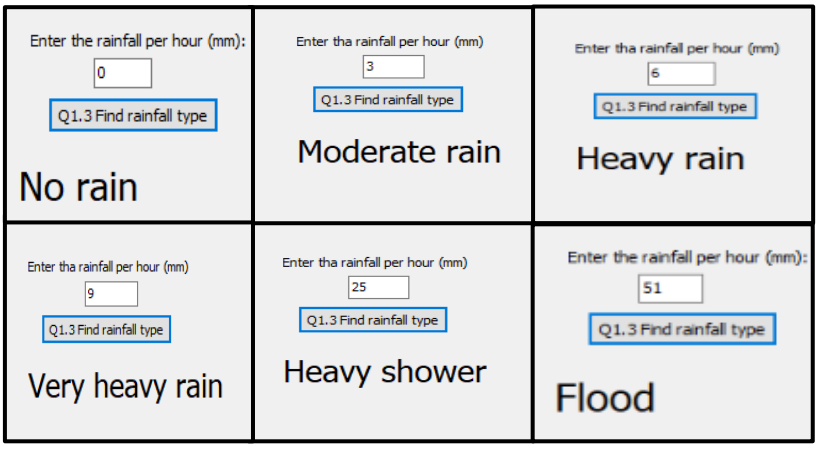
We wish to find out what type of rainfall occurred in our area.
The following descriptions apply for different amounts of rain that falls in 1 hour.

No rain: Less than 1 mm per hour
Moderate rain: Greater than or equal to 1 mm per hour, but less than 4 mm per hour.
Heavy rain: Greater than or equal to 4 mm per hour, but less than 8 mm per hour.
Very heavy rain: Greater than or equal to 8 mm per hour but less than 10 mm per hour.
Heavy shower: Greater than or equal to 10 mm per hour, but less than or equal to 50 mm per hour.
Flood: Greater than 50 mm per hour.

Write code in this button onclick event to determine the type of rainfall that occurred. The user will enter the amount of rain in millimetres per hour in the edit box named **edtRain** and then the program must calculate and display, in the label named **lblRainfallType,** the description of the type of rainfall as described above.

You must also write code to test if an integer has been entered into the edit box named **edtRain**. If the input is not an integer an error message must be displayed, and the procedure must exit.

Examples of output:



(13)

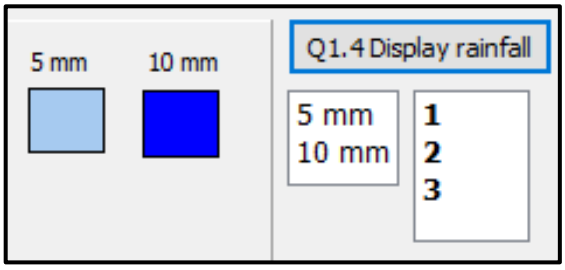
1.4 **Button [1.4 Display Rainfall]**

You are required to calculate and display how many times either 5 mm or 10 mm of rainfall was recorded in 8 different areas.

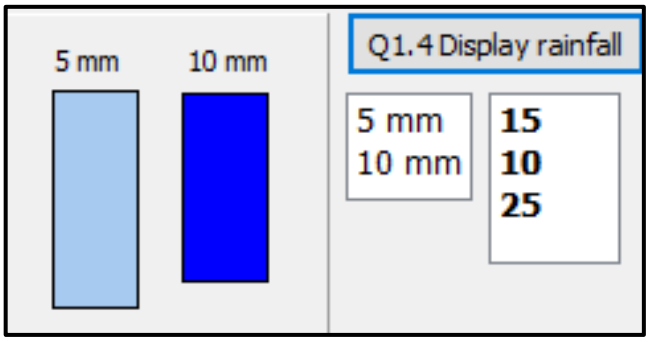
Write code to repeat the following instructions 8 times:

- Generate 2 different random numbers between 5 and 10 (both included).
- If the random numbers are either 5 or 10, then add 1 to the quantity in the list box named **LstQuantity** on the correct line that matches the position of either 5 mm or 10 mm as indicated in the list box named **LstRain.**
- Calculate the sum of the two quantities and display it on the third line of the list box named **LstQuantity**.
  For example: 1 + 2 = 3 and 15 + 10 = 25
- Change the heights of the shapes named **shp5mm** and **shp10mm** by adding the values of the corresponding numbers in **LstQuantity** to the current height of each shape.

