basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2022

MARKS: 150

TIME: 3 hours

This question paper consists of 24 pages with 2 data pages.

INSTRUCTIONS AND INFORMATION

1. This 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 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 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.

Do the following:

- Double click on the following password-protected executable file: **DataENGNov2022.exe**
- Click on the 'Extract' button.
- Enter the following password: **#TIME4RRR&$**

Once extracted, the following list of files will be available in the folder **DataENGNov2022**:

Question1:
Question1_P.dpr
Question1_P.dproj
Question1_P.res
Question1_U.dfm
Question1_U.pas

Question2:
CollectionDB.mdb
CollectionDB - Copy.mdb
ConnectDB_U.dcu
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question3:
logo.jpeg
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas
SolarPowerPlant_U.pas

Question4:
Quest4_P.dpr
Quest4_P.dproj
Quest4_P.res
Quest4_U.dfm
Quest4_U.pas
rrr.jpeg

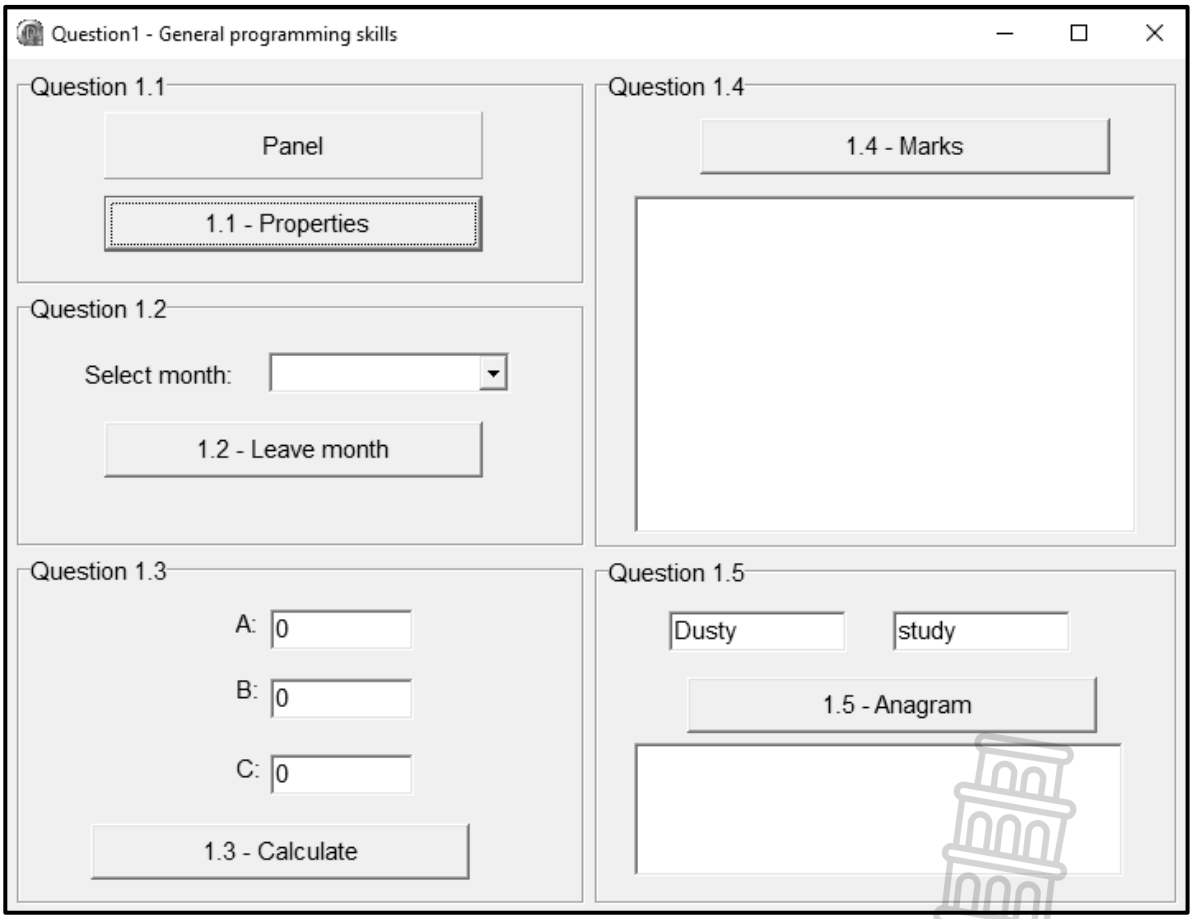
SECTION A

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.

Example of graphical user interface (GUI):



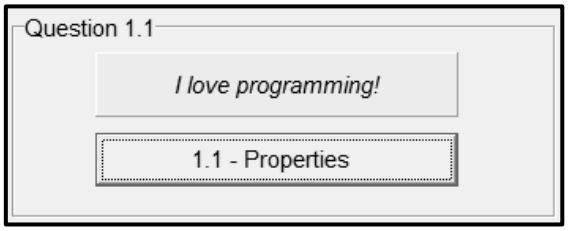
- Complete the code for each section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.5 that follow.

1.1 **Button [1.1 - Properties]**

Write code to change the properties of panel **pnlQ1_1** as follows:

- Set the colour to yellow.
- Set the font to italics.
- Change the caption to 'I love programming!'.

Example of output:



(3)

1.2 **Button [1.2 - Leave month]**

An employee selects the month during which he/she wants to take leave. The company is closed from June to August and leave cannot be taken during this period of time.

The combo box **cmbQ1_2** has been populated with the names of the months from January to December.

The user must select a month from combo box **cmbQ1_2**.

Write code to do the following:

- Extract the month selected from combo box **cmbQ1_2**.
- If the company is open in the selected month, display a message on the label **lblQ1_2** using this format:

```
"Your leave in " <month> " has been granted."
```

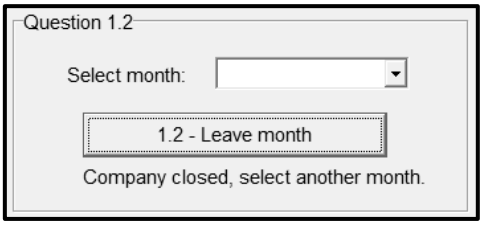
- If the company is closed during the selected month, do the following:
  - Display a message on the label **lblQ1_2** using this format:

```
"Company closed, select another month."
```

  - Clear the selection of the month from combo box **cmbQ1_2**.

Example of output if April is selected:

Example of output if July is selected:



(7)

1.3 **Button [1.3 - Calculate]**

The result of the following formula is required:

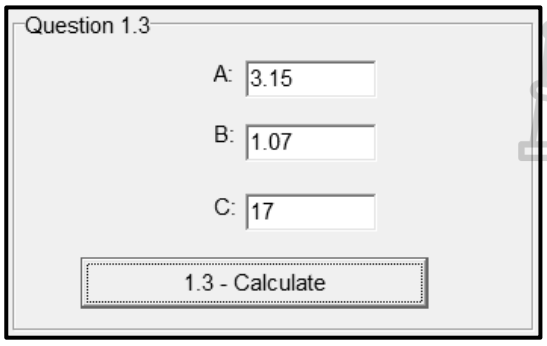
$$C = \sqrt{A^5 + \pi B^2}$$

The user must enter the values for A and B in the edit boxes provided.

Write code to do the following:

- Extract the value of A from edit box **edtQ1_3_1** and the value of B from edit box **edtQ1_3_2**.
- Use appropriate mathematical functions to calculate the value of C, using the formula provided.
- Display the truncated value of C in edit box **edtQ1_3_3**.

Example with the values of 3.15 (A) and 1.07 (B) as input and the result (C) as output:



(8)

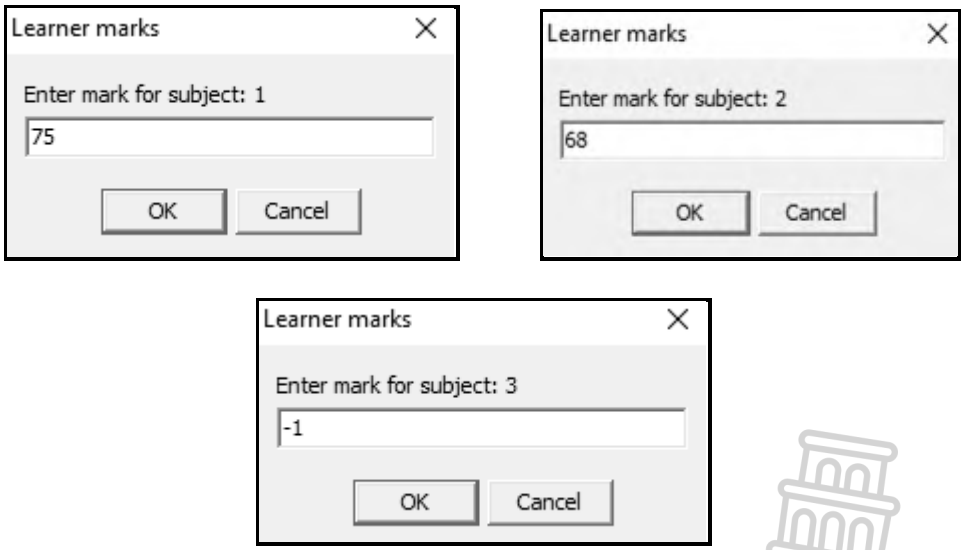
1.4 **Button [1.4 - Marks]**

The number of subjects that learners are enrolled for varies. The marks achieved by a learner for each subject must be entered, and the average mark must be calculated based on the number of marks that was entered.

Write code to do the following:

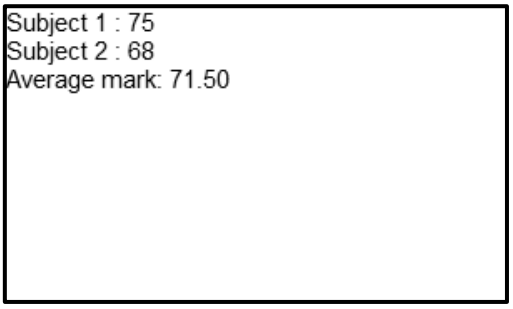
- Allow the user to enter the marks achieved by a learner per subject, starting from subject number 1.
- Use a conditional loop and an input box to enter the marks. The loop must terminate when a mark of -1 is entered.
- As marks are entered, a numbered list of subjects and marks must be displayed in the **redQ1_4** output area, as shown in the examples that follow.
- After the loop for input has been terminated, calculate the average mark achieved by the learner. Display the average mark in the **redQ1_4** output area, formatted to TWO decimal places.

