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:

![Image of panel with properties changed]

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:
  
  o Display a message on the label **lblQ1_2** using this format:

    "Company closed, select another month."

  o 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:

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:
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:

```
Learner marks
Enter mark for subject: 1
75
OK Cancel
```

```
Learner marks
Enter mark for subject: 2
68
OK Cancel
```

```
Learner marks
Enter mark for subject: 3
-1
OK Cancel
```

Examples of output:

Marks for two subjects and the average:

```
Subject 1 : 75
Subject 2 : 68
Average mark : 71.50
```
Marks for eight subjects and the average:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
</tr>
<tr>
<td>5</td>
<td>49</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>72</td>
</tr>
<tr>
<td>8</td>
<td>73</td>
</tr>
</tbody>
</table>

Average mark: 69.75

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:

<table>
<thead>
<tr>
<th>Question 1.5</th>
<th>Question 1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dusty</td>
<td>Virtual</td>
</tr>
<tr>
<td>study</td>
<td>trivial</td>
</tr>
</tbody>
</table>

The words form an anagram.

The words do not form an anagram.

- 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblClients</td>
<td>TADOTable</td>
<td>Refers to the table tblClients</td>
</tr>
<tr>
<td>tblCanCollection</td>
<td>TADOTable</td>
<td>Refers to the table tblCanCollection</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>ClientID</th>
<th>ClientName</th>
<th>ClientSurname</th>
<th>Address</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAC05</td>
<td>Jacob</td>
<td>Human</td>
<td>8 Human Street</td>
<td>Bloemfontein</td>
</tr>
<tr>
<td>PIE12</td>
<td>Piet</td>
<td>Mogorosi</td>
<td>5 Stormer Road</td>
<td>Bloemfontein</td>
</tr>
<tr>
<td>RHO09</td>
<td>Rhoda</td>
<td>Somers</td>
<td>14 Marilyn Way</td>
<td>Bloemfontein</td>
</tr>
<tr>
<td>GER01</td>
<td>Gert</td>
<td>Vermeulen</td>
<td>55 Dawn Street</td>
<td>Bloemfontein</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>CollectionID</th>
<th>CollectionDate</th>
<th>NumberOfCans</th>
</tr>
</thead>
<tbody>
<tr>
<td>C001</td>
<td>2020/01/19</td>
<td>412</td>
</tr>
<tr>
<td>C031</td>
<td>2021/01/05</td>
<td>1000</td>
</tr>
<tr>
<td>C032</td>
<td>2021/01/10</td>
<td>450</td>
</tr>
<tr>
<td>C033</td>
<td>2021/01/16</td>
<td>2150</td>
</tr>
<tr>
<td>C034</td>
<td>2021/01/28</td>
<td>800</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>ClientID</th>
<th>ClientName</th>
<th>ClientSurname</th>
<th>Address</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOH03</td>
<td>Johan</td>
<td>Weston</td>
<td>43 Michelin Street</td>
<td>Vanderbijpark</td>
</tr>
<tr>
<td>JAC05</td>
<td>Jacob</td>
<td>Human</td>
<td>8 Human Street</td>
<td>Bloemfontein</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>ClientName</th>
<th>Total Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busi</td>
<td>R905.26</td>
</tr>
<tr>
<td>Chris</td>
<td>R1 173.47</td>
</tr>
<tr>
<td>Damian</td>
<td>R465.58</td>
</tr>
<tr>
<td>Gert</td>
<td>R661.68</td>
</tr>
<tr>
<td>Henry</td>
<td>R400.11</td>
</tr>
</tbody>
</table>

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:

- **Number Of Cans:** 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.