Example of output after 1 execution of the button onclick event showing that 1 area experienced 5 mm of rainfall and 2 areas experienced 10 mm of rainfall:



Example of output after more than one execution of the button onclick event:



(13)

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

[40]

QUESTION 2: DATABASE PROGRAMMING

The database **Birds.mdb** contains the details of birds and the status of birds in South Africa. The database contains two tables, namely **Bird** and **Status.**

Table: **Status**
This table contains the descriptions of the possible status of a bird.

Field name	Data type	Description
StatusID	Number	A unique number assigned to each status
StatusName	Text (50)	The description of the Status of a bird

Example of data in the **Status** table:

StatusID	StatusName
1	Critically Endangered
2	Endangered
3	Vulnerable
4	Near Threatened
5	Least Concern
6	Not Assessed

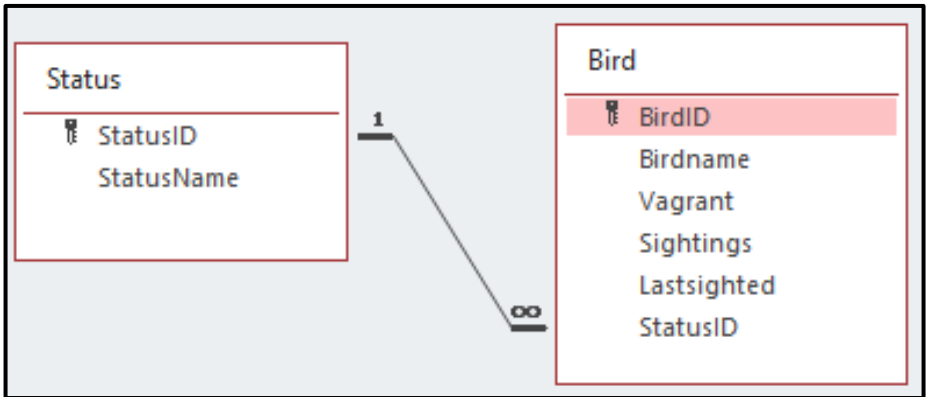
Table: **Bird**
This table contains information of birds and their sightings in South Africa.

Field name	Data type	Description
BirdID	Number	A unique number assigned to a bird
Birdname	Text (50)	The common name of a bird
Vagrant	Yes/No	Bird wanders beyond the limits of their natural range — can turn up almost anywhere
Sightings	Number	Total number of sightings of a bird
Lastsighted	Date (short date)	Date that the bird was last seen
StatusID	Number	Foreign key to connect to the Bird table

Example of data of the first fourteen records of the **Bird** table:

BirdID	Birdname	Vagrant	Sightings	Lastsighted	StatusID
1	African Broadbill	☐	533	2005/01/23	3
2	African Finfoot	☐	208	2007/07/12	3
3	African Grass Owl	☐	161	2012/05/27	3
4	African Marsh Harrier	☐	366	2010/10/08	2
5	African Rock Pipit	☐	918	2005/12/06	4
6	Agulhas Long-billed Lark	☐	726	2006/07/23	4
7	Antarctic Tern	☐	333	2008/07/25	2
8	Barlow's Lark	☐	720	2009/03/19	4
9	Bat Hawk	☐	347	2007/01/17	2
10	Bateleur	☐	364	2011/03/11	2
11	Bearded Vulture	☐	352	2007/12/08	1
12	Blue Crane	☐	206	2012/05/23	4
13	Blue Korhaan	☐	202	2010/10/11	5
14	Blue Petrel	☑	511	2011/01/18	4

