basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE/ NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2020

MARKS: 150

TIME: 3 hours

This question paper consists of 25 pages and 2 data pages.

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into FOUR sections. Candidates must answer ALL the questions in each of the FOUR sections.

2. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.

3. This question paper is set with programming terms that are specific to the Delphi programming language.

4. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements.

5. Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.

6. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper.

7. Routines, such as search, sort and selection, must be developed from first principles. You may NOT use the built-in features of Delphi for any of these routines.

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

9. You must save your work regularly on the disk/CD/DVD/flash disk you have been given, or on the disk space allocated to you for this examination session.

10. Make sure that your examination number appears as a comment in every program that you code, as well as on every event indicated.

11. If required, print the programming code of all the programs/classes that you completed. You will be given half an hour printing time after the examination session.

12. At the end of this examination session you must hand in a disk/CD/DVD/flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read.

13. The files that you need to complete this question paper have been given to you on the disk/CD/DVD/flash disk or on the disk space allocated to you. The files are provided in the form of password-protected executable files.

NOTE: Candidates must use the file **DataENGJun2020.exe**.

Do the following:

- Double click on the password-protected executable file: **DataENGJun2020.exe**.
- Click on the 'Extract' button.
- Enter the following password: **Plant2BGreen!**

Once extracted, the following list of files will be available in the folder **DataENGJun2020**:

SUPPLIED FILES:

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

Question2:
ConnectDB_U.dcu
ConnectDB_U.pas
EmployeesDB.mdb
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question3:
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas
Transaction_U.pas

Question4
Question4_P.dpr
Question4_P.dproj
Question4_P.res
Question4_U.dfm
Question4_U.pas

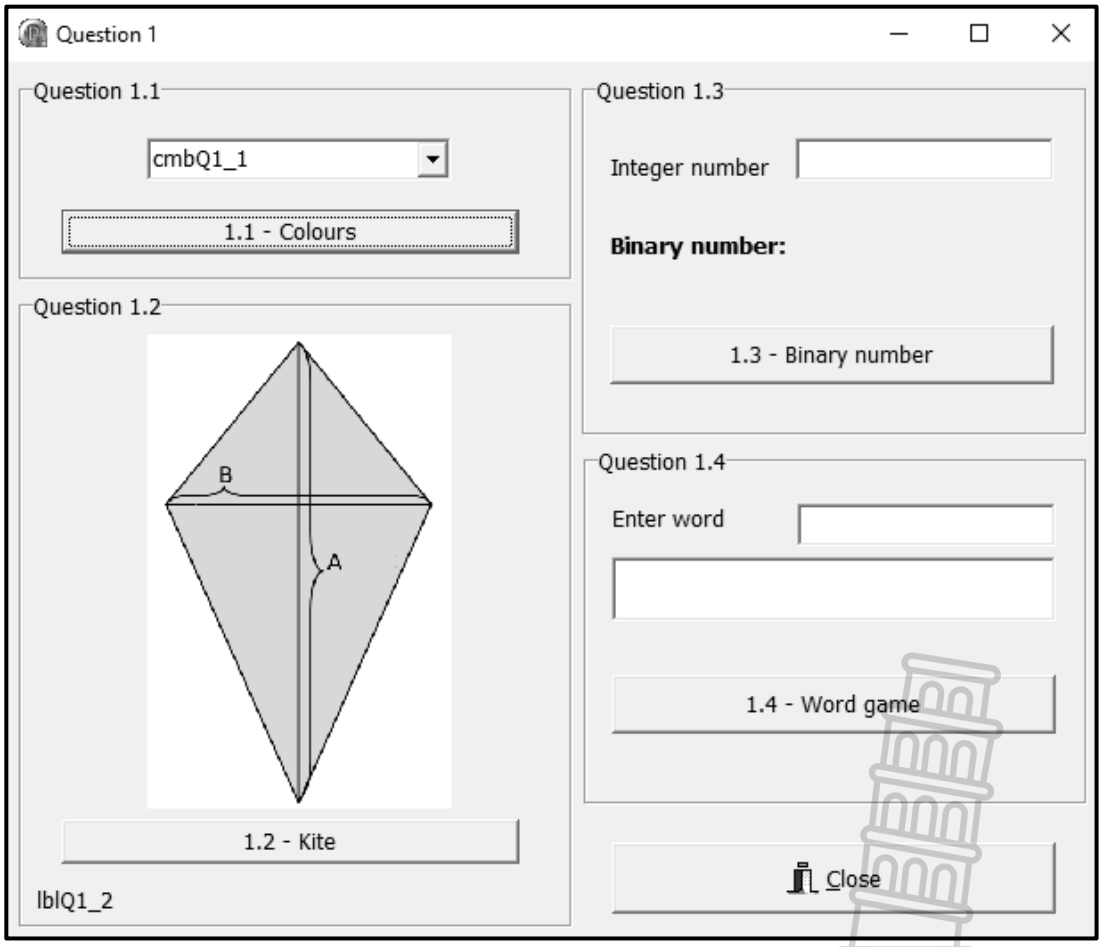
SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete project file called **Question1_P.dpr** 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 user interface displays FOUR different sections named Question 1.1 to Question 1.4. The program has no functionality currently.

Example of the graphical user interface (GUI):



- Complete the code for EACH section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.4 that follow.

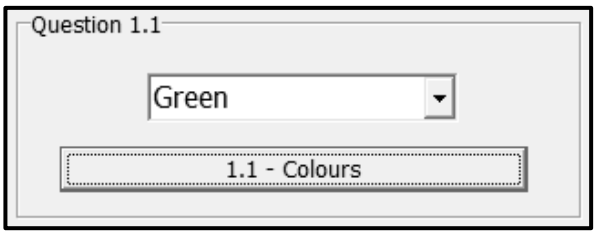
1.1 **Button [1.1 - Colours]**

The combo box **cmbQ1_1** contains the items 'Red' and 'Green' currently.

Write code to change the content of the combo box **cmbQ1_1** as follows when the **btnQ1_1** button is clicked:

- The font size of the text must be 12 pt.
- 'Blue' must appear as a third option on the list of items.
- The item ('Green') must be displayed as the default option.

Example of the appearance of the combo box when the **btnQ1_1** button is clicked:



(4)

1.2 **Button [1.2 - Kite]**

The formula to calculate the area of a kite is as follows:

$$\textbf{area} = \frac{\textbf{length of diagonal one x length of diagonal two}}{\textbf{2}}$$

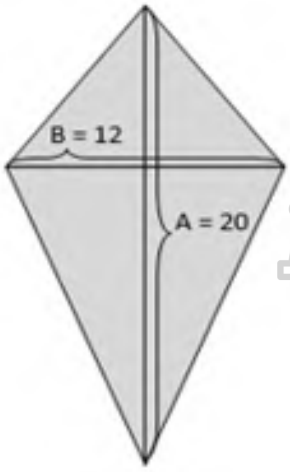
Example of calculation for a kite with a value of 20 for diagonal one (A) and 12 for diagonal two (B):

$$\textbf{area} = \frac{\textbf{length of diagonal one (A) x length of diagonal two (B)}}{\textbf{2}}$$

$$= \frac{20 \times 12}{2}$$

$$= \frac{240}{2}$$

$$= 120$$



Code has been provided to allow the user to enter the values for diagonal one (A) and diagonal two (B) using input dialog boxes. These values are stored in variables **iDiagA** and **iDiagB** respectively.

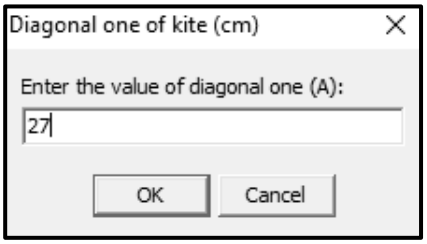
Write code to do the following:

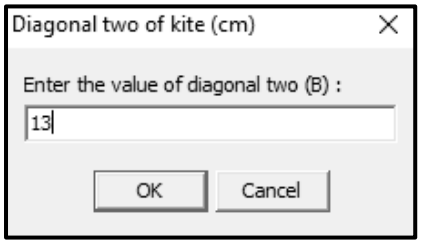
- Declare a suitable numeric variable to store the area of the kite.
- Test whether the value of diagonal one (A) is greater than the value of diagonal two (B) or not.

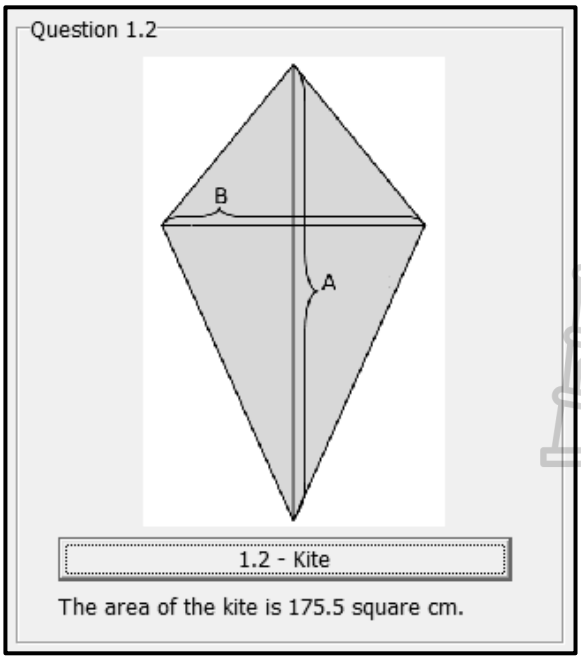
  If the value of diagonal one (A) is greater than the value of diagonal two (B), use the provided formula to calculate the area of the kite. Display the area on label **lblQ1_2** formatted to ONE decimal place.

  If the value of diagonal one (A) is NOT greater than the value of diagonal two (B), display a suitable message that briefly states that the value of diagonal one (A) must be greater than the value of diagonal two (B).

Example of output if the value of diagonal one (A) is 27 cm and the value of diagonal two (B) is 13 cm:







(8)

1.3 **Button [1.3 - Binary number]**

The conversion of an integer number into a binary number is done by repetitive division by 2 until the final answer of the division is equal to zero (0). The integer remainder of each division operation is combined into a string of digits (zeros and ones) starting with the last integer remainder up to the first integer remainder. The string of remainder digits represents the binary value of the original integer value.

The example below illustrates how the integer value of 19 is converted to a binary number:

Integer division by 2	**Remainder**
19 ÷ 2 = 9	1
9 ÷ 2 = 4	1
4 ÷ 2 = 2	0
2 ÷ 2 = 1	0
1 ÷ 2 = 0	1
Binary number: 10011_2	

Concatenate the integer remainder from the last remainder up to the first remainder to construct the binary number that represents the integer value.