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:
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Client collected 2022:</th>
<th>Company collected 2022:</th>
<th>Percentage collected by client:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Busi Nkosi</td>
<td>2000</td>
<td>21650</td>
<td>9.23%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Client collected 2021:</th>
<th>Company collected 2021:</th>
<th>Percentage collected by client:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chris Ferreira</td>
<td>5288</td>
<td>32867</td>
<td>16.10%</td>
</tr>
</tbody>
</table>

- 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:

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

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:

<table>
<thead>
<tr>
<th>SEASON</th>
<th>HOURS OF SUNLIGHT PER DAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summer</td>
<td>10</td>
</tr>
<tr>
<td>Autumn</td>
<td>8</td>
</tr>
<tr>
<td>Winter</td>
<td>6</td>
</tr>
<tr>
<td>Spring</td>
<td>8</td>
</tr>
</tbody>
</table>
The formula to calculate the generation capacity (GC) of the solar power plant is as follows:

\[ GC = Number\ of\ Panels \times \ Power\ Per\ Panel \times \ Hours\ Per\ Day \] (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>
```

3.2 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:
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:

![Question 3.2.2](image)

Plant code: CSE01
Number of panels: 18

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:

![Example Input Output](image)

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:

![Example Output](image)

- 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):

![Graphical User Interface](image-url)
The following arrays have been provided in the program:

- An array `arrNames` which contains the names of ten learners in a class:
  ```markdown
  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:
  ```markdown
  arrVending: array [1 .. 10, 1 .. 15] of String = 
  (['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:

<table>
<thead>
<tr>
<th>Names</th>
<th>Items recycled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruth</td>
<td>C</td>
</tr>
<tr>
<td>Nicole</td>
<td>BBBCCBBBCCCCC</td>
</tr>
<tr>
<td>Loyiso</td>
<td></td>
</tr>
<tr>
<td>Chris</td>
<td>CC</td>
</tr>
<tr>
<td>William</td>
<td>BBBBCCCCBCCCCB</td>
</tr>
<tr>
<td>Thabo</td>
<td>CCB</td>
</tr>
<tr>
<td>Vusi</td>
<td>CB</td>
</tr>
<tr>
<td>Peter</td>
<td>BB</td>
</tr>
<tr>
<td>Jenny</td>
<td>C</td>
</tr>
<tr>
<td>Tommy</td>
<td>BC</td>
</tr>
</tbody>
</table>

4.2 Button [4.2 - Calculate]

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

<table>
<thead>
<tr>
<th>ITEM RECYCLED</th>
<th>AMOUNT PAID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottle (B)</td>
<td>R2.15</td>
</tr>
<tr>
<td>Can (C)</td>
<td>R0.75</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Names</th>
<th>Total amount paid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruth</td>
<td>R0.75</td>
</tr>
<tr>
<td>Nicole</td>
<td>R18.90</td>
</tr>
<tr>
<td>Loyiso</td>
<td>R0.00</td>
</tr>
<tr>
<td>Chris</td>
<td>R1.50</td>
</tr>
<tr>
<td>William</td>
<td>R18.90</td>
</tr>
<tr>
<td>Thabo</td>
<td>R3.65</td>
</tr>
<tr>
<td>Vusi</td>
<td>R2.90</td>
</tr>
<tr>
<td>Peter</td>
<td>R4.30</td>
</tr>
<tr>
<td>Jenny</td>
<td>R0.75</td>
</tr>
<tr>
<td>Tommy</td>
<td>R2.90</td>
</tr>
</tbody>
</table>

Highest payout(s):

<table>
<thead>
<tr>
<th>Names</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nicole</td>
<td>R18.90</td>
</tr>
<tr>
<td>William</td>
<td>R18.90</td>
</tr>
</tbody>
</table>
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:

![Image](Please select both a name and an item.)

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

![Image](Question 4.3 - Add item)

<table>
<thead>
<tr>
<th>Question 4.3</th>
<th>Recycle item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruth</td>
<td></td>
</tr>
<tr>
<td>Nicole</td>
<td></td>
</tr>
<tr>
<td>Loyiso</td>
<td></td>
</tr>
<tr>
<td>Chris</td>
<td></td>
</tr>
<tr>
<td>William</td>
<td></td>
</tr>
<tr>
<td>Thabo</td>
<td></td>
</tr>
<tr>
<td>Vusi</td>
<td></td>
</tr>
<tr>
<td>Peter</td>
<td></td>
</tr>
<tr>
<td>Jenny</td>
<td></td>
</tr>
<tr>
<td>Tommy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Names</th>
<th>Items recycled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruth</td>
<td>C</td>
</tr>
<tr>
<td>Nicole</td>
<td>BBBBBBBBCBBBBBB</td>
</tr>
<tr>
<td>Loyiso</td>
<td></td>
</tr>
<tr>
<td>Chris</td>
<td>CC</td>
</tr>
<tr>
<td>William</td>
<td>BBBBBBBBBBBBBB</td>
</tr>
<tr>
<td>Thabo</td>
<td>CBB</td>
</tr>
<tr>
<td>Vusi</td>
<td>CB</td>
</tr>
<tr>
<td>Peter</td>
<td>BB</td>
</tr>
<tr>
<td>Jenny</td>
<td>C</td>
</tr>
<tr>
<td>Tommy</td>
<td>BC</td>
</tr>
</tbody>
</table>
Example of output when an attempt was made to add another item for the name 'Nicole':