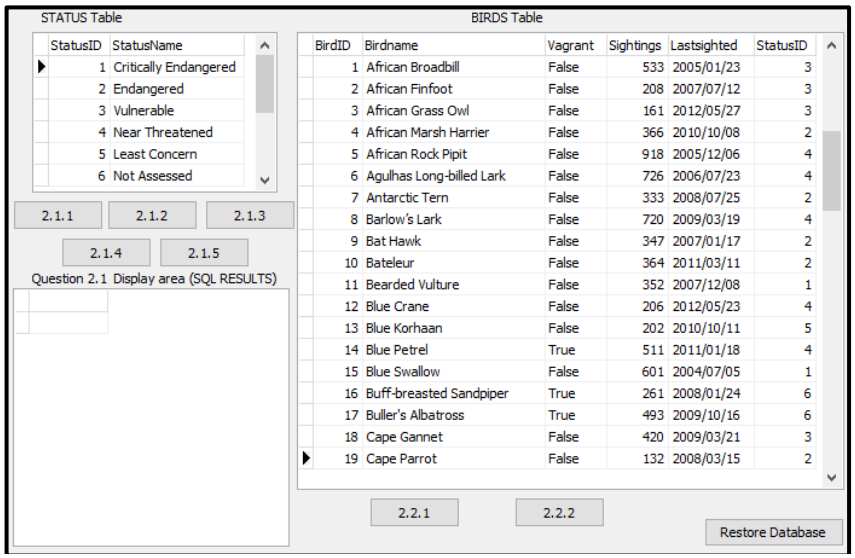
The relationship between the two tables is shown below:



Do the following:

- Open the incomplete project file called **Question2_p.dpr** in the Question 2 folder.
- Enter your name 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 user interface is displayed below:



StatusID	StatusName
1	Critically Endangered
2	Endangered
3	Vulnerable
4	Near Threatened
5	Least Concern
6	Not Assessed

BirdID	Birdname	Vagrant	Sightings	Lastsighted	StatusID
1	African Broadbill	False	533	2005/01/23	3
2	African Finfoot	False	208	2007/07/12	3
3	African Grass Owl	False	161	2012/05/27	3
4	African Marsh Harrier	False	366	2010/10/08	2
5	African Rock Pipit	False	918	2005/12/06	4
6	Agulhas Long-billed Lark	False	726	2006/07/23	4
7	Antarctic Tern	False	333	2008/07/25	2
8	Barlow's Lark	False	720	2009/03/19	4
9	Bat Hawk	False	347	2007/01/17	2
10	Bateleur	False	364	2011/03/11	2
11	Bearded Vulture	False	352	2007/12/08	1
12	Blue Crane	False	206	2012/05/23	4
13	Blue Korhaan	False	202	2010/10/11	5
14	Blue Petrel	True	511	2011/01/18	4
15	Blue Swallow	False	601	2004/07/05	1
16	Buff-breasted Sandpiper	True	261	2008/01/24	6
17	Buller's Albatross	True	493	2009/10/16	6
18	Cape Gannet	False	420	2009/03/21	3
19	Cape Parrot	False	132	2008/03/15	2

- Follow instructions to complete the code for each question, as described in QUESTION 2.1 and QUESTION 2.2.
- **Use SQL code to answer QUESTION 2.1 and Delphi code to answer QUESTION 2.2.**

NOTE:

- The **[Restore Database]** button is provided to restore your data contained in the database to the original content. If you need to test your code on the original data, you may click this button to restore data.
- Do NOT change any of the code provided.
- Code is provided to link the GUI components to the database.
- **TWO** variables are declared as global variables, as described in the table below.
- **Use tblStatus and tblBird components in Question 2.2 only.**

Variable	Data type	Description
tblStatus	TADOTable	Refers to the table named **Status**
tblBird	TADOTable	Refers to the table named **Bird**

2.1 In this section you may ONLY use **SQL statements** to answer QUESTION 2.1.1 to QUESTION 2.1.5.

Code to execute the **SQL statements** and display the results of the queries is provided. The SQL statements are incomplete.

Do the following to complete the incomplete **SQL statements** assigned to the variables sSQL1, sSQL2, sSQL3, sSQL4 and sSQL5 per question respectively.

2.1.1 **Button [2.1.1]**

Write SQL code to display all **Status** details sorted in reverse alphabetical order of **StatusName.**

Example of output:

StatusID	StatusName
3	Vulnerable
6	Not Assessed
4	Near Threatened
5	Least Concern
2	Endangered
1	Critically Endangered

(3)

2.1.2 **Button [2.1.2]**

Write SQL code to display the **BirdName** of all birds that have had less than 200 **Sightings**.