Example of input of marks:

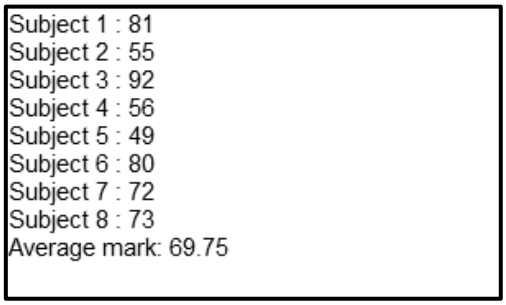


Examples of output:

Marks for two subjects and the average:



Marks for eight subjects and the average:



(11)

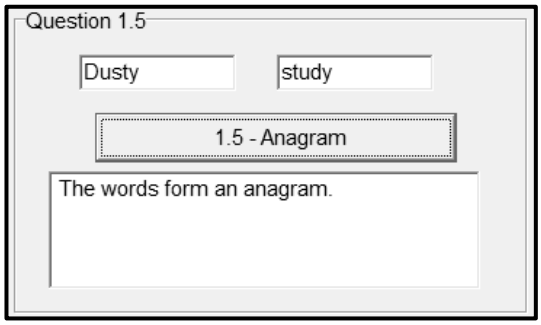
1.5 **Button [1.5 - Anagram]**

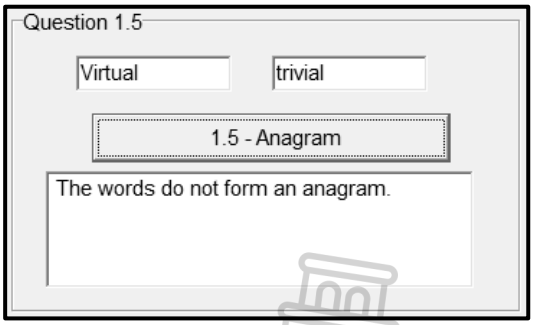
Two words must be analysed to determine whether the words form a perfect anagram or not. A perfect anagram is when another word is formed by rearranging the letters of the original word. For example, the word cat formed from the word act, peach from cheap and dusty from study.

Code has been provided to extract two words from edit boxes **edtQ1_5_1** and **edtQ1_5_2** and convert them to lower case.

Write code to determine whether the two words extracted from the edit boxes form an anagram or not and display a suitable message in memo **memQ1_5**.

Examples of input and output:





(11)

- 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 Collect-a-Can recycling initiative has developed a database called **CollectionDB.mdb**, which contains information about the collection of cans from clients for the years 2020 to 2022.

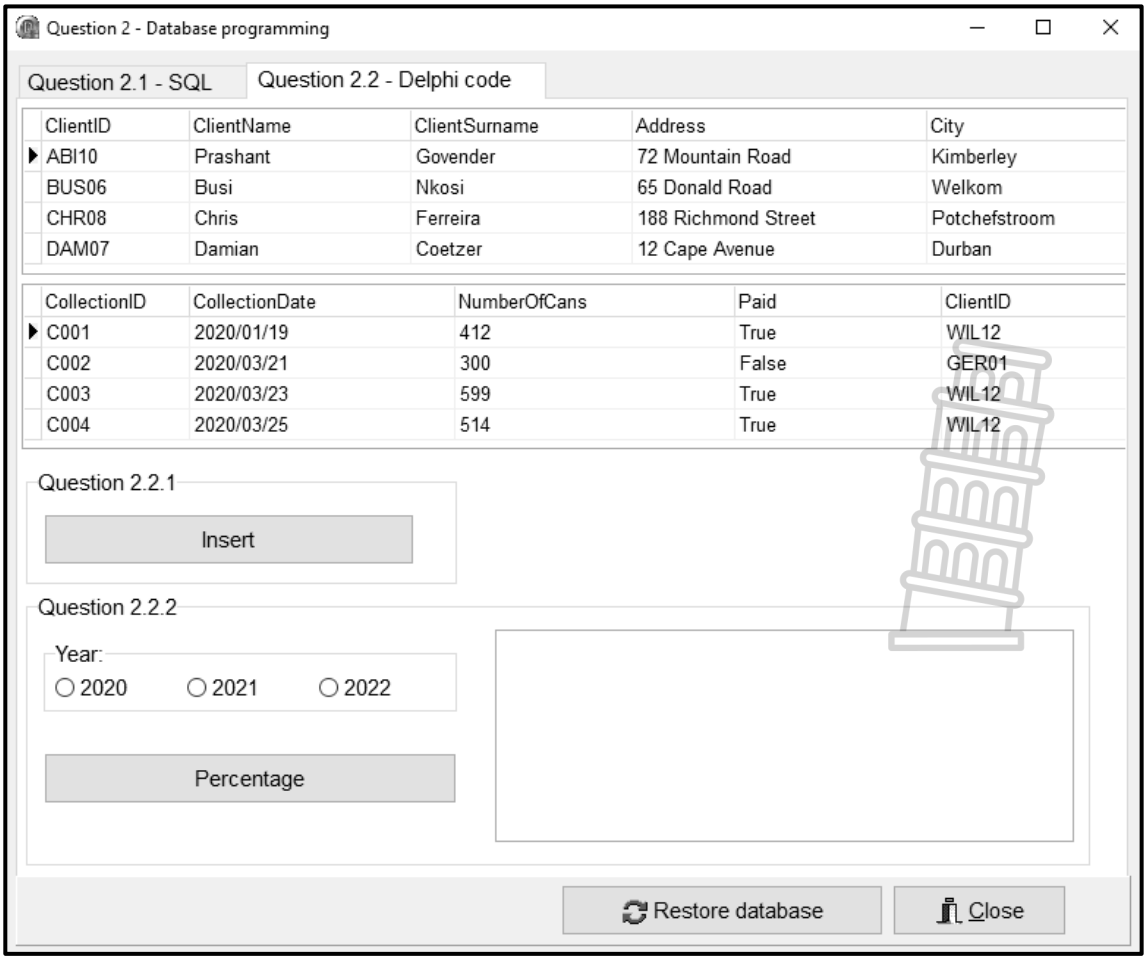
An application is required that will use the **CollectionDB.mdb** database to manage the data and payments to clients who participate in the cash-for-cans initiative.

The database contains two tables called **tblClients** and **tblCanCollection**.

The data pages attached at the end of the question paper provide information on the design of the **CollectionDB.mdb** 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 below to complete the code for each section as described in QUESTION 2.1 and QUESTION 2.2.
- Use SQL statements to answer QUESTION 2.1 and Delphi code to answer QUESTION 2.2.

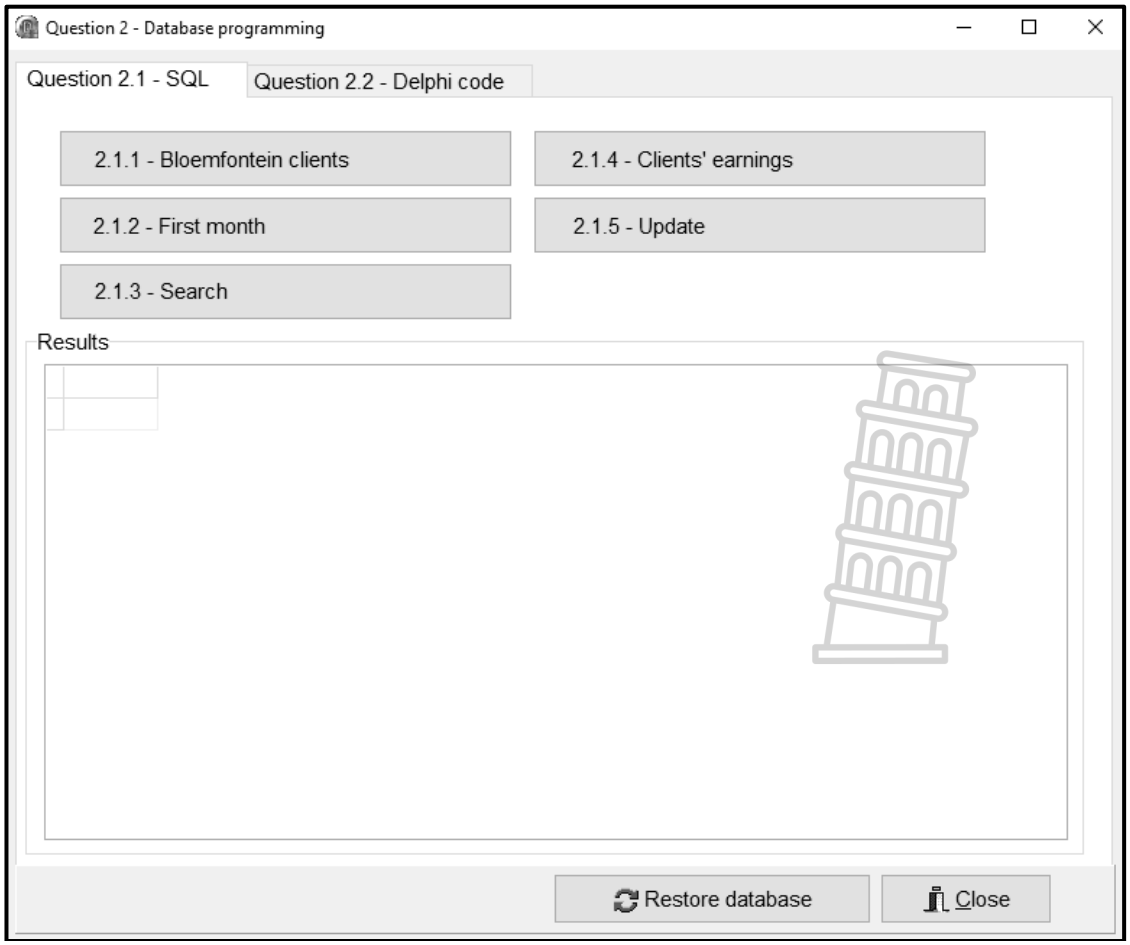
NOTE:

- 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, i.e. 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 public variables, as described in the table below.

Variable	Data type	Description
`tblClients`	`TADOTable`	Refers to the table **tblClients**
`tblCanCollection`	`TADOTable`	Refers to the table **tblCanCollection**

2.1 Tab sheet [Question 2.1 - SQL]

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



NOTE:

- Use ONLY SQL code 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 that will be assigned to the variables **sSQL1**, **sSQL2**, **sSQL3**, **sSQL4** and **sSQL5** are incomplete.