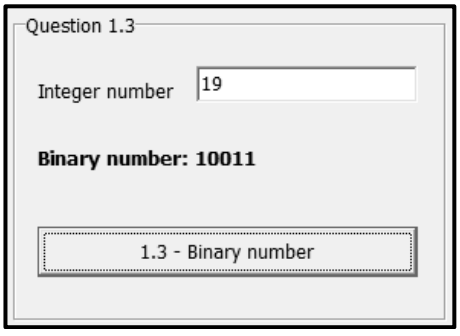
Write code to do the following:

- Retrieve the integer number that was entered by the user in the **edtQ1_3** edit box as a number value using the variable **iNumber**.
- Initialise the string variable **sBinary** to store the binary value that will be constructed.
- Convert the integer number to binary using the following algorithm:

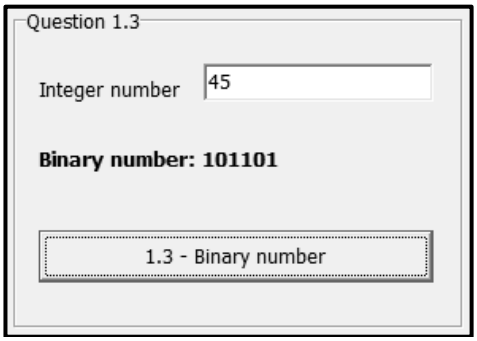
  Repeat until the number value is equal to zero:
  - Determine the remainder of the number value divided by 2 using the variable called **iRemainder**.
  - Join (Append) the remainder to the left of the content of the string variable.
  - Divide the number value by 2 and save the answer in the number value variable (replacing the previous number value).

- Use the label **lblQ1_3** to display the content of the string variable **sBinary**.

Example of output if an integer value of 19 was entered:



Example of output if an integer value of 45 was entered:



(11)

1.4 **Button [1.4 - Word game]**

A word game needs to be developed where each character in a word carries a specific number of points. Points are allocated as follows:

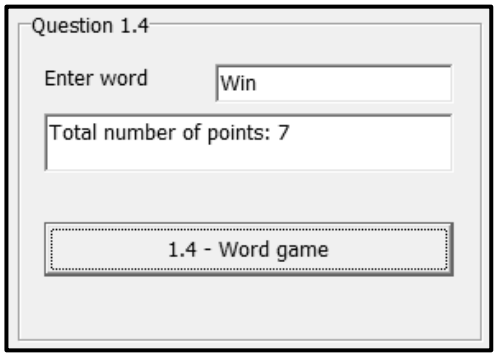
Characters	Value in points
Vowels (A, E, I, O, U)	3 points
Alphabetical characters that are not vowels	2 points
Any other non-alphabetical character, such as an apostrophe or a hyphen	1 point

The player must enter a single word in the edit box **edtQ1_4**. When the **btnQ1_4** button is clicked, the program must calculate and display the total value of the word in terms of points. If more than one word is entered, the player is disqualified.

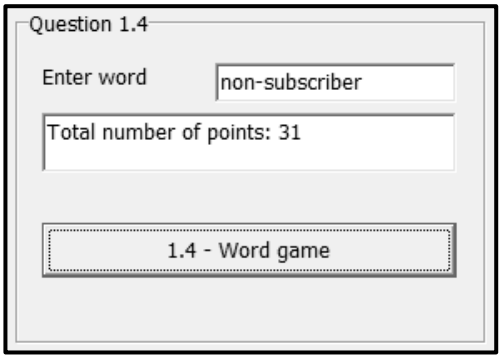
Write code to do the following:

- Retrieve the word that was entered in the edit box **edtQ1_4** and convert the text to upper case.
- Test for the presence of a space in the text which indicates that more than one word was entered:
  - If a space is identified, display a suitable message indicating that the player has been disqualified. Clear the **edtQ1_4** edit box and set the focus on the edit box.
  - If a space is not identified, calculate the total value of the word based on the number of points each character carries. Display the total value of the word in the provided rich edit component.

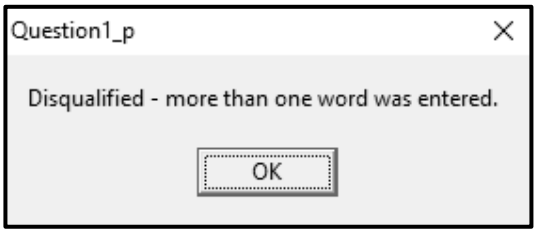
Example of output if the word **Win** was entered:



Example of output if the word **non-subscriber** was entered:



Example of output if the words **Team one** were entered:



(17)

- Ensure that your examination number has been entered 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

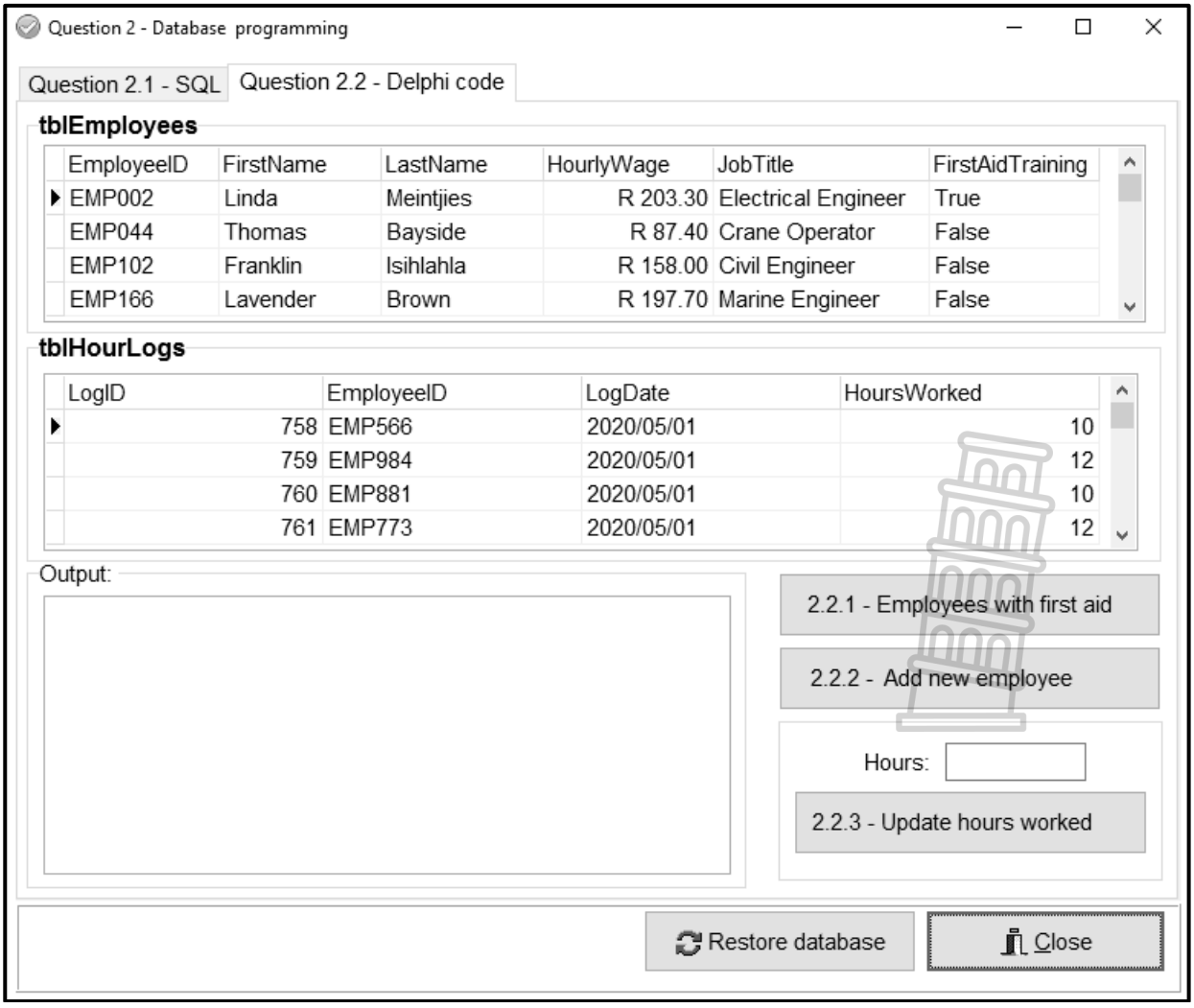
QUESTION 2: DATABASE PROGRAMMING

The database **EmployeesDB** contains the information of the employees of an import and export company. The database contains two tables called **tblEmployees** and **tblHourLogs**.

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

Do the following:

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



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

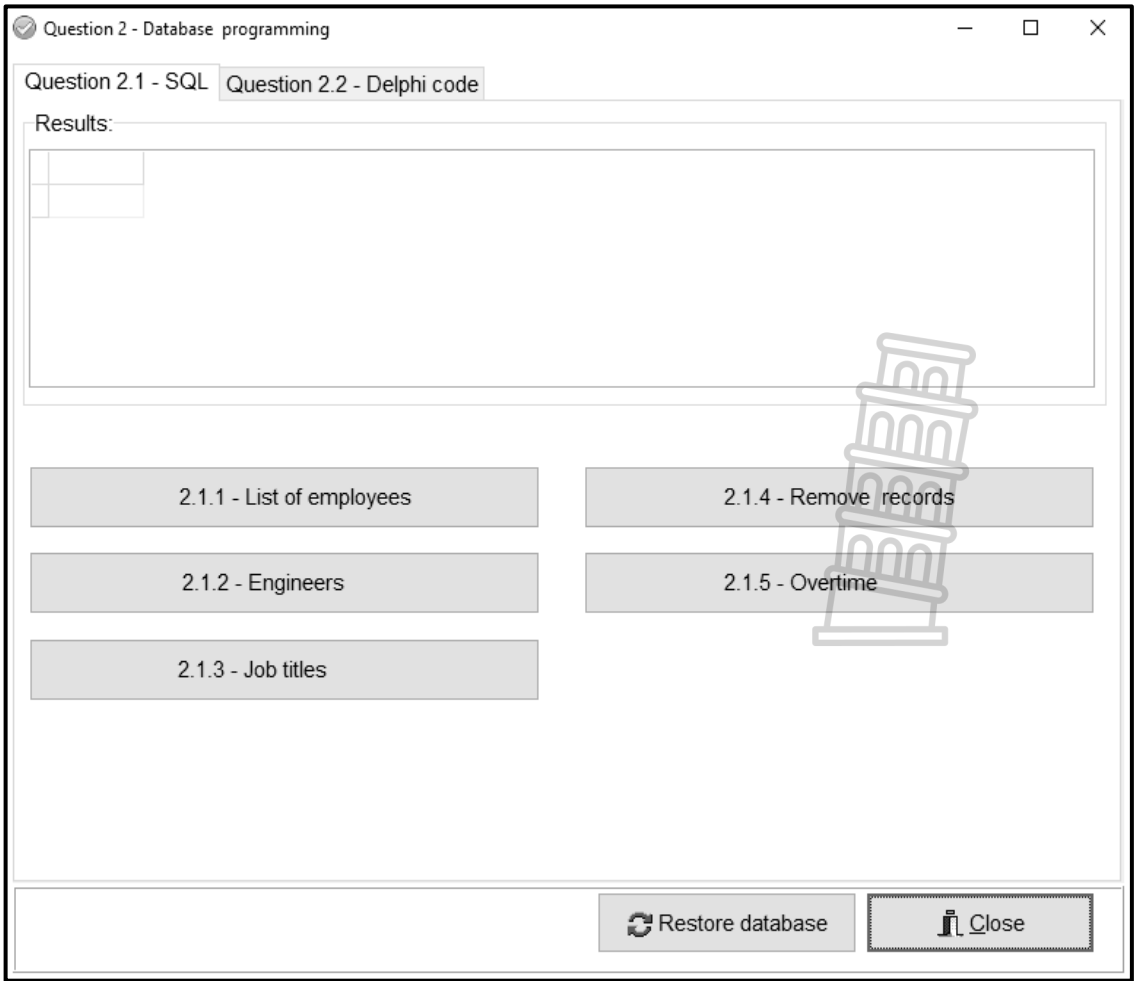
NOTE:

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

Variable	Data type	Description
`tblEmployees`	`TADOTable`	Refers to the table tblEmployees
`tblHourLogs`	`TADOTable`	Refers to the table tblHourLogs

2.1 Tab sheet [Question 2.1 - SQL]

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



NOTE:

- Use ONLY SQL statements to answer QUESTION 2.1.1 to QUESTION 2.1.5.
- Code is provided to execute the SQL statements and display the results of the queries. The SQL statements 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 that follow.