Example of output:

Birdname
African Grass Owl
Cape Parrot
Eastern Bronze-naped Pigeon
Green Barbet
Ground Woodpecker
Maccoa Duck
Mangrove Kingfisher
Pel's Fishing Owl
Swamp Nightjar

(3)

2.1.3 **Button [2.1.3]**

The user must enter the name of a bird. Code has been provided for the bird name "**vulture**" to be entered in an input box and saved in a variable named **sline.**

Write SQL code to display the **BirdName** and **LastSighted** date of all birds that contain the word stored in the variable, **sline**.

Example of output:

Birdname	Lastsighted
Bearded Vulture	2007/12/08
Egyptian Vulture	2006/08/09
Rüppell's Vulture	2011/05/19

(4)

2.1.4 **Button [2.1.4]**

Write an SQL code to delete all birds that have not been assessed and where the status is listed as 'Least concern'. The **StatusID** for 'Not Assessed' is the number **6** and the **StatusID** for 'Least Concern' is the number **5**.

(Code has been written to display the successfully updated table after deleting.)

Example of output of the first 9 records:

Birdname	StatusID
African Broadbill	3
African Finfoot	3
African Grass Owl	3
African Marsh Harrier	2
African Rock Pipit	4
Agulhas Long-billed Lark	4
Antarctic Tern	2
Barlow's Lark	4
Bat Hawk	2

(4)

2.1.5 **Button [2.1.5]**

Write SQL code to display the **StatusName** and **BirdName** of all birds.

Example of output of the first 8 records:

Statusname	Birdname
Critically Endangered	Bearded Vulture
Critically Endangered	Blue Swallow
Critically Endangered	Damara Tern
Critically Endangered	Leach's Storm Petrel
Critically Endangered	Southern Banded Snake Eagle
Critically Endangered	Taita Falcon
Critically Endangered	Wattled Crane
Endangered	African Marsh Harrier

(3)

2.1.6 **Button [2.1.6]**

Write SQL code to display the average quantity for each **StatusID** of all birds that were **LastSighted** in the year 2007 using the description 'AverageSightings'. The average must be rounded to 2 decimal places.

Example of output:

StatusID	AverageSightings
1	357.00
2	347.00
3	283.25
4	730.00

(8)

2.2 In this section, only Delphi programming code may be used to answer QUESTION 2.2.1 and QUESTION 2.2.2.

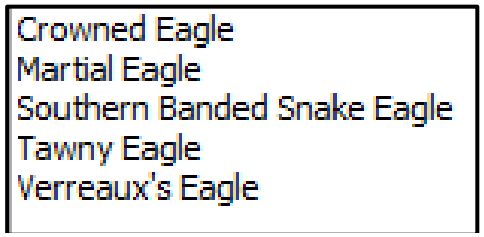
Use the global variables, **tblStatus** and **tblBird,** provided.

NO marks will be awarded for SQL statements in QUESTION 2.2.

2.2.1 **Button [2.2.1]**

Write code to display all **Birdnames** which contain the word 'EAGLE' in the richedit named **redDisplay.**

Example of output:



(6)

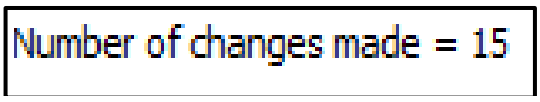
2.2.2 **Button [2.2.2]**

All birds that are categorised as vulnerable must be changed to the status of endangered.

Write code to change the **StatusID** to **2** of all birds that have a **StatusID** of **3**.

Count how many changes have been made and display the result in the richedit named **redDisplay**.