Complete the SQL statements to perform the tasks described in QUESTION 2.1.1 to QUESTION 2.1.5 below.

2.1.1 **Button [2.1.1 - Bloemfontein clients]**

Display all details of clients who live in Bloemfontein from the **tblClients** table, sorted according to the **ClientSurname** field.

Example of output:

ClientID	ClientName	ClientSurname	Address	City
JAC05	Jacob	Human	8 Human Street	Bloemfontein
PIE12	Piet	Mogorosi	5 Stormer Road	Bloemfontein
RHO09	Rhoda	Somers	14 Marilyn Way	Bloemfontein
GER01	Gert	Vermeulen	55 Dawn Street	Bloemfontein

(4)

2.1.2 **Button [2.1.2 - First month]**

Display the **CollectionID**, **CollectionDate** and **NumberOfCans** of collections made in January.

Example of output of the first five records:

CollectionID	CollectionDate	NumberOfCans
C001	2020/01/19	412
C031	2021/01/05	1000
C032	2021/01/10	450
C033	2021/01/16	2150
C034	2021/01/28	800

(3)

2.1.3 **Button [2.1.3 - Search]**

Code has been provided to enter a letter using an input box. The letter is saved in a variable called **sLetter**. Display ALL details of clients whose **ClientID** starts with the letter entered.

Example of output if the letter J was entered:

ClientID	ClientName	ClientSurname	Address	City
JOH03	Johan	Weston	43 Michellin Street	Vanderbijlpark
JAC05	Jacob	Human	8 Human Street	Bloemfontein

(4)

2.1.4 **Button [2.1.4 – Clients' earnings]**

Clients will receive R8,00 for each kilogram of cans collected. The company uses the formula: 1 kilogram = 76 cans.

Calculate and display the total amount that each client will receive for the cans they have collected, formatted to currency. Display the **ClientName** field and the total amount to receive, using the field name **Total Amount**.

Example of output of the first five records:

ClientName	Total Amount
Busi	R905.26
Chris	R1 173.47
Damian	R465.58
Gert	R661.68
Henry	R400.11

(8)

2.1.5 **Button [2.1.5 - Update]**

The details of a specific collection of cans were captured incorrectly.

Modify the details of record **C003** in the **tblCanCollection** table as follows:

- NumberOfCans: 250
- A payment was not made

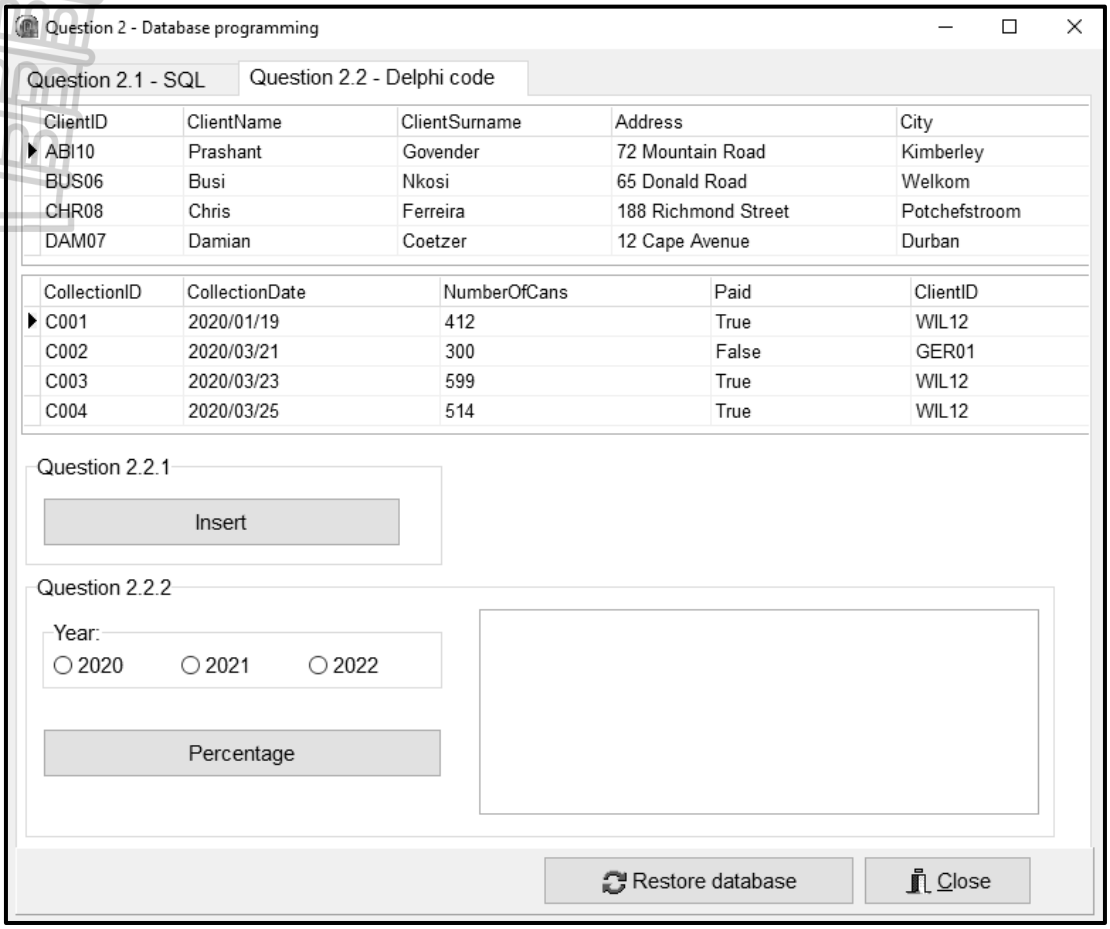
Code has been provided to display a message that indicates that a record has been changed in the database. (4)

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

NOTE:

- Use ONLY Delphi programming code to answer QUESTION 2.2.1 and QUESTION 2.2.2.
- NO marks will be awarded for SQL statements in QUESTION 2.2.

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



2.2.1 **Button [2.2.1 - Insert]**

Write code to add a new record to the **tblClients** table. The details of the client are as follows:

Client ID: **CHA01**
Client name: **Charles**
Client surname: **du Boit**
Address: **24 Van Wouw Street**
City: **Cape Town**

Example of the first four records of the **tblClients** table which shows that the record of the new client has been added successfully to the table:

ClientID	ClientName	ClientSurname	Address	City
CHA01	Charles	du Boit	24 Van Wouw Street	Cape Town
CHR08	Chris	Ferreira	188 Richmond Street	Potchefstroom
DAM07	Damian	Coetzer	12 Cape Avenue	Durban
GER01	Gert	Vermeulen	55 Dawn Street	Bloemfontein

(4)

2.2.2 **Button [2.2.2 - Percentage]**

The company wants to calculate the number of cans collected by a specific client in a specific year as a percentage of the total number of cans collected by the company in that specific year.

The user must do the following:

- Select a client from the DBGrid by clicking on the record.
- Select a year from the radio group **rgpQ2_2_2**.

Code has been provided to extract the year selected from the radio group **rgpQ2_2_2**.

Use the **redQ2_2_2** output area to display the information listed below.

Write code to do the following:

- Display the name and surname of the client selected.
- Determine and display the total number of cans collected by the client for the year selected.
- Determine and display the total number of cans collected by the Collect-a-Can company for the year selected.
- Calculate which percentage of the total number of cans collected in the selected year, was collected by the client. Display the percentage formatted to two decimal places.

Example of output if the client record with ClientID **BUS06** and the year **2022** has been selected:

Busi Nkosi	
Client collected in 2022:	2000
Company collected in 2022:	21660
Percentage collected by client:	9.23%

Example of output if the client record with ClientID **CHR08** and the year **2021** has been selected:

Chris Ferreira	
Client collected in 2021:	5268
Company collected in 2021:	32687
Percentage collected by client:	16.10%

(13)

- 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

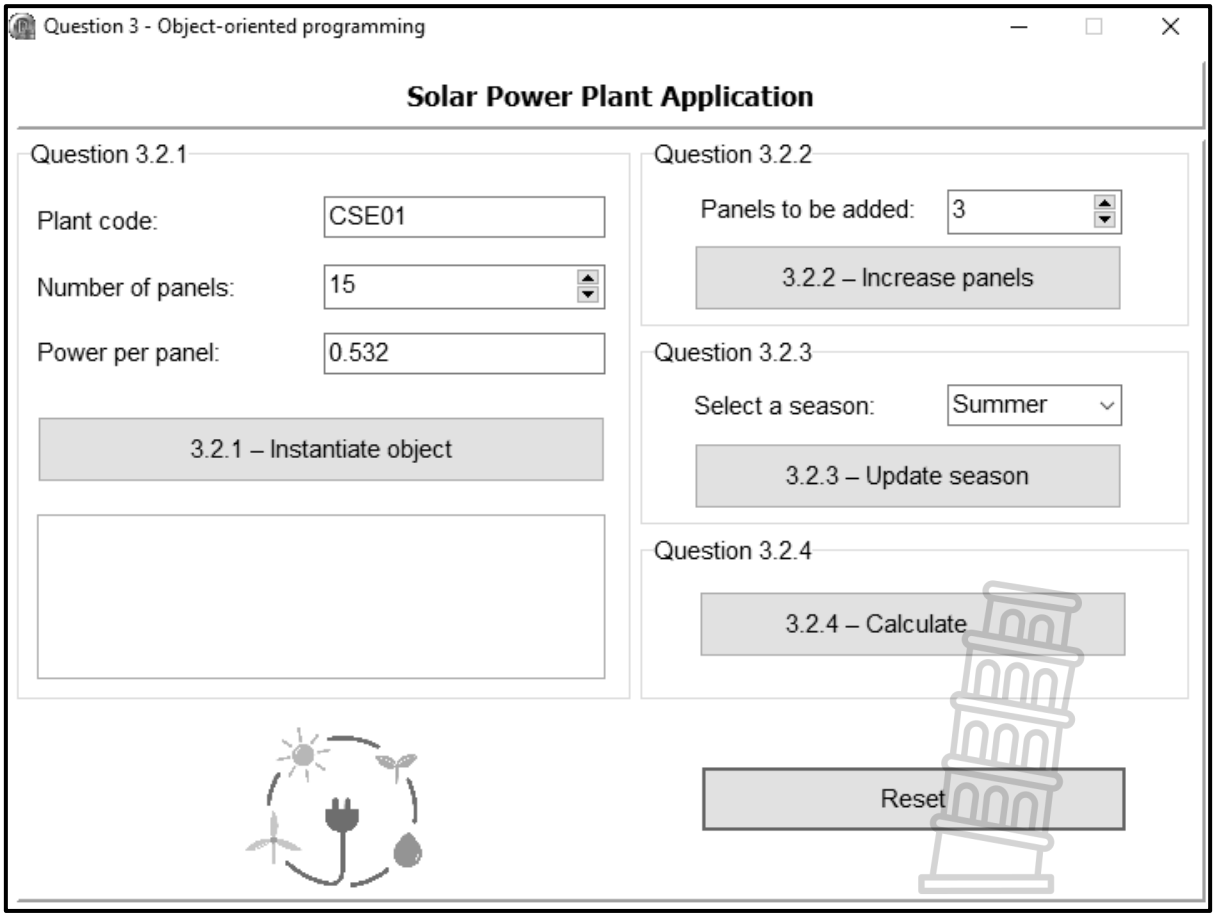
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

The sustainable energy sector wants to keep track of the solar power plants in the country and their capacity to increase the power that they generate.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Open the incomplete object class **SolarPowerPlant_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question3_U.pas** file and the **SolarPowerPlant_U.pas** file.
- Compile and execute the program. The program has limited functionality currently.