2.1.1 **Button [2.1.1 - List of employees]**

Display ALL the information on employees from the **tblEmployees** table, firstly sorted in ascending order according to the **JobTitle** field and secondly according to the **HourlyWage** field in descending order.

Example of output of the first four records:

EmployeeID	FirstName	LastName	HourlyWage	JobTitle	FirstAidTraining
EMP102	Franklin	Isihlahla	R 158.00	Civil Engineer	False
EMP883	Ricardo	Clementi	R 87.40	Crane Operator	False
EMP044	Thomas	Bayside	R 87.40	Crane Operator	False
EMP909	Jonathan	Black	R 84.10	Crane Operator	False

(4)

2.1.2 **Button [2.1.2 - Engineers]**

Display the **EmployeeID**, **LastName** and **FirstName** fields of employees whose job title contains the word 'Engineer'.

Example of output:

EmployeeID	LastName	FirstName
EMP773	Weasly	Percival
EMP166	Brown	Lavender
EMP102	Isihlahla	Franklin
EMP002	Meintjies	Linda

(4)

2.1.3 **Button [2.1.3 - Job titles]**

Display a list of all the different job titles at the company. Each job title must be displayed ONCE only.

Example of output:

JobTitle
Civil Engineer
Crane Operator
Electrical Engineer
Forklift Driver
Human Resources Manager
Marine Engineer

(3)

2.1.4 **Button [2.1.4 - Remove records]**

A number of records with incorrect data on the hours worked have been captured in the **tblHourLogs** table.

Remove all the records in the **tblHourLogs** table where the **HoursWorked** is recorded as 99.

Code has been provided to display a message that indicates that the content of the database has been changed. (2)

2.1.5 **Button [2.1.5 - Overtime]**

Employees are paid overtime for each hour exceeding 8 hours per day. An employee's overtime payment is double his regular hourly wage.

Calculate and display the total amount each employee has been paid for overtime hours worked, formatted as currency. Display the amount using the field name **OvertimeAmt**. Display only the **LastName** and the calculated **OvertimeAmt** as output.

Example of output:

LastName	OvertimeAmt
Bagge	R 20 854.40
Bayside	R 8 914.80
Black	R 8 410.00
Brown	R 18 583.80
Clementi	R 8 215.60

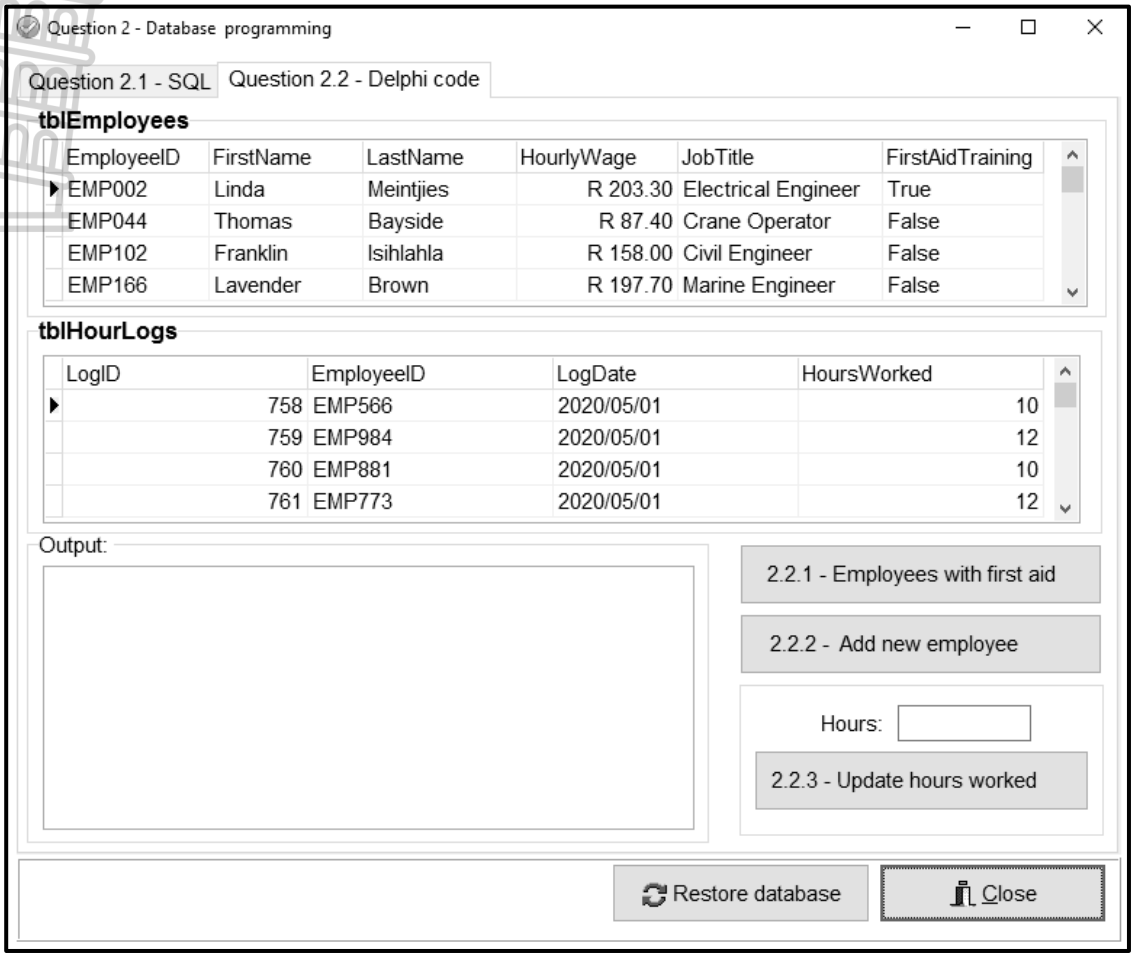
(9)

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 the graphical user interface (GUI) for QUESTION 2.2:



2.2.1 **Button [2.2.1 - Employees with first aid]**

Write code to do the following:

- Display the employee ID, last name and job title of ALL employees who have completed first aid training in the rich edit component **redQ2**.
- Count the number of employees who completed first aid training and display the result below the employee details, as shown in the example below.

NOTE: Code to display the column headings has been provided.

Example of output:

EmployeeID	LastName	JobTitle
EMP002	Meintjies	Electrical Engineer
EMP566	Bagge	Risk Manager
EMP881	Patel	Site Manager

Total number of employees with first aid training: 3

(7)

2.2.2 Button [2.2.2 - Add new employee]

Write code to add a new record to the **tblEmployees** table. The data for the employee is as follows:

Employee ID:	EMP986
First name:	Robert
Last name:	Laubscher
Hourly wage:	195.00
Job title:	Marine Engineer
First aid training:	True

Example of the last few records in the **tblEmployees** table which shows that the record has been added to the table successfully:

EmployeeID	FirstName	LastName	HourlyWage	JobTitle	FirstAidTraining
EMP892	Precious	Idwala	R 98.50	Human Resources Manager	False
EMP909	Jonathan	Black	R 84.10	Crane Operator	False
EMP984	Colin	Lenning	R 68.00	Forklift Driver	False
EMP986	Robert	Laubscher	R 195.00	Marine Engineer	True

(6)

2.2.3 Button [2.2.3 - Update hours worked]

The administrator is able to change the logged number of hours for a record in the **tblHourLogs** table. The user must select a record in the DBGrid **dbgHourLogs** and enter the value in the edit box to replace the current number of hours worked.

Write code to update the **HoursWorked** field of the selected record to the value obtained from the edit box.

Example of the content of the record if the LogID 758 was selected and the value of 11 was entered for **HoursWorked**:

LogID	EmployeeID	LogDate	HoursWorked
758	EMP566	2020/05/01	11

Example of the content of the record if LogID 774 was selected and the value of 8 was entered for **HoursWorked**:

LogID	EmployeeID	LogDate	HoursWorked
774	EMP892	2020/05/02	8

(5)

- Ensure that your examination number has been entered 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

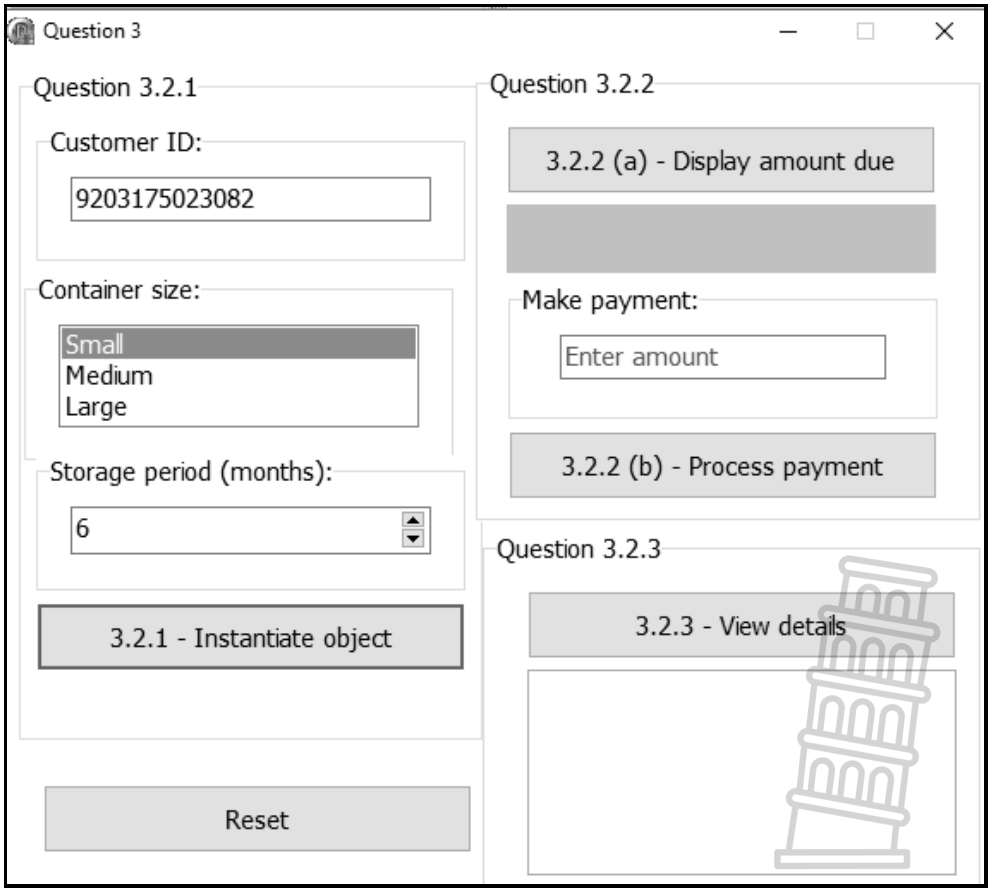
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

An import-export company rent out containers to customers who use it to store goods for a specific period of time. Transactions are created between the company and the customers to keep track of the containers that have been rented out.