Example of output:



(9)

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

[40]

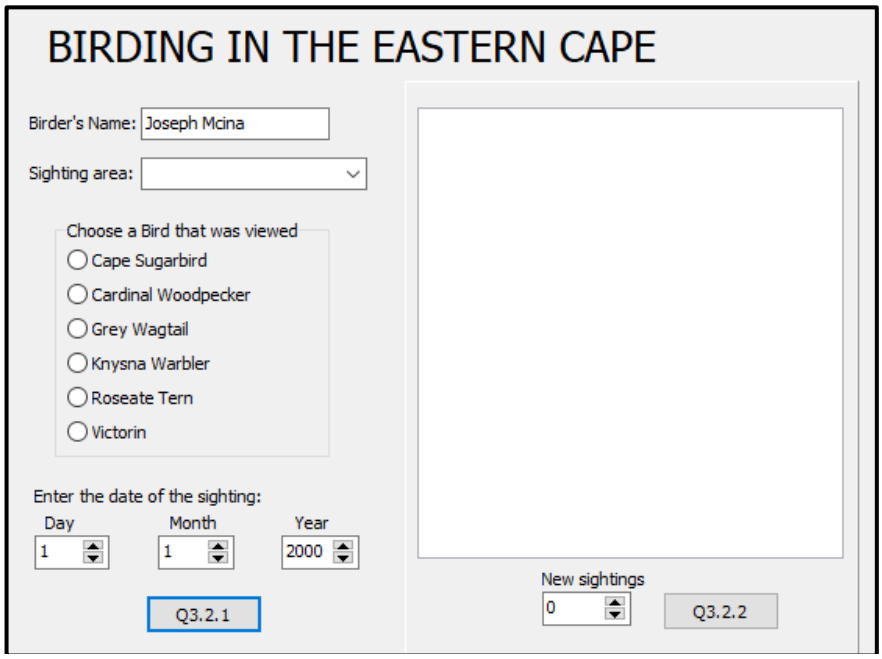
QUESTION 3: OBJECT-ORIENTED PROGRAMMING

Birding is defined as a recreational activity by people called birders to identify and observe wild birds in their natural habitat. There are many wild birds that can be sighted by birders in various biological communities in the Eastern Cape.

Do the following:

- Open the incomplete program in the Question 3 folder.
- Open the incomplete object class **Question3ClassDefinition.pas**.
- Enter your name as a comment in both **Question3ClassDefinition.pas**.and **Question3_u.pas**.
- Compile and execute the program. Currently the program has no functionality.
- Do NOT remove or change any provided code.

The following user interface is displayed:



Complete the code for this program, as specified in QUESTION 3.1 and QUESTION 3.2.

3.1 The incomplete class **(Tsighting)** contains the declaration of five attributes that describe the **objBirder** object.

NAMES OF ATTRIBUTES	DESCRIPTION
fName	The name of the person who is a birder
fArea	The name of the area where the bird was first sighted
fBird	The name of the bird that was sighted
fDate	The date that the first sighting of the bird occurred
fQuantity	The total number of sightings so far

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

3.1.1 Write code for a **constructor** method named **Create** that will receive the name of a birder (person doing the bird watching), area, bird and date of sighting as parameters.

Assign these parameter values to the correct attributes and initialise the attributes for the quantity of birds sighted to ONE. (4)

3.1.2 Write code for a method named **getSightings** which returns the quantity of bird sightings. (2)

3.1.3 Write code for a method named **IncreaseQuantity** that will receive one integer parameter to set the quantity of sightings by adding the parameter to the attribute named **fQuantity.** (4)

3.1.4 Write code for a method name **SightingGap** which must calculate the number of years that have transpired after the first sighting of a bird took place. Use the current date and the attribute named **fDate** to return the number of years as a string with the words 'years since first sighting' joined to the end of the string.

Example of output: '23 years since first sighting.' (7)

3.1.5 Write code for a method named **toString** which must return a string.

The string must contain the attributes of the class joined together in the following format:

<person's name> first viewed a <bird name>
<number of years since first sighting> at <area > on <date of first sighting>

Example of output:

Joseph Mcina first viewed a Cardinal Woodpecker
11 years since first sighting at Cape Recife on 4/11/2010 (5)

3.2 An incomplete unit **Question3_u.pas** has been provided.

It contains code for the object class to be accessible and has a global object variable, **objBirder,** already declared.

Do NOT delete or change any provided code.