Example of graphical user interface (GUI):



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

NOTE: You are NOT allowed to add any additional attributes or user-defined methods, unless explicitly stated in the question.

3.1 The provided incomplete object class (**TSolarPowerPlant**) contains the declaration of four attributes which describe a **SolarPowerPlant** object.

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

Attribute	Description
fPlantCode	A unique code for the solar power plant
fNumberOfPanels	The number of panels installed at the plant
fPowerPerPanel	The maximum power per panel in kilowatt-hour (kWh)
fSeason	The season during which power will be generated – Summer, Autumn, Winter, Spring

Code has been provided for the following accessor methods:

- **getPlantCode** to return the fPlantCode attribute
- **getNumOfPanels** to return the fNumberOfPanels attribute
- **getSeason** to return the fSeason attribute

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 that will receive the plant code, the number of panels and the power per panel as parameters. Assign the parameter values to the respective attributes. Assign the default value 'Summer' to the season attribute. (5)

3.1.2 Write the code for a method called **incNumOfPanels** that receives a value as parameter and increments the **fNumberOfPanels** attribute by the value received. (4)

3.1.3 Write code for a mutator method called **setSeason** that receives a value as a parameter and sets the **fSeason** attribute to the value received. (3)

3.1.4 Write a method called **calculateCapacity** that uses the information that follows to calculate the power generation capacity of the installed panels at the company in the current season. Return the result as a real value.

The season determines the hours of sunlight per day that can be used to generate solar power, as shown in the following table:

SEASON	HOURS OF SUNLIGHT PER DAY
Summer	10
Autumn	8
Winter	6
Spring	8

The formula to calculate the generation capacity (GC) of the solar power plant is as follows:

$$GC = NumberOfPanels \times PowerPerPanel \times HoursPerDay$$ (8)

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

```
Plant code: <PlantCode>
Number of panels: <NumberOfPanels>
Power per panel: <PowerPerPanel>
Season: <Season>
```
(3)

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 **objPlant**.

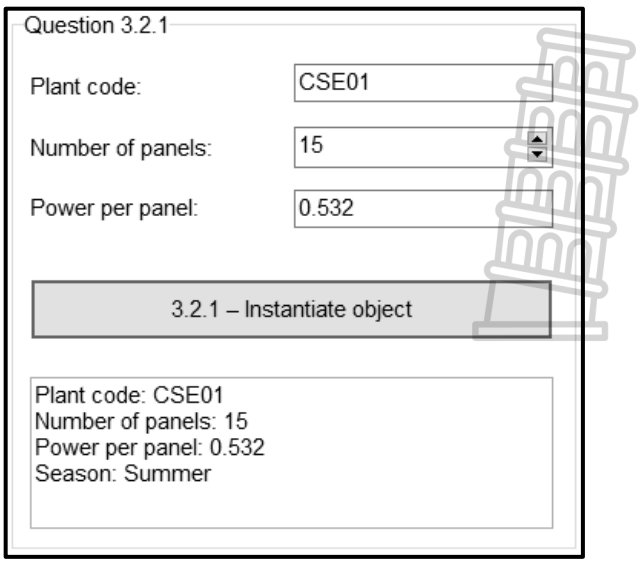
Write code to perform the tasks described in QUESTION 3.2.1 to QUESTION 3.2.4 below.

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

Write code to do the following:

- Extract the plant code from the edit box **edtQ3_2_1_Code**, the number of panels from the spin edit **sedQ3_2_1** and the power per panel from the edit box **edtQ3_2_1_Power**.
- Use the information to instantiate a new **SolarPowerPlant** object.
- Use the **toString** method to display the information of the **SolarPowerPlant** object in the rich edit **redQ3**.

Example of input and output:



(6)

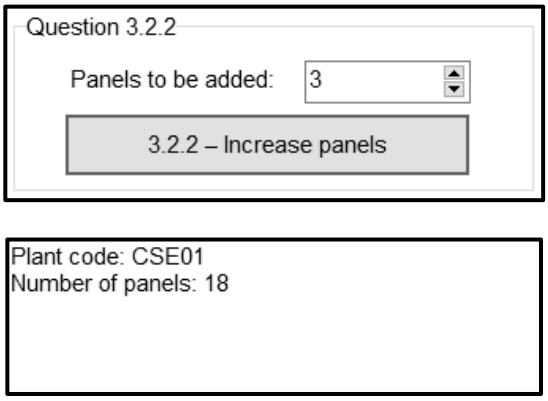
3.2.2 **Button [3.2.2 - Increase panels]**

As the need for more power arises, panels can be added to increase the plant's power generation capacity.

Write code to do the following:

- Extract the value of the number of panels to be added from the spin edit **sedQ3_2_2**.
- Call the **incNumOfPanels** method with the value from the spin edit as argument.
- Call the relevant object methods to display the following in the rich edit **redQ3**:
    - The plant code
    - The number of panels after increasing the attribute value

Example of input and output:



(4)

3.2.3 **Button [3.2.3 - Update season]**

The number of hours for the generation of power differs per season.

The user must select the season from the combo box **cmbQ3_2_3**.

Write code to do the following:

- Extract the season selected from the combo box **cmbQ3_2_3**.
- Call the **setSeason** method using the value from the combo box as an argument.
- Use the **toString** method to display the information of the updated **Plant** object in the rich edit **redQ3**.

Example of input and output:


```
Plant code: CSE01
Number of panels: 18
Power per panel: 0.532
Season: Winter
```

(3)

3.2.4 **Button [3.2.4 - Calculate]**

The maximum capacity of power that can be generated by the panels must be calculated.

Write code to display the following in the rich edit **redQ3**:

- The season as a part of the output string.
- The result of the **calculateCapacity** method converted into a string. The unit 'kW' must also be displayed.

Example of output:

```
The maximum generation capacity per day in
Winter:
57.456 kW
```

(4)

- 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 in the object class and the form class if required.

TOTAL SECTION C: 40

SECTION D

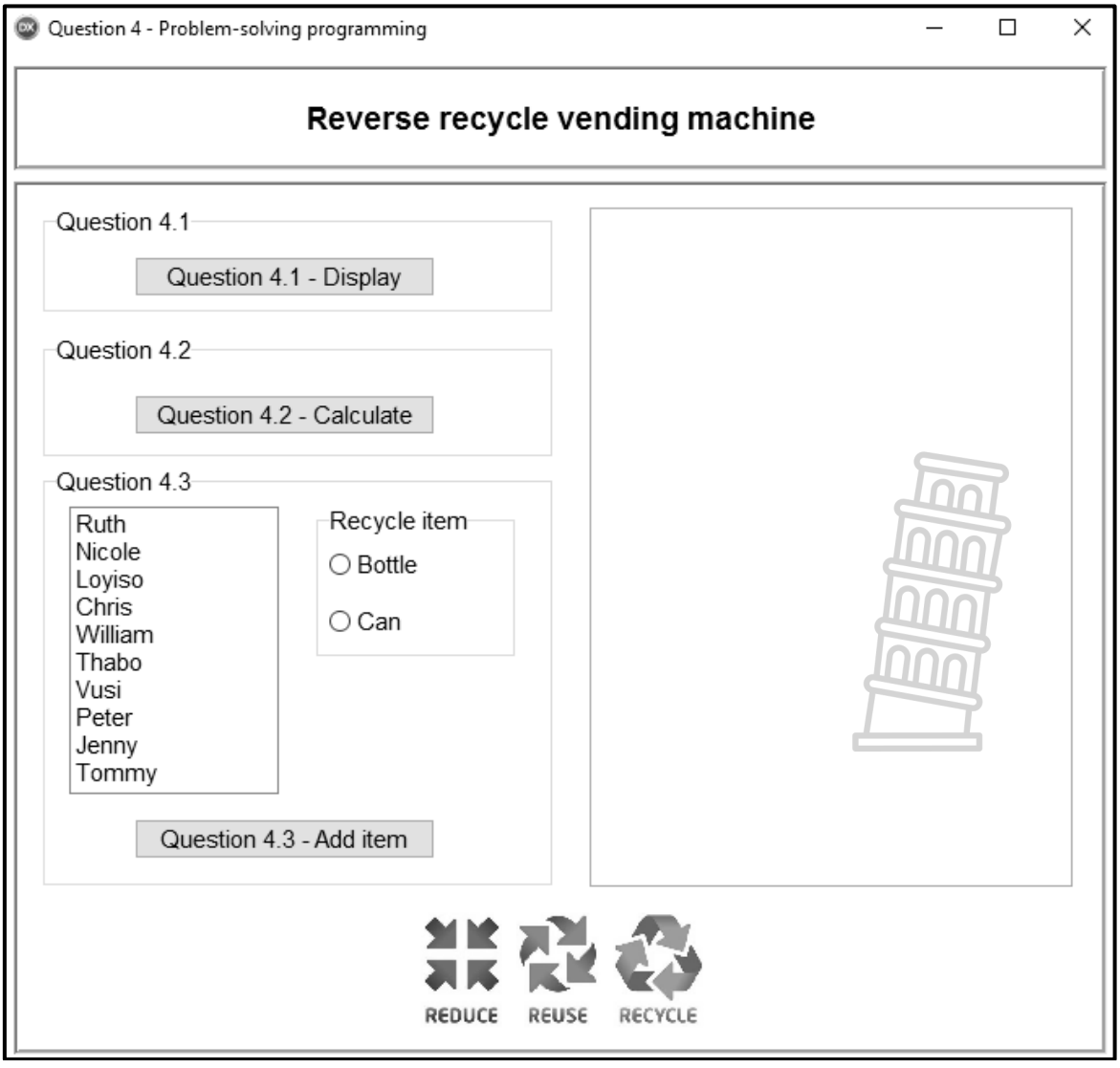
QUESTION 4: PROBLEM-SOLVING PROGRAMMING

Vending machines can be used to dispense items such as sweets, chips, cool drinks, etc. A reverse vending machine can be used for recycling purposes. Your school wants to use reverse vending machines on the school grounds to recycle bottles and cans. A learner will deposit/insert a number of bottles or cans into the reverse vending machine and will then be remunerated. One of the reverse vending machines will be tested in a class with 10 learners.