Do the following:

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

Example of the graphical user interface (GUI):



- Complete the code as specified in QUESTION 3.1 for the **Transaction_U** object class and QUESTION 3.2 for the **Question3_U** form class.

3.1 The incomplete object class (**TTransaction**) provided contains four attributes and an incomplete **toString** method that defines a **Transaction** object.

The attributes for the **Transaction** object have been declared as follows:

Names of attributes	Description
fCustomerID	The South African identification number of a customer who is requesting to rent a container
fContainerSize	A character that indicates the size of the container: • S = Small • M = Medium • L = Large
fStoragePeriod	Indicates the number of months that the container will be made available to the customer
fAmountPaid	The amount that the customer has already paid to use the container

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

3.1.1 Write code for a **constructor** method that will receive the customer ID, the container size and the storage period in months as parameter values. Assign these values to the respective attributes. Set the **fAmountPaid** attribute to the value of zero. (4)

3.1.2 Write code for an accessor method called **getAmountPaid** for the **fAmountPaid** attribute. (2)

3.1.3 Write code for a method called **updateAmountPaid** that will receive a value as a parameter and add the received value to the **fAmountPaid** attribute. (3)

3.1.4 Write code for a method called **calculateCost** that will use the **Transaction** object's attributes to calculate and return the cost for hiring the container.

The **monthly** cost of the container will depend on the size of the container, as indicated in the table below.

Container size	Amount per month
S	R1000.00
M	R1750.00
L	R2500.00

The **total** cost of hiring the container is calculated as follows:

- The monthly cost of the container is multiplied by the storage period in months.
- A discount of 10% of the total cost is applied for every six months of the storage period, up to a maximum of 50%.

Example 1:

Container size:	M
Storage period (months):	3
Discount (%):	0%
Cost:	R5250.00

Example 2:

Container size:	S
Storage period (months):	6
Discount (%):	10%
Cost:	R5400.00

Example 3:

Container size:	L
Storage period (months):	25
Discount (%):	40%
Cost:	R37500.00

Example 4:

Container size:	S
Storage period (months):	72
Discount (%):	50%
Cost:	R36000.00

(11)

3.1.5 An incomplete **toString** method has been supplied.

Write code to complete the **toString** method to return the attributes of the **Transaction** object in the following format:

```
Customer ID: <fCustomerID>
Container size: <fContainerSize>
Storage period (months): <fStoragePeriod>
Amount paid: <fAmountPaid>
```

Example:

```
Customer ID: 7203075021082
Container size: M
Storage period (months): 7
Amount paid: R0.00
```

(3)

3.2 An incomplete unit **Question3_U** has been provided and contains code for the object class to be accessible. A global **TTransaction** variable called **objTransaction** has been provided.

Do the following to complete the code for QUESTION 3.2.1 to QUESTION 3.2.3 in the main form unit:

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

Write code to do the following:

- Extract the customer's ID from **edtQ3_2_1**, the first letter of the container size selected in **lstQ3_2_1** and the storage period in months from **sedQ3_2_1**.
- Instantiate a **Transaction** object using the values extracted from the user interface.
- Disable the **btnQ3_2_1** button.

NOTE: The provided code for the **[Reset]** button will remove the **Transaction** object from the memory and enable **btnQ3_2_1** so that a new **Transaction** object can be instantiated. (7)

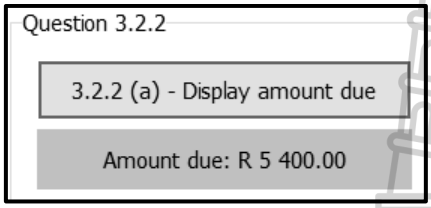
3.2.2 (a) **Button [3.2.2 (a) - Display amount due]**

Write code to do the following:

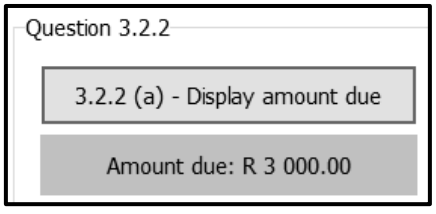
- Use the relevant method to calculate the amount due for hiring a container.
- Display the amount due on the panel **pnlQ3_2_2** in the following format:

```
Amount due: <amount due>
```

Example of output with the default values provided are selected, with no amount paid yet:



Example of output if the customer selected a small container size for a storage period of 3 months, with no amount paid yet:



(3)

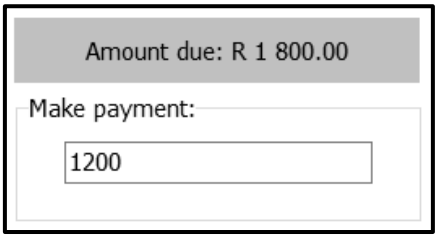
(b) **Button [3.2.2 (b) - Process payment]**

The customer needs to enter an amount to be paid in the provided edit box **edtQ3_2_2**.

Write code to do the following:

- Extract the amount that was entered in box **edtQ3_2_2**.
- Use the appropriate method to update the attribute of the amount paid with the amount extracted from **edtQ3_2_2**.
- Use the appropriate methods to calculate the updated amount due.
- Display the updated amount due on the panel **pnlQ3_2_2**.

Example of output if the customer selected a small container size for a storage period of 3 months, with an amount of R1 200 paid:



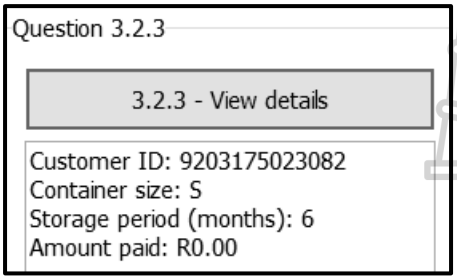
(5)

3.2.3 **Button [3.2.3 - View details]**

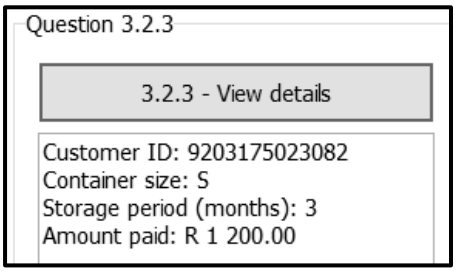
The customer must be able to view the details of the **Transaction** object. Code to clear the rich edit component has been provided.

Write code to display the attributes of the **Transaction** object in the provided rich edit by using the **toString** method.

Example of output with the default value selected and no amount paid yet:



Example of output if the customer selected a small container size for a storage period of 3 months, with an amount of R1 200 paid:



(2)

- Ensure that your examination number has been entered as a comment in the first line of the object class and the form class.
- Save all files.
- Print the code if required.

TOTAL SECTION C: 40

SECTION D

QUESTION 4: PROBLEM-SOLVING PROGRAMMING

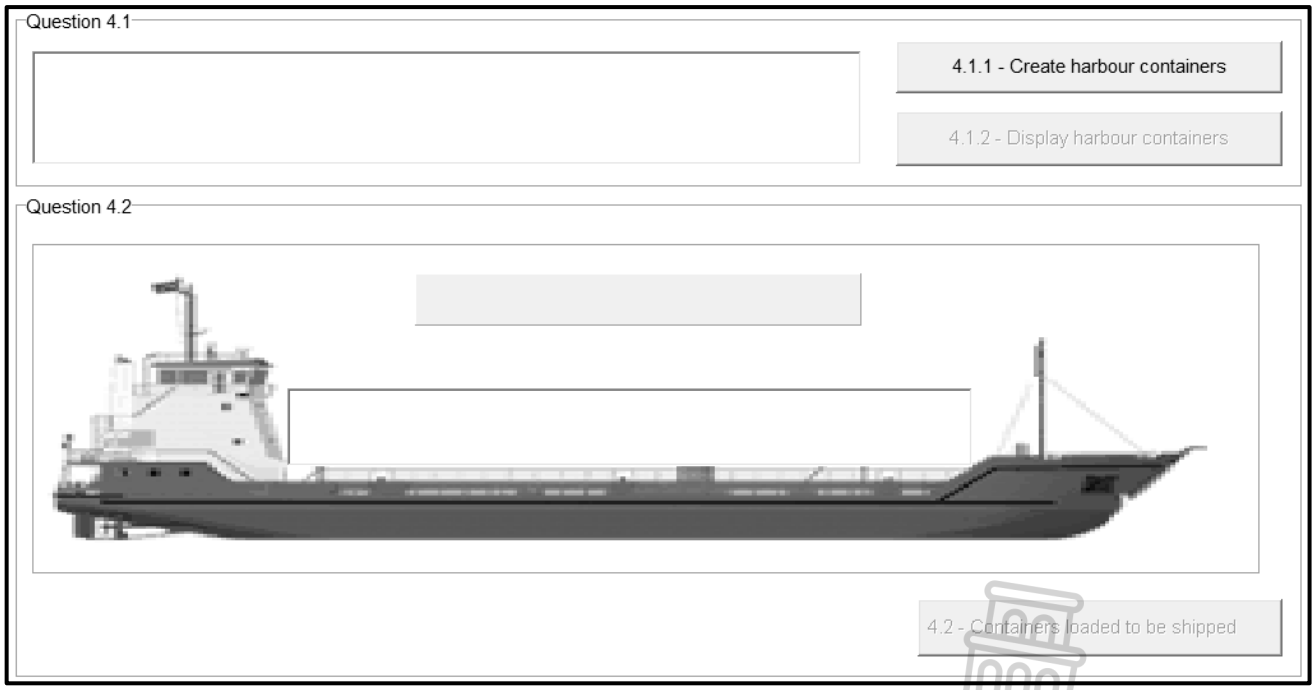
SCENARIO

Tons of cargo are loaded into containers and stored in the harbour for shipment.

Do the following:

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

Example of the graphical user interface (GUI):



- Read the information below and complete the code for the tasks described in QUESTION 4.1 and QUESTION 4.2 that follow.