Follow the instructions below to code the solution:

3.2.1 **Button Q3.2.1**

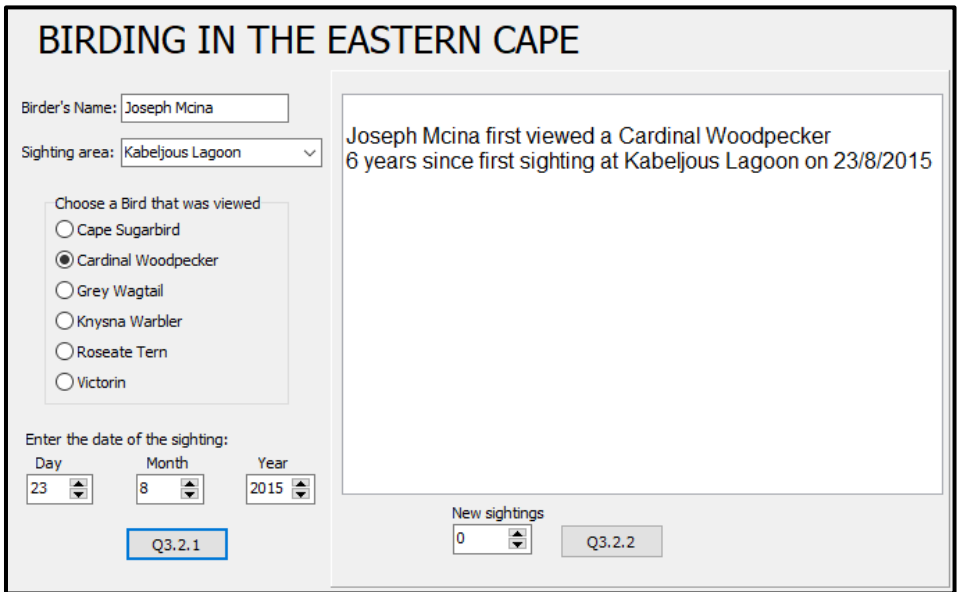
The user will enter a birder's name, choose a sighting area and bird name and enter the date that the bird was first sighted using the provided input components named **edtBirderName, cmbArea, rgpBirds, sedDay, sedMonth** and **sedYear.**

Write code to do the following:

- The date must be concatenated by joining the year, month and date in the format 'Day/Month/Year'. (Example: 23/9/2012)
- Instantiate the object, **objBirder,** using the birder's name, the sighting area, the bird name and the date first sighted.
- Clear the richedit.

Use a **method of the class** to display the birder's name, the bird name, the number of years that have passed since first sighting, the area of the sighting and the date of the first sighting.

Example of output:



(11)

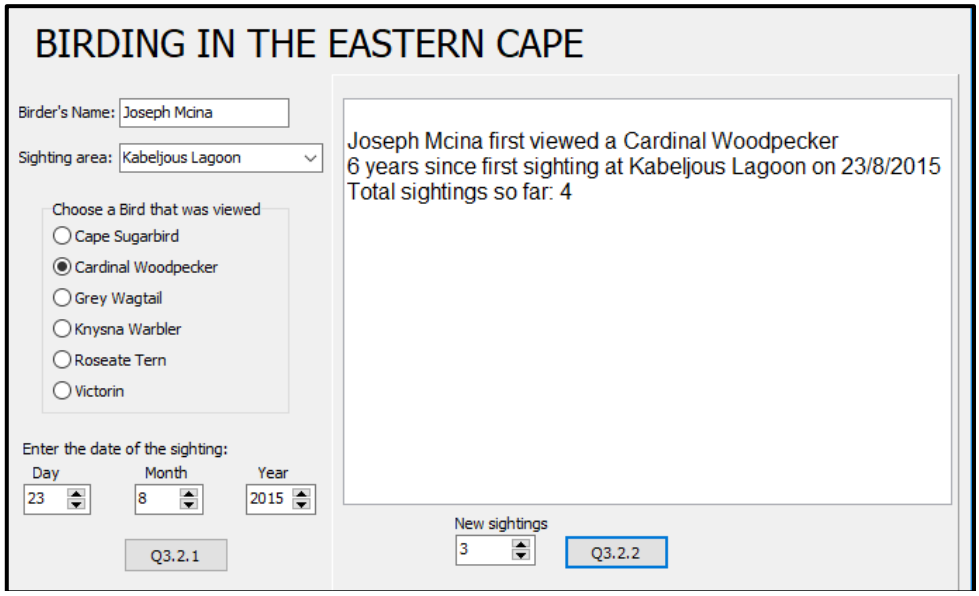
3.2.2 **Button Q3.2.2**

Write code to receive the input from the spinedit named **sedSighting.**

Use the methods of the class to do the following:

- Increase the quantity of sightings by using the value from the spinedit.
- Display the words 'Total sightings so far', as well as the quantity of sightings in the richedit named **redDisplay**.

Example of output if 3 more sightings was added to the quantity:



(5)

- Enter your name and surname as a comment in the first line of the program file. (In both the class and the main program that uses the class)
- Save your programs.
- A printout of the code of both units may be required.

[38]

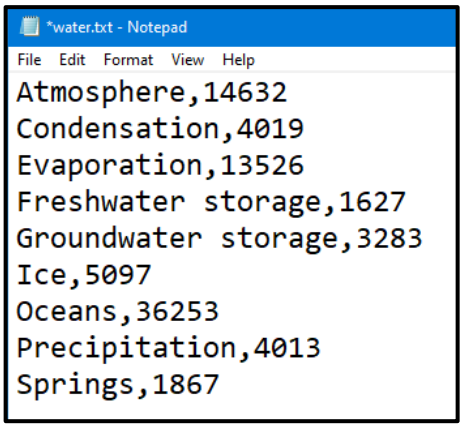
QUESTION 4: PROBLEM-SOLVING PROGRAMMING

Technically, there is no beginning point of the water cycle, but when asked 'Where does the water cycle start?', people have different answers. A survey was conducted with many people who answered this question. You will use the data from this survey to perform calculations and display the answers.

Do the following:

- Open the incomplete program in the **Question 4** folder.
- Enter your name and surname as a comment in the first line of the Question4_u.pas file.
- Compile and execute the program. Currently the program has no functionality.

Example of the text file named **water.txt**:



Complete the code for each question, QUESTION 4.1 and QUESTION 4.2.

NOTE:

- Good programming techniques must be applied in the design and coding of your solution.
- You may NOT change the code provided.

Given:
A global array named **arrTypes** has been declared.
A global array named **arrQty** has been declared.
A global array counter name **iCount** has been declared.
A global totalling variable name **iSum** has been declared.

4.1 **Button Q4.1**

Write code to do the following:

- Read lines from the text file named **water.txt** into the two arrays **arrTypes** and **arrQty** by separating the type of method and adding it to **arrTypes** and the number of people who chose that method into the array, **arrQty**.
- Calculate the total number of people who made a choice and store that into the global variable named **iSum**.
- Sort the two arrays according to the numbers from low to high in **arrQty**.
- Display both arrays in **redDisplay** into neat columns and display the total number of people who took part.

Example of output:

Atmosphere	14632
Condensation	4019
Evaporation	13526
Freshwater storage	1627
Groundwater storage	3283
Ice	5097
Oceans	36253
Precipitation	4013
Springs	1867
Total number of people = 84317	

(19)

4.2 **Button Q4.2**

Write code to calculate the percentage of each quantity in **arrQty** using the global variable **iSum.** Round the answer to no decimal places.

Example**:** percentage = quantity for a type chosen / sum of all choices x 100

Display the percentages as a symbol for each of four categories.

The FOUR categories consist of the following:

Percentages between and equal to 1 and 5.
Percentages between and equal to 6 and 10.
Percentages between and equal to 11 and 20.
Percentages between and equal to 21 and 50.

Display a symbol (x) in a neat grid to display the range of percentages for each type from **arrTypes**.

Example of output:

```
                    Percentage ranges
Type                1-5       6-10      11-20     21-50
-------------------------------------------------------
Freshwater storage  X
Springs             X
Groundwater storage X
Precipitation       X
Condensation        X
Ice                           X
Evaporation                             X
Atmosphere                              X
Oceans                                            X
```

(13)

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

[32]

TOTAL: 150