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 **Quest4_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

Example of graphical user interface (GUI):



The following arrays have been provided in the program:

- An array **arrNames** which contains the names of ten learners in a class:

```
arrNames: array [1 .. 10] of String = (
  'Ruth', 'Nicole', 'Loyiso', 'Chris', 'William',
  'Thabo', 'Vusi', 'Peter', 'Jenny', 'Tommy');
```

- A two-dimensional array, **arrVending**, that stores the type of items that were recycled by the learners. The array is partially populated and allows for up to 15 items to be recycled by each of the 10 learners in the class:

```
arrVending: array [1 .. 10, 1 .. 15] of String =
  (('C', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
   ('B', 'B', 'B', 'C', 'C', 'C', 'B', 'B', 'B', 'C', 'C', 'C', 'C',
'C', ''),
   ('', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
   ('C', 'C', '', '', '', '', '', '', '', '', '', '', '', '', ''),
   ('B', 'B', 'C', 'C', 'B', 'B', 'C', 'C', 'C', 'C', 'B', 'C', 'C',
'B', ''),
   ('C', 'C', 'B', '', '', '', '', '', '', '', '', '', '', '', ''),
   ('C', 'B', '', '', '', '', '', '', '', '', '', '', '', '', ''),
   ('B', 'B', '', '', '', '', '', '', '', '', '', '', '', '', ''),
   ('C', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
   ('B', 'C', '', '', '', '', '', '', '', '', '', '', '', '', ''));
```

Complete the code for each section of QUESTION 4, as described in QUESTION 4.1 to QUESTION 4.3 below.

4.1 **Button [4.1 - Display]**

The names of learners are provided in array **arrNames**. A corresponding list of items that have been recycled for each learner, has been provided in the array **arrVending**.

The items in array **arrVending** are represented using the letters `'B'` or `'C'` where `'B'` represents a bottle and `'C'` represents a can.

Code has been provided to display the heading.

Write code to display the content of arrays **arrNames** and **arrVending** in the following format:

```
<Name><tab><recycled item><recycled item><recycled item>...
```

Example of output:

```
----------------------------
Names       Items recycled
----------------------------
Ruth        C
Nicole      BBBCCCBBBCCCCC
Loyiso
Chris       CC
William     BBCCBBCCCCBCCB
Thabo       CCB
Vusi        CB
Peter       BB
Jenny       C
Tommy       BC
```

(5)

4.2 **Button [4.2 - Calculate]**

Learners using the vending machine for recycling items will be paid for each item as follows:

ITEM RECYCLED	AMOUNT PAID
Bottle (B)	R2.15
Can (C)	R0.75

Write code to display the following in the rich edit **redQ4**:

- The name of each learner and the total amount each learner will be paid for the bottle and can items that the learner put into the vending machine.
- The name(s) of the learner(s) that will receive the highest amount.

Example of output:

```
----------------------------
Names       Total amount paid
----------------------------
Ruth           R0.75
Nicole        R18.90
Loyiso         R0.00
Chris          R1.50
William       R18.90
Thabo          R3.65
Vusi           R2.90
Peter          R4.30
Jenny          R0.75
Tommy          R2.90
----------------------------
Highest payout(s):
----------------------------
Nicole        R18.90
William       R18.90
```

(14)

4.3 **Button [4.3 - Add item]**

The user must do the following when a learner wants to add an item to be recycled:

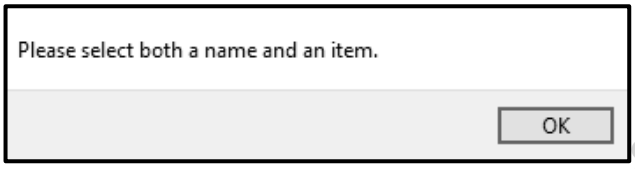
- Select the name of the learner from the list box **lstQ4**.
- Select the type of item to be added for recycling from the radio group **rgpQ4**.

The bottle or can item selected must then be added to the next available empty space in array **arrVending** at the index corresponding to the name of the learner selected. A suitable message must be displayed if the vending space for the specified learner is full.

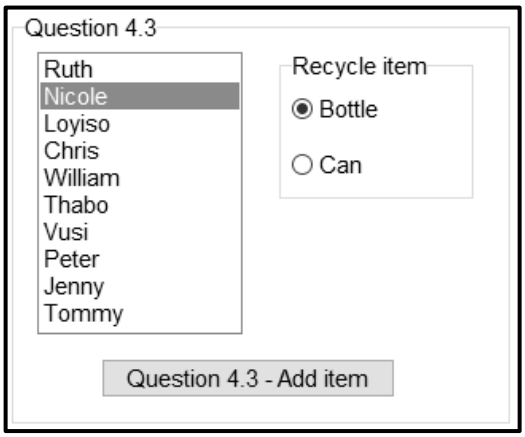
Write code to do the following:

- Extract the name of the learner from the list box **lstQ4** and the item that needs to be recycled from the radio group **rgpQ4**.
- Display a suitable message when neither the name nor the item has been selected.
- If there is an empty space available in the array for the specified learner:
  - Add the recycled item selected (B or C) to the array **arrVending** at the correct index for the learner selected.
  - Display the contents of the array **arrNames** and the updated array **arrVending** after each item has been added.
- If there is no space available in the array for the specified learner, display a suitable message to indicate that the vending machine is full.

Example of output when a name or item has not been selected:

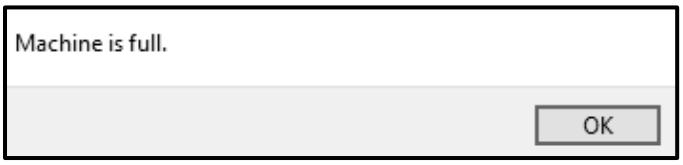


Example of output when the name 'Nicole' and the item 'Bottle' were selected:




```
----------------------------
Names       Items recycled
----------------------------
Ruth        C
Nicole      BBBCCCBBBCCCCCB
Loyiso
Chris       CC
William     BBCCBBCCCCBCCB
Thabo       CCB
Vusi        CB
Peter       BB
Jenny       C
Tommy       BC
```

Example of output when an attempt was made to add another item for the name 'Nicole':



(11)

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

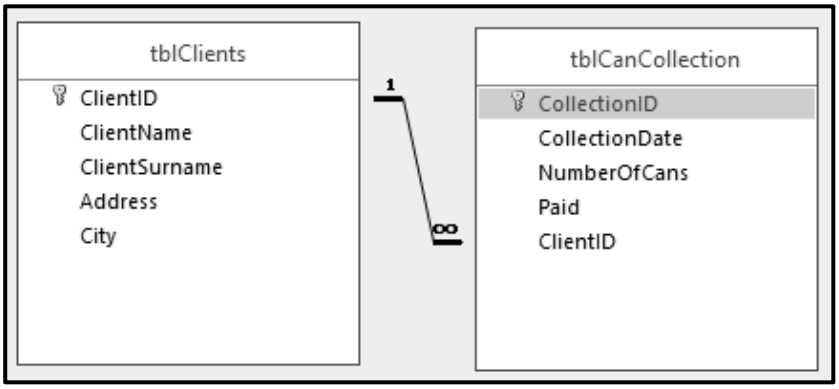
TOTAL SECTION D: 30
GRAND TOTAL: 150

INFORMATION TECHNOLOGY P1

DATABASE INFORMATION QUESTION 2:

The database **CollectionDB** consists of table **tblClients** and **tblCanCollection**.

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



The design of the database tables is as follows:

Table: **tblClients**

This table contains details of the clients.

Field name	Data type	Description
ClientID	Text (5)	Unique ID for the client
ClientName	Text (15)	The name of the client
ClientSurname	Text (15)	The surname of the client
Address	Text (20)	The address of the client used for the pickup of cans
City	Text (15)	The city where the client resides

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

ClientID	ClientName	ClientSurname	Address	City
ABI10	Prashant	Govender	72 Mountain Road	Kimberley
BUS06	Busi	Nkosi	65 Donald Road	Welkom
CHR08	Chris	Ferreira	188 Richmond Street	Potchefstroom
DAM07	Damian	Coetzer	12 Cape Avenue	Durban
GER01	Gert	Vermeulen	55 Dawn Street	Bloemfontein
HEN11	Henry	Marques	1 Kingsley Drive	Cape Town
JAC05	Jacob	Human	8 Human Street	Bloemfontein
JOH03	Johan	Weston	43 Michellin Street	Vanderbijlpark
PHI04	Phillip	Brown	11 Park Road	Sasolburg
PIE12	Piet	Mogorosi	5 Stormer Road	Bloemfontein
RHO09	Rhoda	Somers	14 Marilyn Way	Bloemfontein
WIL12	Willem	de Wit	2 Arrow Street	Sasolburg

Table: **tblCanCollection**

This table contains information of all the collections.

Field name	Data type	Description
CollectionID	Text (5)	Unique code for each collection
CollectionDate	Date/Time	Date of the collection
NumberOfCans	Number	Integer value that indicates the number of cans collected
Paid	Boolean	A Boolean value that indicates that the client received payment for the cans collected
ClientID	Text (5)	The ID of the client who collected the cans

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

CollectionID	CollectionDate	NumberOfCans	Paid	ClientID
C001	2020/01/19	412	☑	WIL12
C002	2020/03/21	300	☐	GER01
C003	2020/03/23	599	☑	WIL12
C004	2020/03/25	514	☑	WIL12
C005	2020/03/26	1200	☐	CHR08
C006	2020/03/30	480	☐	DAM07
C007	2020/04/02	511	☑	GER01
C008	2020/04/15	200	☐	BUS06
C009	2020/04/17	419	☐	DAM07
C010	2020/04/24	500	☑	CHR08

NOTE:

- Connection code has been provided.
- The database is password-protected, therefore you will not be able to access the database directly.