  The program contains the code shown below for the declaration of an array called **arrContainers**.

```
arrContainers: array [1..50] of real;
```

  arrContainers is declared with a maximum size of 50 elements to store random real numbers that represent the different weights of each container in tons.

NOTE: Code is provided to activate, deactivate and clear certain components for each event.

4.1 Write code as described in QUESTION 4.1.1 and QUESTION 4.1.2 to generate and display the weight of harbour containers.

4.1.1 **Button [4.1.1 - Create harbour containers]**

Write code to generate fifty (50) random real values in the range from 1 to 99 (both included) and rounded to ONE decimal place, and assign the values to the one-dimensional array called **arrContainers**.

NOTE: Use the provided array **arrTempContainers** if your attempt to populate the **arrContainers** array with random values was unsuccessful. (4)

4.1.2 **Button [4.1.2 - Display harbour containers]**

Write code to display the weight of the containers from the **arrContainers** array in the rich edit **redQ4_1** arranged in 5 rows and 10 columns as shown in the example below.

Example of possible output:

31.2	43.4	5.1	41.2	52.6	41.9	97	49.3	15.8	39.5
78.8	96.3	67.8	47.2	31.4	18.4	27	1.4	33.5	16.5
58.3	9.9	62.9	11.8	62.5	59	13.4	49.8	60.4	74.5
13.5	67.7	94	39.4	47.4	13.1	26.2	63	89	22.3
54	16.9	38.6	46.2	22.5	11.4	65.1	48.6	41.4	47.9

NOTE: The values shown in the example will differ from your program's output, as the values are randomly generated every time the **[4.1.1 - Create harbour containers]** button is clicked. (8)

4.2 **Button [4.2 - Containers loaded to be shipped]**

A ship in the harbour must be loaded with containers from the one-dimensional array **arrContainers**. The total weight loaded onto the ship must be as close as possible, but not exceed, the maximum value of 200 tons.

Containers must be loaded row by row from the **arrContainers** array until the maximum weight is reached. If adding the weight of a specific container to the total weight causes the total weight to exceed the maximum weight of 200 tons, that specific container must not be loaded onto the ship. The program must search row by row through the remaining containers to find another container with a lower weight that can still be loaded onto the ship without exceeding the maximum weight.

The number of containers loaded onto the ship will depend on the random data generated in the **arrContainers** array.

Example of output for the random values generated in QUESTION 4.1.1.

31.2	43.4	5.1	41.2	52.6	15.8	1.4

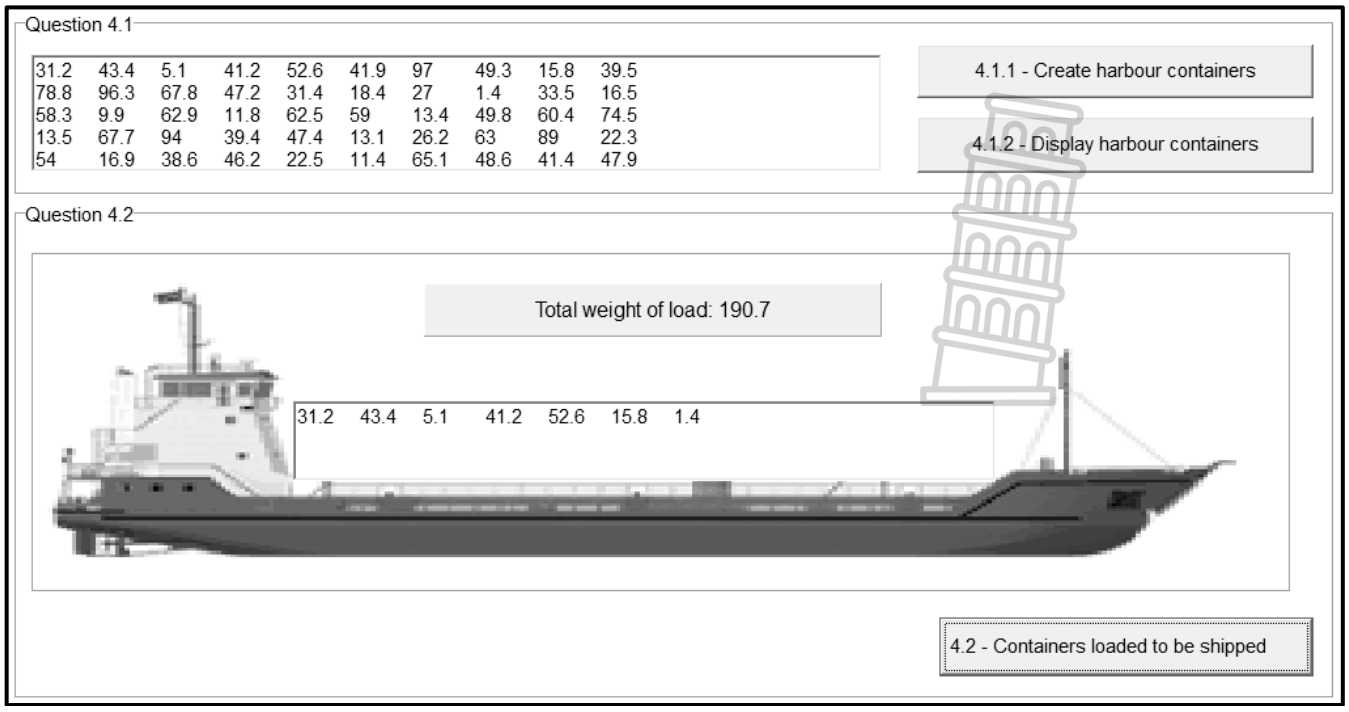
Explanation:

The weight of the first five containers in row 1 adds up to a total value of 173.5 tons. Adding the weight of each of the next three containers (41.9, 97, and 49.3 respectively) will cause the allowed maximum weight to be exceeded, and are therefore not loaded. The next suitable container to be loaded will be the container with the weight of 15.8 tons and thereafter the container with the weight of 1.4 tons. The total weight of the containers to be loaded adds up 190.7 tons – the closest total to the value of 200 tons. No more containers can be loaded.

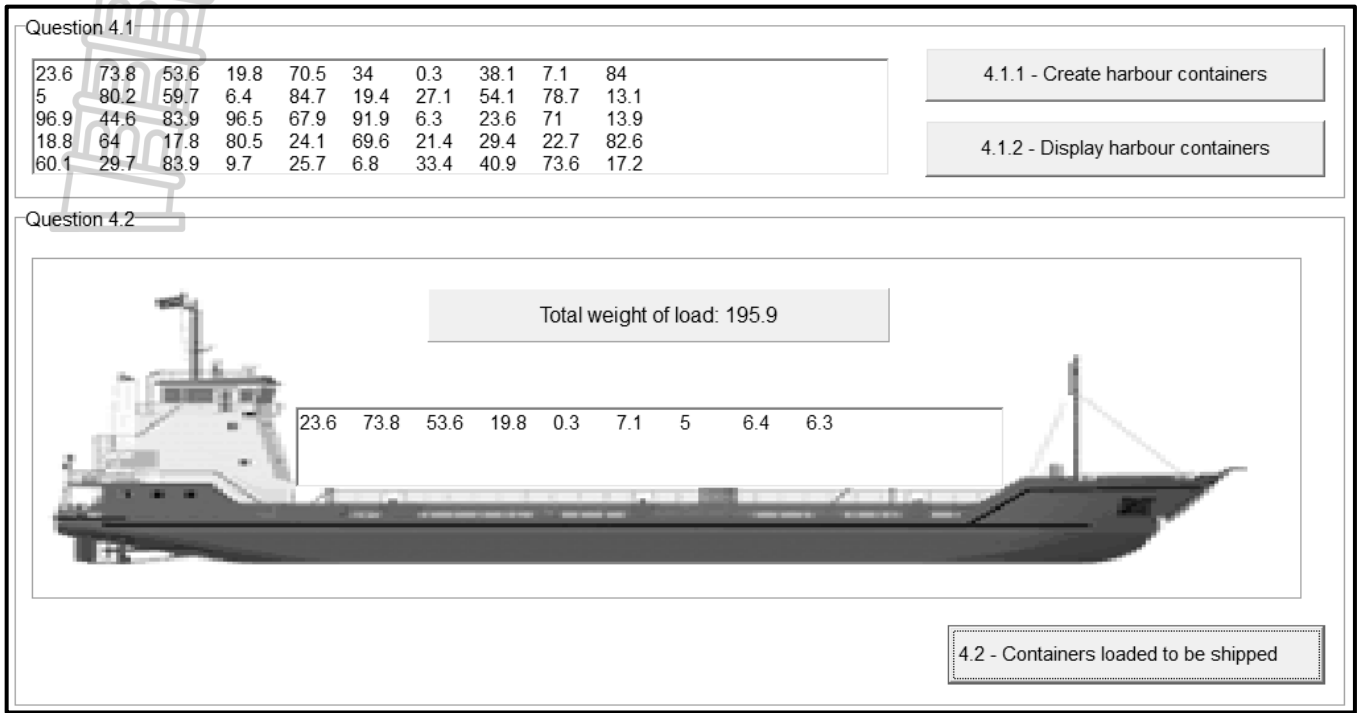
Write code to do the following:

- Display the weight of the containers loaded onto the ship in the rich edit component called **redQ4_2**.
- Write the weight of the containers loaded onto the ship to a new text file called **Tons.txt**.
- Terminate the process of loading the containers onto the ship as soon as the maximum possible weight is loaded onto the ship.
- Display the total weight loaded onto the ship on the panel **pnlQ4**.

Example of possible output:



Output for a different set of data generated:



(18)

- Ensure that your examination number has been entered 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 FOR QUESTION 2:

The design of the database tables is as follows:

Table: **tblEmployees**

This table contains the details of the staff working at the import-export company.

Field name	Data type	Description
EmployeeID	Text (8)	A unique ID that identifies an employee
FirstName	Text (40)	The first name of the employee
LastName	Text (40)	The last name of the employee
HourlyWage	Currency	The amount the employee is paid per hour
JobTitle	Text (80)	The title of the employee's position at the company
FirstAidTraining	Boolean	A field that indicates whether an employee has completed a first aid training course or not

Example of the records of the **tblEmployees** table:

EmployeeID	FirstName	LastName	HourlyWage	JobTitle	FirstAidTraining
EMP002	Linda	Meintjies	R203.30	Electrical Engineer	☑
EMP044	Thomas	Bayside	R87.40	Crane Operator	☐
EMP102	Franklin	Isihlahla	R158.00	Civil Engineer	☐
EMP166	Lavender	Brown	R197.70	Marine Engineer	☐
EMP566	Eustace	Bagge	R212.80	Risk Manager	☑
EMP773	Percival	Weasly	R175.80	Electrical Engineer	☐
EMP881	Jayshree	Patel	R199.00	Site Manager	☑
EMP883	Ricardo	Clementi	R87.40	Crane Operator	☐
EMP892	Precious	Idwala	R98.50	Human Resources Manager	☐

Table: **tblHourLogs**

This table contains the information of the hours worked which is logged by the staff daily.