```
Machine is full.
```

- 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.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientID</td>
<td>Text (5)</td>
<td>Unique ID for the client</td>
</tr>
<tr>
<td>ClientName</td>
<td>Text (15)</td>
<td>The name of the client</td>
</tr>
<tr>
<td>ClientSurname</td>
<td>Text (15)</td>
<td>The surname of the client</td>
</tr>
<tr>
<td>Address</td>
<td>Text (20)</td>
<td>The address of the client used for the pickup of cans</td>
</tr>
<tr>
<td>City</td>
<td>Text (15)</td>
<td>The city where the client resides</td>
</tr>
</tbody>
</table>

Example of the records in the tblClients table:

<table>
<thead>
<tr>
<th>ClientID</th>
<th>ClientName</th>
<th>ClientSurname</th>
<th>Address</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABI10</td>
<td>Prashant</td>
<td>Govender</td>
<td>72 Mountain Road</td>
<td>Kimberley</td>
</tr>
<tr>
<td>BUS06</td>
<td>Busi</td>
<td>Nkosi</td>
<td>65 Donald Road</td>
<td>Welkom</td>
</tr>
<tr>
<td>CHR08</td>
<td>Chris</td>
<td>Ferreira</td>
<td>188 Richmond Street</td>
<td>Potchefstroom</td>
</tr>
<tr>
<td>DAM07</td>
<td>Damian</td>
<td>Coetzee</td>
<td>12 Cape Avenue</td>
<td>Durban</td>
</tr>
<tr>
<td>GER01</td>
<td>Gert</td>
<td>Vermeulen</td>
<td>55 Dawn Street</td>
<td>Bloemfontein</td>
</tr>
<tr>
<td>HEN11</td>
<td>Henry</td>
<td>Marques</td>
<td>1 Kingsley Drive</td>
<td>Cape Town</td>
</tr>
<tr>
<td>JAC05</td>
<td>Jacob</td>
<td>Human</td>
<td>8 Human Street</td>
<td>Bloemfontein</td>
</tr>
<tr>
<td>JOH03</td>
<td>Johan</td>
<td>Weston</td>
<td>43 Michelin Street</td>
<td>Vanderbijlpark</td>
</tr>
<tr>
<td>PH04</td>
<td>Phillip</td>
<td>Brown</td>
<td>11 Park Road</td>
<td>Sasolburg</td>
</tr>
<tr>
<td>PIE12</td>
<td>Piet</td>
<td>Mogorosi</td>
<td>5 Stormer Road</td>
<td>Bloemfontein</td>
</tr>
<tr>
<td>RHU09</td>
<td>Rhoda</td>
<td>Somers</td>
<td>14 Marilyn Way</td>
<td>Bloemfontein</td>
</tr>
<tr>
<td>WIL12</td>
<td>Willem</td>
<td>de Wit</td>
<td>2 Arrow Street</td>
<td>Sasolburg</td>
</tr>
</tbody>
</table>
Table: **tblCanCollection**

This table contains information of all the collections.

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

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

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

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