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2022

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 29 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 11) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 12 to 29) contain examples of solutions for Question 1 to Question 4 in programming code.

- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 11) 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	**Button - [1.1 – Properties]** Set the panel colour to yellow ✓ Set the font style to italic ✓ Change caption to ”I love programming!” ✓	**3**	
1.2	**Button - [1.2 – Leave month]** Extract month name / month index from cmbQ1_2 ✓ Check IF (sMonth = 'June') ✓ OR (sMonth = 'July') OR (sMonth = 'August') ✓ then Display message 'Company closed, select another month.' ✓ Deselect month from cmbQ1_2 (Index -1) ✓ Else ✓ Display message 'Your leave in ' + sMonth + ' has been granted.' ✓ ALTERNATIVE IF conditions: • IF iMonth IN [5,6,7] then • IF (iMonth >= 5) AND (iMonth <= 7) then **NOTES:** If candidates adjust the index according to month numbers, the range must be from 6 to 8	**7**	

1.3	**Button - [1.3 - Calculate]** Extract A and B from edit boxes✓ converted to real ✓ `C := sqrt✓(power(A,5)✓ + pi✓ * sqr(B)✓);` Alternative to functions: • Sqrt or Power(Value, 0.5) • Power (do not accept hard-coding) • Pi or 22/7 or 3.14 • Sqr or Power(B, 2) or (B * B) Display Truncated (Floor or Trunc) ✓ value of C converted to String ✓ **NOTE:** Penalise once only for a logical mistake	**8**	
1.4	**Button - [1.4 – Marks]** Initialise counter and total ✓ Input mark ✓ Start conditional loop✓ to test mark <> -1 ✓ Increase counter variable ✓ Add mark to total mark ✓ Display the subject number ✓ and mark ✓ Input next mark ✓ Calculate average mark ✓ Display average mark in output area converted to String formatted to TWO decimal places ✓ **NOTES:** If repeat until used the condition will be: until mark = -1 If a single input is used inside the loop, an IF must be used inside the loop to prevent -1 to be accepted as a mark	**11**	

1.5	**Button - [1.5 – Anagram]** If length of two word strings are equal ✓ Loop ✓ from 1 to length of word 1 string ✓ Use character at index ✓ Find position ✓ of character in word 2 ✓ Delete character at position ✓ from word 2 ✓ If length of word 2 is null ✓ Display the message "The words form an anagram." ✓ Else Display a message "The words do not form an anagram." ✓ **Concepts**: Loop through length of first word //2 Extract character at index //1 Find position of character in second word //2 <u>Mechanism to match words</u>: //3 • Overwrite char at index word1 and char position in word2 with space • Test and change flag if Not found (flag set before loop) • Delete character from second word (if length tested same before the loop) <u>Test anagram passed and display messages</u>: //3 IF words are equal / word 2 = '' // flag still true ShowMessage IS anagram ShowMessage NOT anagram **Sorting as an alternative**: Sort characters in first word: Double looping //2 Comparing characters within the word //2 Swopping characters //3 Sorting characters in second word //1 <u>Test and display messages</u>: //3 IF words are equal ShowMessage IS anagram ShowMessage NOT anagram	**11**	
	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 – Bloemfontein clients]**
	`SELECT * ✓`
	`FROM tblClients ✓`
	`WHERE City = "Bloemfontein" ✓`
	`ORDER BY ClientSurname ✓`
		4	

2.1.2	**Button [2.1.2 – First month]**
	`SELECT CollectionID, CollectionDate,`
	`NumberOfCans FROM tblCanCollection ✓`
	`WHERE Month ✓ (CollectionDate) = 1 ✓`
	
	`Alternative:`
	`Extracting month using String operators`
		3	

2.1.3	**Button [2.1.3 – Search]**
	`SELECT * FROM tblClients ✓`
	`WHERE ClientID LIKE ✓ "'+ sLetter ✓ +'%" ✓`
	
	`Alternative:`
	`WHERE left(ClientID, 1) = quotedStr(sLetter)`
		4	

2.1.4	**Button [2.1.4 – Clients' earnings]**
	`SELECT ClientName,`
	`FORMAT(SUM ✓ (NumberOfCans /76 * 8) ✓,`
	`"Currency") ✓`
	`AS ✓ [Total Amount] ✓`
	`FROM tblCanCollection A, tblClients B ✓`
	`WHERE A.ClientID = B.ClientID ✓`
	`GROUP BY ClientName ✓`
		8	

2.1.5	**Button [2.1.5 – Update]**
	`UPDATE tblCanCollection ✓`
	`SET NumberOfCans = 250 ✓, Paid = False ✓`
	`WHERE CollectionID = "C003" ✓`
		4	

	Subtotal:	**23**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**DATABASE MANIPULATION using Delphi code**		
2.2.1	**Button [2.2.1 – Insert]** Insert/append mode ✓ Assign to correct field names ✓ Using the correct values for each field ✓ Post ✓ tblClients.Insert; tblClients['ClientID'] := 'CHA01'; tblClients['ClientName'] := 'Charles'; tblClients['ClientSurname'] := 'du Boit'; tblClients['Address'] := '24 Van Wouw Street'; tblClients['City'] := 'Cape Town'; tblClients.Post;	**4**	
2.2.2	**Button [2.2.2 – Percentage]** Display the name and surname of the client selected ✓ Initialise variables ✓ Go to the first record in the tblCanCollection table ✓ Use a loop to step through the tblCanCollection table ✓ Test year = year selected ✓ Increment company total of collections for the year ✓ if (ClientID in tblCanCollection = ClientID in tblClients) ✓ Increment client total of collections for the year ✓ Move to the next record in tblCanCollection ✓ *End loop (tblCanCollection)* Calculate percentage of client collection as part of the total collection of company ✓ (rPercentage := rTotalClient / rTotalCompany * 100) Display total client collection value formatted to String ✓ Display company collection value formatted to String ✓ Display percentage of client collection formatted to two decimals with % symbol added to result ✓	**13**	
	Subtotal:	**17**	
	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	**Constructor Create** Constructor heading with three correct parameters ✓and correct data types for parameters ✓ Assign parameter values to the three attributes ✓✓ (fPlantCode, fNumberOfPanels, fPowerPerPanel) Set season attribute to "Summer" ✓	**5**	
3.1.2	**procedure incNumOfPanels** Procedure heading ✓ with integer parameter ✓ fNumberOfPanels := fNumberOfPanels ✓ + parameter ✓	**4**	
3.1.3	**procedure setSeason** procedure heading ✓ with string type parameter ✓ fSeason = parameter ✓	**3**	
3.1.4	**function calculateCapacity** function heading with real data type ✓ Test if fSeason = 'Summer' ✓ hours = 10 ✓ else if fSeason = 'Winter' ✓ hours = 6 ✓ else hours = 8 ✓ Result = ✓ fNumberOfPanels * fPowerPerPanel * hours ✓	**8**	
3.1.5	**function toString** Function declared with string as return data type ✓ All attributes part of string to return ✓ In correct format with text and with #13 ✓	**3**	
	Subtotal: Object class	**23**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 – Instantiate object]** Extract the data from the components ✓ converted to correct data types ✓ objPlant := ✓ TSolarPowerPlant.create ✓ Use three arguments ✓ `(PlantCode,NumberOfPanels,PowerPerPanel)` Display information in the rich edit using toString method ✓	**6**	
3.2.2	**Button [3.2.2 – Increase panels]** Call incNumOfPanels method ✓ using value from the spin edit as an argument ✓ Display plant code using getPlantCode method ✓ and number of panels using getNumOfPanels method, converted to string ✓	**4**	
3.2.3	**Button [3.2.3 – Update season]** Call setSeason method ✓ using the season selected from the combo box as an argument ✓ Display information in the rich edit using toString method ✓	**3**	
3.2.4	**Button [3.2.4 – Calculate]** Display a suitable description ✓ including the season, using the getSeason method ✓ Display the result of the calculateCapacity method ✓, converted to string with "kW" added as unit ✓	**4**	
	Subtotal Form class:	**17**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING

CENTRE NUMBER:	**EXAMINATION NUMBER:**

QUESTION	**DESCRIPTION**	**MAX MARKS**	**LEARNER'S MARKS**
4.1	**Button [4.1 – Display]** Outer loop `I` from 1 to length of array ✓ Or 10 Start output string with name[`I`] ✓ and add tab (#9) to output string Inner loop `J` from 1 to length of arrVending[`J`] ✓ OR 15 Join characters from arrVending to output string ✓ Display output string on rich edit ✓	**5**	
4.2	**Button [4.2 – Calculate]** Initialise Max variable to 0 ✓ Outer loop `I` from 1 to length of arrNames (10) ✓ Initialise amount paid to 0 ✓ Inner loop `J` from 1 to length of arrVending[`I`] (15) ✓ Test if item recycled is a bottle ✓ Increment amount paid with 2.15 ✓ Test if item is a can Increment amount paid with 0.75 ✓ *End inner loop* Display name and amount paid, converted to currency ✓ **Alternative 1**: Test if Amount >= Max ✓ if Amount > Max ✓ Set Max to amount ✓and clear output ✓ Add name and amount to Max output string ✓ *End outer loop* Display Max output string in redQ4 ✓ **Alternative 2 (using parallel array)**: Test if Amount > Max //1 Set Max to amount //1 Save learner amount in parallel array //1 *End outer loop* Loop from 1 to 10 //1 If Amount = Max //1 Display name and amount //1	**14**	

4.3	**Button [4.3 – Add item]** Test if name OR item are NOT selected ✓ Display message if name or item not selected ✓ Extract item from component ✓ Extract OR determine row(index) of name ✓ <u>PLACING ITEM</u>: Loop index from 1 to 15 ✓ Test if arrVending empty at index ✓ Place item in arrVending ✓ Flag / Break ✓ Display updated array ✓ <u>NO SPACE MESSAGE</u>: Check flag OR use another mechanism to determine availability of space ✓ Display message for no space available✓	**11**	

	TOTAL SECTION D: **GRAND TOTAL:**	**30** **150**	

SUMMARY OF LEARNER'S MARKS:

CENTRE 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                3 marks
// ============================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  // Question 1.1

   pnlQ1_1.Color := clYellow;
   pnlQ1_1.Font.Style := [fsItalic];
   pnlQ1_1.Caption := 'I love programming!';
end;
//============================================================
// Question 1.2                7 marks
// ============================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
begin
  // Question 1.2
  if (cmbQ1_2.ItemIndex > 4) AND (cmbQ1_2.ItemIndex < 8) then
    begin
      lblQ1_2.Caption :='Company closed, select another
                         month.';
      cmbQ1_2.ItemIndex := -1;
    end
  else
    lblQ1_2.Caption :='Your leave in ' + cmbQ1_2.text +
                                ' has been granted.';
end;

// ============================================================
// Question 1.3                 8 marks
// ============================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  rA,rB,rC : Real;
begin
  // Question 1.3
  rA := StrToFloat(edtQ1_3_1.Text);
  rB := StrToFloat(edtQ1_3_2.Text);
  rC := Sqrt(Power(rA,5)+ pi * Sqr(rB));

  edtQ1_3_3.Text := IntToStr(Trunc(rC));
end;

// ============================================================
// Question 1.4             11 marks
// ============================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
   iMark, i, iTotal : Integer;
   rAverage : Real;
begin
  // Provided code
  redQ1_4.Clear;
```

```
  // Question 1.4
  iMark := StrToInt(InputBox('Learner marks','Enter mark for
                             subject : 1',''));
  i := 0;
  iTotal := 0;
  while iMark > -1 do
    begin
      Inc(i);
      iTotal := iTotal + iMark;
      redQ1_4.Lines.Add('Subject ' + IntToStr(i) + ' : ' +
                         IntToStr(iMark));
      iMark := StrToInt(InputBox('Learner marks','Enter mark for
                        subject : ' + IntToStr(i+1), ''));
    end;
  rAverage:= iTotal / i;
  redQ1_4.Lines.Add('Average mark:' +
                      FloatToStrF(rAverage,ffFixed,5,2));
end;

// ========================================================================
// Question 1.5              11 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  sWord1,sWord2 : String;
  I : integer;
begin
  //Provided code
  redQ1_5.Clear;
  sWord1 := Lowercase(edtQ1_5_1.Text);
  sWord2 := Lowercase(edtQ1_5_2.Text);

  // Question 1.5

  if length(sWord1) = length(sWord2) then
  begin
    for i := 1 to length(sWord1) do
      delete(sWord2,(pos(sWord1[i],sWord2)),1);
    if sWord2 = '' then
      memQ1_5.Lines.add( 'The words form an anagram.')
    else
      memQ1_5.Lines.add('The words do not form an anagram.');
  end
  else
    memQ1_5.Lines.add('The words do not form an anagram.');
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, DB, ExtCtrls, StdCtrls,
  ComCtrls, Grids, DBGrids, Buttons, ADODB, Math, DateUtils,
  ConnectDB_U;

type
  TfrmQuestion2 = class(TForm)
    pgcDBAdmin: TPageControl;
    tabsQ2SQL: TTabSheet;
    btnQ2_1_5: TBitBtn;
    grpresults: TGroupBox;
    btnQ2_1_1: TBitBtn;
    btnQ2_1_2: TBitBtn;
    b: TTabSheet;
    grpQ2_2_1: TGroupBox;
    redQ2_2_2: TRichEdit;
    btnQ2_2_2: TButton;
    dbgClients: TDBGrid;
    dbgCollections: TDBGrid;
    grpQ2_2_2: TGroupBox;
    btnQ2_2_1: TButton;
    btnQ2_1_4: TBitBtn;
    pnlBtns: TPanel;
    bmbClose: TBitBtn;
    bmbRestoreDB: TBitBtn;
    dbgrdSQL: TDBGrid;
    btnQ2_1_3: TButton;
    rgpQ2_2_2: TRadioGroup;
    cmbQ2_2_2: TComboBox;
    lblQ2_2_2: TLabel;
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure bmbRestoreDBClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion2: TfrmQuestion2;
  dbCONN : TConnection;
```

```
// --- Global variables provided ---
  tblClients, tblCanCollection : TADOTable;

implementation

{$R *.dfm}

procedure TfrmQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
// Restores the Database
  dbCONN.RestoreDatabase;
  redQ2_2_2.Clear;
  dbCONN.SetupGrids(dbgClients, dbgCollections, dbgrdSQL);
end;

//=====================================================================
// Question 2.1.1          4 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
// Question 2.1.1

  sSQL1 := 'SELECT * FROM tblClients ' +
           'WHERE City = "Bloemfontein"' +
           'ORDER BY ClientSurname';

// Provided code - do not change
  dbCONN.runSQL(sSQL1);
end;

//=====================================================================
//Question 2.1.2          3 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
begin
// Question 2.1.2

  sSQL2 := 'SELECT CollectionID, CollectionDate, NumberOfCans ' +
           'FROM tblCanCollection ' +
           'WHERE Month(CollectionDate) = 1';

// Provided code - do not change
  dbCONN.runSQL(sSQL2);
end;
```

```
//=================================================================
// Question 2.1.3          4 marks
//=================================================================

procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3, sLetter : String;
begin
// Question 2.1.3
// Provided code
  sLetter := InputBox('','Enter first letter of ClientID','');

    sSQL3 := 'SELECT * FROM tblClients ' + ✓
             'WHERE ClientID LIKE ✓ "'+ sLetter ✓ +'%"'; ✓

    // Alternative:
      // WHERE left(clientID, 1) = quotedStr(sLetter)

// Provided code - do not change
  dbCONN.runSQL(sSQL3);
end;

//=================================================================
// Question 2.1.4          8 marks
//=================================================================

procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
begin
// Question 2.1.4

  sSql4 := 'SELECT ClientName, ' +
           'FORMAT(SUM(NumberOfCans / 76 * 8),"Currency") ' +
           'AS [Total Amount] ' +
           'FROM tblCanCollection A, tblClients B ' +
           'WHERE A.ClientID = B.ClientID ' +
           'GROUP BY ClientName';

// Provided code - do not change
  dbCONN.runSQL(sSQL4);
end;

//=================================================================
// Question 2.1.5          4 marks
//=================================================================

procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSql5 : String;
  bChange : boolean;
begin
// Question 2.1.5
```

```
  sSql5 := 'UPDATE tblCanCollection ' +
           'SET NumberOfCans = 250, Paid = False ' +
           'WHERE CollectionID = "C003"';

// Provided code - do not change
dbCONN.ExecuteSQL(sSQL5, bChange);

if bChange then
 begin
   MessageDlg('Database has been updated.', mtInformation, [mbOK], 0);
 end;
end;

//=====================================================================
// Question 2.2.1          4 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
begin
// Question 2.2.1

tblClients.Insert;
tblClients['ClientID'] := 'CHA01';
tblClients['ClientName'] := 'Charles';
tblClients['ClientSurname'] := 'du Boit';
tblClients['Address'] := '24 Van Wouw Street';
tblClients['City'] := 'Cape Town';
tblClients.Post;
end;

//=====================================================================
// Question 2.2.2          13 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  iCompanyTotal, iClientTotal : Integer;
  sYear : String;
  rPercentage : Real;
begin
// Provided code - do not change
  redQ2_2_2.Clear;
  sYear := rgpQ2_2_2.Items[rgpQ2_2_2.ItemIndex];
// ============================================
// Question 2.2.2

  redQ2_2_2.Lines.Add(tblClients['ClientName']+'
                  '+tblClients['ClientSurname']+#13);
  iCompanyTotal := 0;
  iClientTotal := 0;
  tblCanCollection.First;
  while NOT (tblCanCollection.Eof) do
    begin
      if sYear = IntToStr(YearOf(tblCanCollection['CollectionDate'])) then
        begin
```

```
        iCompanyTotal := iCompanyTotal +
                         tblCanCollection['NumberOfCans'];

      if (tblCanCollection['ClientID'] = tblClients['ClientID']) then
          iClientTotal := iClientTotal +
                          tblCanCollection['NumberOfCans'];
      end;
    tblCanCollection.Next;
  end;
  rPercentage := iClientTotal / iCompanyTotal * 100;
  redQ2_2_2.Lines.Add('Client collection for ' + sYear + ': ' + #9 +
                      IntToStr(iClientTotal));
  redQ2_2_2.Lines.Add('Company collection for ' + sYear + ': ' + #9 +
                      IntToStr(iCompanyTotal));
  redQ2_2_2.Lines.Add('Percentage collected by client: ' + #9 +
                FloatToStrF(rPercentage, ffFixed, 3, 2) + '%');
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  redQ2_2_2.Paragraph.TabCount := 2;
  redQ2_2_2.Paragraph.Tab[0] := 180;
  redQ2_2_2.Paragraph.Tab[1] := 150;
end;

procedure TfrmQuestion2.FormShow(Sender: TObject);
begin
  // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblClients := dbCONN.tblOne;
  tblCanCollection := dbCONN.tblMany;
  dbCONN.setupGrids(dbgClients, dbgCollections, dbgrdSQL);
  pgcDBAdmin.ActivePageIndex := 0;
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 3