Field name	Data type	Description
LogID	AutoNumber	A unique number assigned to an employee's daily working hours
EmployeeID	Text (8)	A unique ID that identifies the employee to whom the hours logged belongs
LogDate	Date/Time	The date on which the hours were logged
HoursWorked	Number	The number of work hours logged for a specific employee for the day

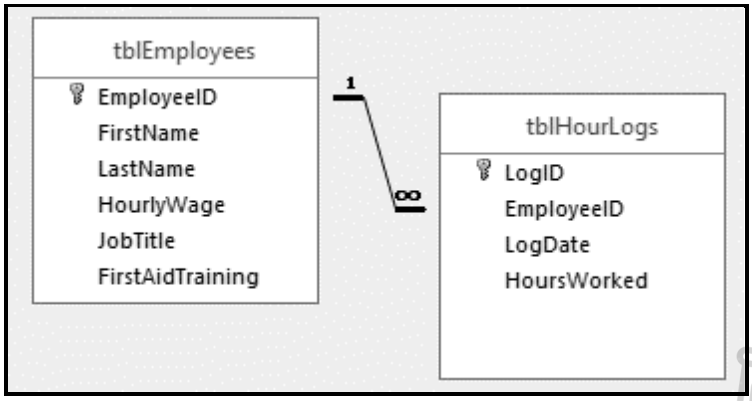
Example of the first eleven records of the **tblHourLogs** table:

LogID	EmployeeID	LogDate	HoursWorked
758	EMP566	2020/05/01	10
759	EMP984	2020/05/01	12
760	EMP881	2020/05/01	10
761	EMP773	2020/05/01	12
762	EMP909	2020/05/01	8
763	EMP892	2020/05/01	8
764	EMP166	2020/05/01	11
765	EMP102	2020/05/01	10
766	EMP044	2020/05/01	12
767	EMP883	2020/05/01	9
768	EMP002	2020/05/01	8

NOTE:

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

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



+

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE/
NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2020

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 28 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 3-10) include the marking grid for each question for using either one of the two programming languages.

- **Annexures E, F, G and H** (pages 12-28) contain examples of solutions for Questions 1 to 4 in programming code.

- Copies of **Annexures A, B, C and D** (pages 3-11) 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 – Colours]** Change the font size of cmbQ1_1 to 12pt✓ Add blue to cmbQ1_1✓✓ Set item index to 0 / set text to Green✓	**4**	
1.2	**Button [1.2 – Kite]** Create variable for area (rArea)✓ Test if iDiagA > iDiagB✓ rArea = iDiagA * iDiagB ✓ / 2✓ Display rArea on label ✓ formatted to one decimal✓ Else✓ Display error message✓	**8**	
1.3	**Button [1.3 – Binary number]** Extract decimal number from edit box✓, convert to integer✓ Initialise output string for binary value✓ Loop while ✓decimal number > 0✓ Remainder = decimal number MOD 2 ✓ (result 0 or 1) Decimal number ✓ = decimal number DIV 2 ✓ Add the remainder converted to a string✓ to beginning of output string ✓ Display binary number on label lblQ1_3 ✓ Alternatives: Loop while decimal number div 2 > 0 Repeat until decimal number = 0	**11**	
1.4	**Button [1.4 – Word game]** Initialise total to 0 ✓ Extract word from edit box ✓ and change to uppercase✓ Test if ✓ it is not a single word (has spaces) ✓ Clear edit box✓ Set focus to the edit box✓ Show error message✓ Else✓ Loop through word using length of word ✓ Extract character at loop variable✓ Test if it is a vowel ✓ add 3 to total ✓ Else Test for other alphabetical character✓ Add 2 to total ✓ Else Add 1 to total ✓ Display total value of word in richedit✓	**17**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID - DATABASE PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
2.1	**SQL statements**		
2.1.1	**Button [2.1.1 – List of employees]** `SELECT * FROM tblEmployees ORDER BY JobTitle, HourlyWage DESC` **Concepts:** SELECT * ✓ FROM tblEmployees ✓ ORDER BY JobTitle ✓ ,HourlyWage DESC ✓	**4**	
2.1.2	**Button [2.1.2 – Engineers]** `SELECT EmployeeID, LastName, FirstName FROM tblEmployees WHERE JobTitle LIKE "%Engineer%"` **Concepts:** SELECT EmployeeID, LastName, FirstName ✓ FROM tblEmployees ✓ WHERE JobTitle LIKE ✓ "%Engineer%"' ✓	**4**	

2.1.3	**Button [2.1.3 – Job titles]**
	`SELECT DISTINCT JobTitle FROM tblEmployees`
	`OR`
	`SELECT JobTitle FROM tblEmployees GROUP BY JobTitle`
	Concepts:
	SELECT DISTINCT ✓ JobTitle ✓ FROM tblEmployees ✓
	OR
	SELECT JobTitle FROM tblEmployees
	GROUP BY JobTitle
		3	

2.1.4	**Button [2.1.4 – Remove records]** `DELETE FROM tblHourLogs WHERE HoursWorked = 99` **Concepts:** DELETE FROM tblHourLogs ✓ WHERE HoursWorked = 99 ✓	**2**	

QUESTION 2: MARKING GRID – CONTINUE

2.1.5	**Button [2.1.5 – Overtime]** `SELECT LastName, FORMAT(SUM((HoursWorked - 8) * HourlyWage * 2), "CURRENCY") AS OvertimeAmt FROM tblEmployees E, tblHourLogs H WHERE E.EmployeeID = H.EmployeeID AND HoursWorked > 8 GROUP BY LastName` SELECT LastName, ✓ FORMAT(SUM ✓((HoursWorked - 8) ✓ * HourlyWage * 2 ✓), "CURRENCY") ✓ AS OvertimeAmt ✓ FROM tblEmployees E, tblHourLogs H ✓ WHERE E.EmployeeID = H.EmployeeID ✓ AND HoursWorked > 8 GROUP BY LastName ✓ **Concepts:** Retrieve LastName (1) Format to overtime pay as currency (1) Use SUM function correctly (1) Calculate number of hours worked overtime (1) Multiply result of above step by HourlyWage * 2 (1) Name calculated field OverTimeAmt (1) Extract data from tblEmployees and tblHourLogs (1) Test if there is a relationship between PK and FK (1) Group results by LastName (1)	**9**	
	Subtotal:	**22**	

2.2	**DATABASE MANIPULATION using Delphi code**		
2.2.1	**Button [2.2.1 – Employees with first aid]** Display column headings // Provided Initialise counter to 0 Set tblEmployees to start reading first record ✓ Loop while not tblEmployees.Eof ✓ Test if tblEmployees['FirstAidTraining'] = True ✓ Display EmployeeID, LastName and JobTitle ✓ Increment counter by 1 ✓ Go to next record in tblEmployees ✓ Display counter ✓ converted to string	**7**	

2.2.2	**Button [2.2.2 – Add new employee]**
	
	tblEmployees.Insert ✓
	tblEmployees['EmployeeID'] := 'EMP986'; ✓
	tblEmployees['FirstName'] := 'Robert'
	tblEmployees['LastName'] := 'Laubscher' ✓
	tblEmployees['HourlyWage'] := 195.00 ✓
	tblEmployees['JobTitle'] := 'Marine Engineer'
	tblEmployees['FirstAidTraining'] := True ✓
	tblEmployees.Post ✓
	
	Alternatives
	`tblEmployees['HourlyWage'] := 'R195.00';`
	`tblEmployees['HourlyWage'] := FloatToStrF(195.00,ffCurrency,6,2);`
	
	Concepts:
	Insert /Append (1)
	String variables – ID (1),
	FirstName and LastName (1)
	JobTitle and HourlyWage (1)
	FirstAidTraining (1)
	Post (1)
		6	

2.2.3	**Button [2.2.3 – Update hours worked]** Get number of hours (iHours or sHours) from edtQ2_2_3✓, typecast to Integer tblHourLogs.Edit ✓ tblHourLogs['HoursWorked'] ✓ := iHours ✓ tblHourLogs.Post ✓	**5**	
	Subtotal:	**18**	
	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID - OBJECT-ORIENTED PROGRAMMING

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.1	**Constructor method:** Header with correct parameter values (correct order, data types)✓ Assign customer ID parameter value to fCustomerID ✓ Assign container size parameter value to fContainerSize ✓ Assign storage period parameter value to fStoragePeriod Set fAmountPaid to 0 ✓	**4**	
3.1.2	**getAmountPaid method:** Function heading with real/double as return data type ✓ fAmountPaid assigned to result ✓	**2**	
3.1.3	**updateAmountPaid method:** Procedure heading with real/double parameter value ✓ Increment fAmountPaid ✓ Using the parameter value ✓	**3**	

3.1.4	**calculateCost method:**
	
	Function declared with double/real return data type
	If containerSize = 'S'
	  Set initial cost to 1000.00
	Else if containerSize = 'M'
	  Set initial cost to 1750.00
	Else
	  Set initial cost to 2500.00
	
	Cost = storagePeriod * initial cost
	Percentage discount = 10% for each increment of 6 months using the floor/trunk/div function or other applicable code
	Test if discount > 50
	  Set discount = 50
	Calculate discount value (Percentage x Amount)
	Return cost as result
		11	

	Adapted marking approach *Allocate marks for the following concepts:* Function declared✓ with a return data type ✓ *A test for the container size✓✓* *Assigning the correct cost✓ per size✓* *Calculate cost* ✓ using the correct formula *Determine percentage discount using months*✓ *Limit discount percentage to 50*✓ *Calculate discount value* ✓ *Return cost as result* ✓		
3.1.5	**toString method:** Labels (Customer ID, Container size, Storage period, Amount paid) ✓ Correct attributes✓ Correct conversions (amountPaid – float; storagePeriod – integer) ✓	**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 customer ID from edit box✓ Extract first character ✓ of selected item in list box✓ Extract number of months from spin edit✓ *Instantiate the objTransaction object:* objTransaction:= ✓ TTransaction.Create✓ (customerID,size,months) Disable **btnQ3_2_1** ✓	**7**	
3.2.2 (a)	**Button [3.2.2 (a) – Display amount due]** Call the calculateCost method ✓ Display the amount due ✓ on **pnlQ3_2_2** ✓	**3**	
3.2.2 (b)	**Button [3.2.2 (b) – Process payment]** Retrieve the amount from **edtQ3_2_2** ✓ Call updateAmountPaid ✓ with extracted amount as argument Calculate amount due using the object methods: Amount due := calculateCost - getAmountPaid Concepts: Correct method call✓ Correct calculation✓ Display amount due on **pnlQ3_2_2** ✓	**5**	
3.2.3	**Button [3.2.3 – View details]** Call the toString method ✓ Display in the richedit ✓	**2**	
	Subtotal: Form class	**17**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID–PROBLEM SOLVING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
SECTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1.1	**Button [4.1.1 – Create harbour containers]** Loop 50 times ✓ Generate real numbers ✓between 1 and 99 (inclusive)✓ Rounded to 1 decimal Store the values in the array arrContainers✓	**4**	
4.1.2	**Button [4.1.2 – Display harbour containers]** Set index to 0✓ Loop 5 times (rows) ✓ sLine := ' ' ✓ Loop 10 times (columns) ✓ Increment the index with 1 ✓ sLine := sLine + FloatToStr(arrContainers[iIndex]) ✓ + #9;✓ Display sLine in the rich edit redQ4_1_2✓ **Alternative**: sLine := sLine + FloatToStr(arrContainers[c+(10*(r-1)]) (3) + #9; (1) Concept: Using loop variables (1) Calculation (1) Floattostr (1) Space (1)	**8**	