Object class:

```
unit SolarPowerPlant_U;

interface

type
  TSolarPowerPlant = class(TObject)
  private
  var
    fPlantCode: String;
    fNumberOfPanels: Integer;
    fPowerPerPanel: Real;
    fSeason: String;
  public
    // Provide code
    function getPlantCode : String;
    function getNumOfPanels : Integer;
    function getSeason : String;
    // Code here

    constructor create(sPlantCode: String; iNumOfPanels: Integer;
                       rPowerPerPanel:Real);
    procedure incNumOfPanels(iNumber : Integer);
    procedure setSeason(sNewSeason : String);
    function calculateCapacity : Real;
    function toString: String;
  end;

implementation

{ TStorage }

uses
  SysUtils, Math;

{ TSolarPowerPlant }

// ========================================================
// Question 3.1.1          5 marks
// ========================================================

constructor TSolarPowerPlant.create(sPlantCode: String;
  iNumOfPanels: Integer; rPowerPerPanel: Real);
begin
  fPlantCode := sPlantCode;
  fNumberOfPanels := iNumOfPanels;
  fPowerPerPanel := rPowerPerPanel;
  fSeason := 'Summer';
end;
```

```
// ========================================================
// Question 3.1.2            4 marks
// ========================================================

procedure TSolarPowerPlant.incNumOfPanels(iNumber: integer);
begin
  fNumberOfPanels := fNumberOfPanels + iNumber;
end;

// ========================================================
// Question 3.1.3            3 marks
// ========================================================

procedure TSolarPowerPlant.setSeason(sNewSeason: String);
begin
  fSeason := sNewSeason;
end;

// ========================================================
// Question 3.1.4            8 marks
// ========================================================

function TSolarPowerPlant.calculateCapacity: Real;
var
  iHours : Integer;
begin

  if fSeason = 'Summer' then
  begin
    iHours := 10;
  end
  else if fSeason = 'Winter' then
  begin
    iHours := 6;
  end
  else
  begin
    iHours := 8;
  end;

  result := (fNumberOfPanels * fPowerPerPanel) * iHours;

end;
```

```
// ========================================================
// Question 3.1.5            3 marks
// ========================================================

function TSolarPowerPlant.toString: String;
var
  sString : String;
begin
  sString := 'Plant code: ' + fPlantCode + #13;
  sString := sString + 'Number of panels: ' + intToStr(fNumberOfPanels) +
#13;
  sString := sString + 'Power per panel: ' + floatToStr(fPowerPerPanel) +
#13;
  sString := sString + 'Season: ' + fSeason + #13;
  result := sString;
end;
// ========================================================
// Provided code
// ========================================================

function TSolarPowerPlant.getPlantCode: String;
begin
  result := fPlantCode;
end;

function TSolarPowerPlant.getNumOfPanels: Integer;
begin
  result := fNumberOfPanels;
end;

function TSolarPowerPlant.getSeason: String;
begin
  result := fSeason;
end;

end.
```

Main form unit:

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    gbxQ3_2_1: TGroupBox;
    gbxQ3_2_3: TGroupBox;
    btnQ3_2_1: TButton;
    btnReset: TButton;
    gbxQ3_2_2: TGroupBox;
    btnQ3_2_2: TButton;
    edtQ3_2_1_Power: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    sedQ3_2_2: TSpinEdit;
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ3_2_3: TButton;
    Image1: TImage;
    Label6: TLabel;
    edtQ3_2_1_Code: TEdit;
    sedQ3_2_1: TSpinEdit;
    Label4: TLabel;
    cmbQ3_2_3: TComboBox;
    Label5: TLabel;
    gbxQ3_2_4: TGroupBox;
    btnQ3_2_4: TButton;
    redQ3: TRichEdit;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnQ3_2_4Click(Sender: TObject);
  private
  public
  end;
var
  frmQuestion3: TfrmQuestion3;

implementation
{$R *.dfm}

uses
  SolarPowerPlant_U;

var
  objPlant: TSolarPowerPlant;
```

```
// ========================================================
// Question 3.2.1            6 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.1

  objPlant:= tSolarPowerPlant.create(edtQ3_2_1_Code.Text,
     sedQ3_2_1.Value, strToFloat(edtQ3_2_1_Power.Text));
  redQ3.Lines.Add(objPlant.toString);
end;

// ========================================================
// Question 3.2.2            4 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  rUpdatedPower: Real;
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.2

  objPlant.incNumOfPanels(sedQ3_2_2.Value);
  redQ3.Lines.Add('Plant code: ' + objPlant.getPlantCode);
  redQ3.Lines.Add('Number of panels: ' +
                IntToStr(objPlant.getNumOfPanels));

end;
// ========================================================
// Question 3.2.3            3 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.3

  objPlant.setSeason(cmbQ3_2_3.Text);
  redQ3.Lines.Add(objPlant.toString)

end;
```

```
// ========================================================
// Question 3.2.4            4 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_4Click(Sender: TObject);
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.4

  redQ3.Lines.Add('The maximum generation capacity per day in ' +
                  objPlant.getSeason + ': ');
  redQ3.Lines.Add(floatToStr(objPlant.calculateCapacity) + ' kW');

end;

// ========================================================
// Provided code
// ========================================================

procedure TfrmQuestion3.btnResetClick(Sender: TObject);
begin
  objPlant.Free;
  edtQ3_2_1_Power.Clear;
  edtQ3_2_1_Code.Clear;
  sedQ3_2_1.Value := 15;
  sedQ3_2_2.Value := 50;
  redQ3.Clear;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_u;

interface

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

type
  TfrmQuestion4 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ4_3: TButton;
    redQ4: TRichEdit;
    btnQ4_1: TButton;
    GroupBox1: TGroupBox;
    rgpQ4: TRadioGroup;
    btnQ4_2: TButton;
    Image1: TImage;
    lstQ4: TListBox;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    procedure btnQ4_3Click(Sender: TObject);
    procedure btnQ4_1Click(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }

  public
    { Public declarations }
  end;

var
  frmQuestion4: TfrmQuestion4;

  arrNames: array [1 .. 10] of String = (
    'Ruth',
    'Nicole',
    'Loyiso',
    'Chris',
    'William',
    'Thabo',
    'Vusi',
    'Peter',
    'Jenny',
    'Tommy'
  );
```

```
  arrVending: array [1 .. 10, 1 .. 15] of String =
 (('C', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'B', 'B', 'C', 'C', 'C', 'B', 'B', 'B', 'C', 'C', 'C', 'C', 'C',
''),
    ('', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('C', 'C', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'B', 'C', 'C', 'B', 'B', 'C', 'C', 'C', 'C', 'B', 'C', 'C', 'B',
''),
    ('C', 'C', 'B', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('C', 'B', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'B', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('C', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'C', '', '', '', '', '', '', '', '', '', '', '', '', ''));
implementation

{$R *.dfm}

// ============================================================
// Question 4.1              5 marks
// ============================================================

procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  I, J: Integer;
  sLine: String;
begin
  // Provided code
  redQ4.Clear;
  redQ4.Lines.Add('----------------------------');
  redQ4.Lines.Add('Names'+#9+'Items recycled');
  redQ4.Lines.Add('----------------------------');

  // Question 4.1

  for I := 1 to length(arrNames) do
  begin
    sLine := arrNames[I] + #9;
    for J := 1 to length(arrVending[J]) do
    begin
      sLine := sLine + arrVending[I, J];
    end;
    redQ4.Lines.Add(sLine);
  end;

end;

// ============================================================
// Question 4.2              14 marks
// ============================================================

procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  I, J: Integer;
  rAmount, rMax: Real;
  iMax: Integer;
  arrTotal: Array [1 .. 10] of Real;
  sMaxs: String;
```

```
begin
  // Provided code
  redQ4.Clear;
  redQ4.Lines.Add('----------------------------');
  redQ4.Lines.Add('Names'+#9+'Total amount paid');
  redQ4.Lines.Add('----------------------------');

   // Question 4.2
   // Alternative
  {rMax := -1;
  for I := 1 to length(arrNames) do
  begin
    rAmount := 0;
    for J := 1 to length(arrVending[I]) do
      if arrVending[I, J] = 'B' then
       rAmount := rAmount + 2.15
       else   if arrVending[I, J] = 'C' then
       rAmount := rAmount + 0.75;

    if rAmount > rMax then
      rMax := rAmount;

   arrTotal[I] := rAmount;
   redQ4.Lines.Add(arrNames[I] + #9 + format('%8.2m', [rAmount]));
  end;//I

  //Provided code
  redQ4.Lines.Add('----------------------------');
  redQ4.Lines.Add('Highest payout(s):');
  redQ4.Lines.Add('----------------------------');

  //Code here

  for I := 1 to length(arrTotal) do
  begin
  redQ4.SelAttributes.Style:= [fsBold];
  redQ4.SelAttributes.Color:= clRed;
    if arrTotal[I] = rMax then
    begin
      redQ4.Lines.Add(arrNames[I] + #9 + format('%8.2m', [rMax]));
    end;

  end;}

rMax := -1;
for I := 1 to length(arrNames) do
  begin
    rAmount := 0;
    for J := 1 to length(arrVending[I]) do
      if arrVending[I, J] = 'B' then
         rAmount := rAmount + 2.15
       else if arrVending[I, J] = 'C' then
                    rAmount := rAmount + 0.75;

    redQ4.Lines.Add(arrNames[I] + #9 + format('%8.2m', [rAmount]));
```

```
    if rAmount >= rMax then
      begin
        if rAmount > rMax then
          begin
            rMax := rAmount;
            sMaxs := '';
          end;
        sMaxs := sMaxs + arrNames[I] + #9 + format('%8.2m',
                                                   [rAmount]) + #13;
      end;
  end;//I

  //Provided code
  redQ4.Lines.Add('---------------------------');
  redQ4.Lines.Add('Highest payout(s):');
  redQ4.Lines.Add('---------------------------');
  redQ4.Lines.Add(sMaxs);
end;

// Provided code
procedure TfrmQuestion4.FormShow(Sender: TObject);
begin
  redQ4.Paragraph.TabCount := 1;
  redQ4.Paragraph.Tab[0] := 70;
end;

// ============================================================
// Question 4.3                11 marks
// ============================================================

procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  iRow, iCol: Integer;
  sItem: String;
  bAdded: Boolean;
begin
  // Question 4.3
  // Alternative
 {bAdded := False;
  if lstQ4.ItemIndex = -1 then
  begin
    showMessage('Please select a name');
    exit;
  end
  else
  begin
      case rgpQ4.ItemIndex of
    -1:
      showMessage('Please select an item');
    else
     sItem := rgpQ4.Items[rgpQ4.ItemIndex][1];
   end;
```

```
    iRow := lstQ4.ItemIndex+1;
    iCol:=0;
    while (bAdded=false) AND ( iCol<15) do
    begin
     Inc(iCol);
     if arrVending[iRow,iCol] ='' then
     begin
       arrVending[iRow,iCol] := sItem;
       bAdded := True;
     end;
    end;
    if bAdded = false then
    begin
      showMessage('Machine is full.');
    end;

  end; }

  bAdded := False;
  if (lstQ4.ItemIndex > -1) AND (rgpQ4.ItemIndex > -1) then
  begin
    sItem := rgpQ4.Items[rgpQ4.ItemIndex][1];
    iRow := lstQ4.ItemIndex+1;
    iCol:=0;
    while (bAdded=false) AND ( iCol<15) do
     begin
       Inc(iCol);
       if arrVending[iRow,iCol] = '' then
          begin
            arrVending[iRow,iCol] := sItem;
            bAdded := True;
          end;
     btnQ4_1.Click;
     end;

     if bAdded = false then
     begin
       showMessage('Machine is full.');
     end;

  end
  else
    ShowMessage('Please select both a name and an item.');

  //Provided code
  rgpQ4.ItemIndex := -1;
  lstQ4.ItemIndex := -1;

end;

end.
```