4.2	**Button [4.2 – Containers loaded to be shipped]** Set TotalTons to zero (0) ✓ Initialise string variable used for display ✓ Set array index (1) ✓ Loop index < 50✓ If (totalTons + weight of the containers at Index) ✓ <= 200✓ then Add weight of the containers at Index✓ in array to TotalTons✓ Add the weight of the containers at Index✓ in array and add #9✓ to sLine✓ Increment the index✓ Outside loop: Display the weight of the containers loaded onto the ship in the richedit ✓ Display the total weight of the containers loaded on the panel ✓ Assign internal file name to external file name✓ Rewrite ✓ Write the weight of the containers loaded onto the ship to the text file✓ Close the text file ✓ Alternative For loop from 1 to 50 (3 marks)	**18**	

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

SUMMARY OF LEARNER'S MARKS:

CENTER NUMBER:		**LEARNER'S EXAM 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

```
unit Question1_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, ExtCtrls, Spin, pngimage, ComCtrls, Math,
Buttons;

TfrmQuestion1 = class(TForm)
    GroupBoxQ1_1: TGroupBox;
    GroupBoxQ1_4: TGroupBox;
    edtQ1_4: TEdit;
    Label9: TLabel;
    redQ1_4: TRichEdit;
    GroupBoxQ1_3: TGroupBox;
    edtQ1_3: TEdit;
    btnQ1_3: TButton;
    Label10: TLabel;
    lblQ1_3: TLabel;
    btnQ1_1: TButton;
    cmbQ1_1: TComboBox;
    btnClose: TBitBtn;
    GroupBoxQ1_2: TGroupBox;
    lblQ1_2: TLabel;
    btnQ1_2: TButton;
    btnQ1_4: TButton;
    Image2: TImage;
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion1: TfrmQuestion1;

implementation

{$R *.dfm}

//=====================================================================
//                      Question 1.1 – 4 marks
//=====================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
// Question 1.1
  cmbQ1_1.Font.Size := 12;
  cmbQ1_1.Items.Add('Blue');
  cmbQ1_1.ItemIndex := 0;
end;
```

```
//=====================================================================
//                      Question 1.2 – 8 marks
//=====================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  iDiagA, iDiagB: integer;
  rArea: real;
begin
//Provided code

  iDiagA := StrToInt(InputBox('Diagonal one of kite(cm)', 'Enter the
value of diagonal one (A): ', '20'));
  iDiagB := StrToInt(InputBox('Diagonal two of kite (cm)',
      'Enter the value of diagonal two (B):', ''));

// Question 1.2
  if iDiagA > iDiagB then
  begin
    rArea := (iDiagA * iDiagB) / 2;
    lblQ1_2.Caption := 'The area of the kite is ' + FloatToStrF
      (rArea, ffFixed, 10, 1) + ' square cm.';
  end
  else
  begin
    ShowMessage('The value of diagonal two (B) must be less than the
                     value of diagonal one (A)');
  end;
end;

//=====================================================================
//                      Question 1.3 – 11 marks
//=====================================================================

procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
// Provided code
var
  iNumber, iRemainder: integer;
  sBinary: String;
begin
// Question 1.3
  iNumber := StrToInt(edtQ1_3.Text);
  sBinary := '';
  while iNumber <> 0 do
  begin
    iRemainder := iNumber MOD 2;
    iNumber := iNumber DIV 2;
    sBinary := IntToStr(iRemainder) + sBinary;
  end;
  lblQ1_3.Caption := 'Binary number: ' + sBinary;
end;
```

```
//=====================================================================
//                    Question 1.4 – 17 marks
//=====================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sWord: String;
  iTotal, iValue, iRand, iLen, iCount: integer;
  cChar: char;
begin
// Provided code
  redQ1_4.Clear;
// Question 1.4
  iTotal := 0;
  sWord := Uppercase(edtQ1_4.Text);

  if pos(' ', sWord) = 0 then
  begin
    iLen := Length(sWord);

    for iCount := 1 to iLen do
    begin
      cChar := sWord[iCount];

      if cChar in ['A', 'E', 'I', 'O', 'U'] then
        iTotal := iTotal + 3
      else
         if cChar in ['A'..'Z'] then
           iTotal := iTotal + 2
        else
           iTotal := iTotal + 1;
    end;

    redQ1_4.lines.Add('Total number of points: ' + IntToStr(iTotal));
  end
  else
  begin
    edtQ1_4.Clear;
    edtQ1_4.SetFocus;
    ShowMessage('Disqualified - more than one word was entered.');
  end;
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, Buttons, ExtCtrls, ConnectDB_U, DB, ADODB,
Grids, DBGrids, ComCtrls, DateUtils, DBCtrls;

type
  TfrmQuestion2 = class(TForm)
    pnlBtns: TPanel;
    btnRestoreDB: TBitBtn;
    grpTblHourLogs: TGroupBox;
    grpTblEmployees: TGroupBox;
    dbgEmployees: TDBGrid;
    dbgHourLogs: TDBGrid;
    tabsQ2_2ADO: TTabSheet;
    tabsQ2_1SQL: TTabSheet;
    btnQ2_2_1: TButton;
    redQ2: TRichEdit;
    grpresults: TGroupBox;
    dbgSQL: TDBGrid;
    grpOutput: TGroupBox;
    pgcTabs: TPageControl;
    pnlTables: TPanel;
    btnQ2_1_1: TButton;
    btnQ2_1_2: TButton;
    btnQ2_1_3: TButton;
    btnQ2_1_4: TButton;
    btnQ2_1_5: TButton;
    btnQ2_2_3: TButton;
    btnQ2_2_2: TButton;
    procedure btnRestoreDBClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure btnQ2_2_3Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);

  private
  public
  end;

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

```
// --- Global variables provided ---
  tblEmployees, tblHourLogs: TADOTable;
  qryDB: TADOQuery;
implementation
{$R *.dfm}
{$R+}
```

Question 2.1 - SQL section

```
//=====================================================================
//                    Question 2.1.1 – 4 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
  var
    sSQL1: String;
begin
  sSQL1 := 'SELECT * FROM tblEmployees ' +
           'ORDER BY JobTitle, HourlyWage DESC';

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

//=====================================================================
//                    Question 2.1.2 – 4 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
  var
    sSQL2: String;
begin
  sSQL2 := 'SELECT EmployeeID, LastName, FirstName FROM tblEmployees ' +
           'WHERE JobTitle LIKE "%Engineer%"';

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

//=====================================================================
//                    Question 2.1.3 – 3 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
  var
    sSQL3: String;
begin
  sSQL3 := 'SELECT DISTINCT JobTitle FROM tblEmployees';

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

```
//=====================================================================
//                      Question 2.1.4 – 2 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
  var
    sSQL4: String;
    bChange: boolean;
begin
  sSQL4 := 'DELETE FROM tblHourLogs ' +
           'WHERE HoursWorked = 99';

  // Provided code - do not change
  dbCONN.ExecuteSQL(sSQL4, dbgHourLogs);
  if bChange then
    begin
      MessageDlg('Database updated', mtInformation, [mbOK], 0);
    end;
end;

//=====================================================================
//                      Question 2.1.5 – 9 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
  var
    sSQL5: String;
begin
  sSQL5 := 'SELECT LastName, FORMAT(SUM((HoursWorked - 8) * (HourlyWage
* 2)), "CURRENCY") ' +
           'AS OvertimeAmt ' +
           'FROM tblEmployees E, tblHourLogs H ' +
           'WHERE E.EmployeeID = H.EmployeeID ' +
           'GROUP BY LastName';

  // Provided code - do not change
  dbCONN.DBExtract(sSQL5);
end;
```

Question 2.2 - Delphi section

```
//=====================================================================
//                    Question 2.2.1 – 7 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
  var
    iCount: integer;
begin
  // Provided code
  redQ2.Clear;
  redQ2.Paragraph.TabCount := 3;
  redQ2.Paragraph.Tab[0] := 100;
  redQ2.Paragraph.Tab[1] := 190;
  redQ2.SelAttributes.Style := [fsBold, fsUnderline];
  redQ2.Lines.Add('EmployeeID' + #9 + 'LastName' + #9 + 'JobTitle');

  // Enter your code here for Question 2.2.1
  iCount := 0;
  tblEmployees.First;
  while tblEmployees.Eof = False do
    begin
      if tblEmployees['FirstAidTraining'] = True then
        begin
          redQ2.Lines.Add(tblEmployees['EmployeeID'] + #9 +
                          tblEmployees['LastName'] + #9 +
                          tblEmployees['JobTitle']);
          Inc(iCount);
        end;
      tblEmployees.Next;
    end;
  redQ2.Lines.Add(#10 + 'Total number of employees with first aid
     training: ' + IntToStr(iCount));
end;

//=====================================================================
//                    Question 2.2.2 – 6 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
begin
  // Question 2.2.2
  tblEmployees.Insert();
  tblEmployees['EmployeeID'] := 'EMP876';
  tblEmployees['FirstName'] := 'Robert';
  tblEmployees['LastName'] := 'Laubscher';
  tblEmployees['HourlyWage'] := 195.00;
  tblEmployees['JobTitle'] := 'Marine Engineer';
  tblEmployees['FirstAidTraining'] := True;
  tblEmployees.Post();
end;
```

```
//=====================================================================
//                    Question 2.2.3 – 5 marks
//=====================================================================
procedure TfrmQuestion2.btnQ2_2_3Click(Sender: TObject);
  var
    iHours: Integer;
begin
  // Question 2.2.3
  iHours := StrToInt(edtQ2_2_3.text);
  tblHourLogs.Edit();
  tblHourLogs['HoursWorked'] := iHours;
  tblHourLogs.Post();
end;

//=====================================================================
//              Setup DB connections - DO NOT CHANGE!
//=====================================================================
{$REGION DB CONNECTION}
// ====================================================
procedure TfrmQuestion2.btnRestoreDBClick(Sender: TObject);
begin
  // Restores the Database
  dbCONN.RestoreDatabase(dbgEmployees, dbgHourLogs, dbgSQL);
  redQ2.Clear;
  // Formatting field datatypes
  tblEmployees := dbCONN.tblOne;
  tblHourLogs := dbCONN.tblMany;
  qryDB := dbCONN.qry;
end;

// ====================================================
procedure TfrmQuestion2.FormClose(Sender: TObject; var Action:
TCloseAction);
begin // Disconnects from database and closes all open connections
  dbCONN.dbDisconnect;
end;

// ====================================================
procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  CurrencyString := 'R';
  ShortDateFormat := 'YYYY/MM/DD';
  // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblEmployees := dbCONN.tblOne;
  tblHourLogs := dbCONN.tblMany;
  qryDB := dbCONN.qry;
  dbCONN.SetupGrids(dbgEmployees, dbgHourLogs, dbgSQL);
  pgcTabs.ActivePageIndex := 0;
end;
// ====================================================
{$ENDREGION}

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

OBJECT CLASS UNIT:

```
unit Transaction_U;

interface

type
  TTransaction = class(TObject)
    private
      var
        fCustomerID: String;
        fContainerSize: char;
        fStoragePeriod: integer;
        fAmountPaid: real;
    public
      constructor create(sCustomerID: String; cContainerSize: char;
           iStoragePeriod: integer);
      function getAmountPaid: real;
      procedure updateAmountPaid(pAmountPaid: real);
      function calculateCost: real;
      function toString: String;
  end;

implementation

{ TTransaction }

uses
  SysUtils, Math;

//=====================================================================
//                    Question 3.1.1 – 4 marks
//=====================================================================
constructor TTransaction.create(sCustomerID: String; cContainerSize:
                                char; iStoragePeriod: integer);
begin
  fCustomerID := sCustomerID;
  fContainerSize := cContainerSize;
  fStoragePeriod := iStoragePeriod;
  fAmountPaid := 0.00;
end;

//=====================================================================
//                    Question 3.1.2 – 2 marks
//=====================================================================

function TTransaction.getAmountPaid: real;
begin
  result := fAmountPaid;
end;
```

```
//=====================================================================
//                      Question 3.1.3 – 3 marks
//=====================================================================

procedure TTransaction.updateAmountPaid(pAmountPaid: real);
begin
  fAmountPaid := fAmountPaid + pAmountPaid;
end;

//=====================================================================
//                      Question 3.1.4 – 11 marks
//=====================================================================

function TTransaction.calculateCost: real;
  var
    rCost, rPercentageDiscount: real;
begin
  case fContainerSize of
    'S': rCost := 1000.00;
    'M': rCost := 1750.00;
    'L': rCost := 2500.00;
  end;

  rCost := fStoragePeriod * rCost;
  rPercentageDiscount := Floor(fStoragePeriod / 6) * 0.1;
  if rPercentageDiscount > 0.5 then
  begin
     rPercentageDiscount := 0.5;
  end;
  result := rCost - (rCost * rPercentageDiscount);
end;

//=====================================================================
//                      Question 3.1.5 – 3 marks
//=====================================================================

function TTransaction.toString: String;
begin
  result := 'Customer ID: ' + fCustomerID + #10 +
            'Container size: ' + fContainerSize + #10 +
            'Storage period (months): ' + IntToStr(fStoragePeriod) + #10
            + 'Amount paid: ' + Format('%.2m', [fAmountPaid]);
end;

end.
```

MAIN CLASS UNIT:

```
unit Question3_U;

interface

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

type
  TQuestion3 = class(TForm)
    gbxQ3_2_1: TGroupBox;
    gbxContainerSize: TGroupBox;
    lstQ3_2_1: TListBox;
    gbxCustomerID: TGroupBox;
    edtQ3_2_1: TEdit;
    gbxNumberOfMonths: TGroupBox;
    spnQ3_2_1: TSpinEdit;
    gbxQ3_2_3: TGroupBox;
    redQ3_2: TRichEdit;
    btnQ3_2_1: TButton;
    btnQ3_2_3: TButton;
    btnReset: TButton;
    gbxQ3_2_2: TGroupBox;
    btnQ3_2_2_b: TButton;
    gbxAmount: TGroupBox;
    edtQ3_2_2: TEdit;
    pnlQ3_2_2: TPanel;
    btnQ3_2_2_a: TButton;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
    procedure btnQ3_2_2_bClick(Sender: TObject);
    procedure btnQ3_2_2_aClick(Sender: TObject);
  private

  public

  end;

var
  Question3: TQuestion3;

implementation

{$R *.dfm}

uses
  Transaction_U;

var
  objTransaction: TTransaction;
```

```
//=========================================================================
//                    Question 3.2.1 – 7 marks
//=========================================================================

procedure TQuestion3.btnQ3_2_1Click(Sender: TObject);
var
    sCustomerID: String;
    cContainerSize: char;
    iStoragePeriod: integer;
begin
  //  Question 3.2.1
  sCustomerID := edtQ3_2_1.Text;
  cContainerSize := lstQ3_2_1.Items[lstQ3_2_1.ItemIndex][1];
  iStoragePeriod := spnQ3_2_1.Value;

  objTransaction := TTransaction.create(sCustomerID,cContainerSize,
                                        iStoragePeriod);

  btnQ3_2_1.Enabled := False;
end;

//=========================================================================
//                    Question 3.2.2(a) – 3 marks
//=========================================================================

procedure TQuestion3.btnQ3_2_2_aClick(Sender: TObject);
begin
  //  Question 3.2.2 (a)
  pnlQ3_2_2.Caption := 'Amount due: ' + Format('%.2m',
[objTransaction.calculateCost]);
end;

//=========================================================================
//                    Question 3.2.2(b) – 5 marks
//=========================================================================

procedure TQuestion3.btnQ3_2_2_bClick(Sender: TObject);
var
    rAmountPaid: real;
begin
  //  Question 3.2.2 (b)
  rAmountPaid := StrToFloat(edtQ3_2_2.Text);

  objTransaction.updateAmountPaid(rAmountPaid);

  pnlQ3_2_2.Caption := 'Amount due: ' + Format('%.2m',
      [objTransaction.calculateCost - objTransaction.getAmountPaid]);
end;
```

```
//=====================================================================
//                    Question 3.2.3 – 2 marks
//=====================================================================

procedure TQuestion3.btnQ3_2_3Click(Sender: TObject);
begin
  //  Provided code
  redQ3_2.Clear;
  //  Question 3.2.3
  redQ3_2.Lines.Add(objTransaction.toString);
end;

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

procedure TQuestion3.FormCreate(Sender: TObject);
begin
  lstQ3_2_1.ItemIndex := 0;
end;

procedure TQuestion3.btnResetClick(Sender: TObject);
begin
    btnQ3_2_1.Enabled := True;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;

interface

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


type
  TfrmQuestion4 = class(TForm)
    GroupBox1: TGroupBox;
    redQ4_1: TRichEdit;
    btnQ4_1_2: TButton;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    Image1: TImage;
    redQ4_2: TRichEdit;
    pnlQ4: TPanel;
    btnQ4_2: TButton;
    btnQ4_1_1: TButton;
    procedure btnQ4_2Click(Sender: TObject);
    procedure btnQ4_1_2Click(Sender: TObject);
    procedure btnQ4_1_1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);

  private
    { Private declarations }
    tFile : TextFile;
    arrContainers : array [1..50] of real;
    // arrShip    : array [1..5, 1..10] of real;  optional
  public
    { Public declarations }
  end;
//=============================================================
// Provided code - DO NOT CHANGE
//=============================================================

 const
  arrTempContainers : array[1..50] of real =
                      (31.2,43.4,5.1,41.2,52.6,41.9,97,49.3,15.8,39.5,
                      78.8,96.3,67.8,47.2,31.4,18.4,27,1.4,33.5,16.5,
                      58.3,9.9,62.9,11.8,62.5,59,13.4,49.8,60.4,74.5,
                      13.5,67.7,94,39.4,47.4,13.1,26.2,63,89,22.3,
                      54,16.9,38.6,46.2,22.5,11.4,65.1,48.6,41.4,47.9);
var
  frmQ4: TfrmQ4;

implementation

{$R *.dfm}
```

```
//=============================================================================
//                          Question 4.1.1 – 4 marks
//=============================================================================

procedure TfrmQuestion4.btnQ4_1_1Click(Sender: TObject);
var
  iRow : integer;
begin
 // Provided code
  btnQ4_2.Enabled := false;
  redQ4_2.Clear;
  redQ4_1.Clear;
  pnlQ4.Caption := ' ' ;

// Question 4.1.1

for iRow := 1 to 50 do
   arrContainers[iRow] := RoundTo(Random() + Random(99),-1);

btnQ4_1_2.Enabled := true;
 //for iRow := 1 to 50 do
//   arrContainers[iRow] := arrTempContainers[iRow];

end;
//=============================================================================
//                          Question 4.1.2 – 8 marks
//=============================================================================

procedure TfrmQuestion4.btnQ4_1_2Click(Sender: TObject);
var
   iRow, iCol, iIndex : integer;
   sLine : String;

begin
  // Provided code
  redQ4_1.Clear;
  btnQ4_2.Enabled := true;

  // Question 4.1.2
  iIndex := 0;
  for iRow := 1 to 5 do
   begin
    sLine := '';
    for iCol := 1 to 10 do
      begin
       Inc(iIndex);
       sLine := sLine  + FloatToStr(arrContainers[iIndex])+ #9;
      end;
    redQ4_1.Lines.Add(sLine);
  end;
end;
```

```
//=============================================================================
//                      Question 4.2 – 18 marks
//=============================================================================

procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
 rTotalTons : real;
 iRow,iCol,iIndex : integer;
 sLine : String;

begin
// Provided code
   redQ4_2.Clear;

//Question 4.2

  rTotalTons := 0;
  iIndex :=  1;
  sLine := '';
  while (iIndex <= 50) and (rTotalTons <= 200) do
   begin
   rTotalTons := rTotalTons + arrContainers[iIndex];
     if (rTotalTons > 200) then
       rTotalTons := rTotalTons - arrContainers[iIndex]
     else
       sline := sLine + FloatToStr(arrContainers[iIndex])+#9 ;

     Inc(iIndex);
    end;

// Alternative
  iRow := 1;
  iIndex := 1;    //one dim array
  sLine := '';

  while (rTotalTons < 200) and (iRow < 5) do
     begin
       iCol := 1;
       while (iCol < 11) and (iIndex <= 50) do
         begin
           arrShip[iRow,iCol] := arrContainers[iIndex];
           if rTotalTons + arrShip[iRow,iCol] <= 200 then //rTotalTons +
                                  arrContainers[iIndex] <= 200
             begin
                sline := sLine + FloatToStr(arrShip[iRow,iCol])+#9 ;
                rTotalTons := rTotalTons + arrShip[iRow,iCol];
                Inc(iCol);
             end;
           Inc(iIndex);
         end;
     Inc(iRow);
     end;
  AssignFile(tFile,'Tons.txt');
  Rewrite(tFile);
   writeln(tfile,sline);
   CloseFile(tFile);
   redQ4_2.Lines.Add(sLine);
```

```
   pnlQ4.Caption := 'Total weight of load: ' + FloatToStr (rTotalTons );

end;

// Provided code
procedure TfrmQuestion4.FormCreate(Sender: TObject);
begin
  btnQ4_1_2.Enabled := false;
  btnQ4_2.Enabled := false;
end;

end.
```