basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

# SENIOR CERTIFICATE EXAMINATIONS/
# NATIONAL SENIOR CERTIFICATE EXAMINATIONS

## INFORMATION TECHNOLOGY P1

## 2022

**MARKS: 150**

**TIME: 3 hours**

**This question paper consists of 24 pages and 2 data pages.**

## INSTRUCTIONS AND INFORMATION

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

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

3. This question paper is set with programming terms that are specific to Delphi programming language. 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 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. 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: **DataENGJun2022.exe**.
- Click on the 'Extract' button.
- Enter the following password: **del$Jun@2022**

Once extracted, the following list of files will be available in the folder **DataENGJun2022**:

**Question 1:**
Question1_P.dpr
Question1_P.dproj
Question1_P.res
Question1_U.dfm
Question1_U.pas

**Question 2:**
ChikoroDrivingSchool.mdb
ChikoroDrivingSchool - Copy.mdb
ConnectDB_U.pas
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

**Question 3:**
DataQ3.txt
DeliveryTrip_U.pas
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas

**Question 4:**
Question4_P.dpr
Question4_P.dproj
Question4_P.res
Question4_U.dfm
Question4_U.pas

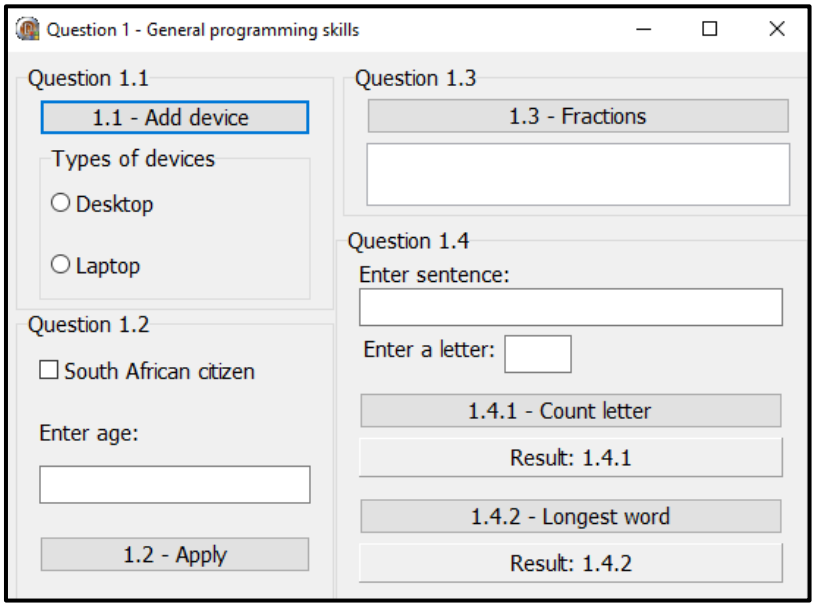
## SECTION A

## QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your 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 1.4 that follow.

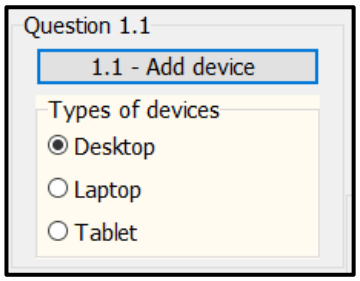
### 1.1 Button [1.1 – Add device]

The radio group **rgpQ1_1** currently contains the items 'Desktop' and 'Laptop'.

Write code to do the following:

- Add the item 'Tablet' to the radio group **rgpQ1_1**.
- Set the colour of the radio group to 'Cream'.
- Show the first item 'Desktop' to be selected.

Example of output:




(3)

1.2 **Button [1.2 – Apply]**

South African citizens of 16 years and older can apply for a South African ID card.

The user must tick the check box South African citizen if the person is a South African citizen and enter the person's age in the edit box **edtQ1_2**.

The following code has been provided:

```
const
   CURRENT_YEAR = 2022;
var
   iAge : integer;
```

Write code to do the following:

1.2.1 Extract and store the age that was entered in the edit box **edtQ1_2** in the provided **iAge** variable. (2)

1.2.2 Test whether the check box has been ticked. If not, use the ShowMessage dialogue box to display the message 'The applicant must be a South African citizen.' (2)

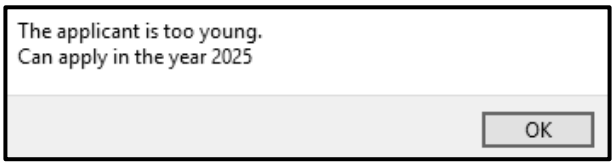
1.2.3 Test if the age that was entered meets the criteria of 16 years or older. If not, do the following:

- Use the constant variable CURRENT_YEAR and do a calculation to determine the year that the applicant will be eligible to apply for an ID card.
- Use the ShowMessage dialogue box to display a message consisting of the following lines of text:
  The first line of text: 'The applicant is too young.'
  The second line of text: 'Can apply in the year' <yearToApply> (5)

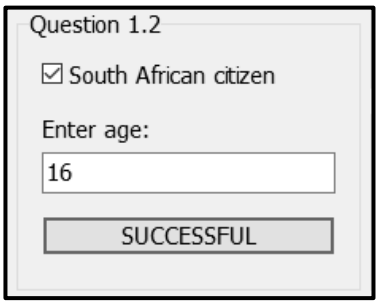
1.2.4 If both criteria are met, change the text of the button **btnQ1_2** to 'SUCCESSFUL'. (2)

Example of output if the check box for South African citizen is not ticked and the value of 17 is entered:

Example of output if the check box for South African citizen is ticked and the value of 13 is entered:




Example of output if the check box for South African citizen is ticked and the value of 16 is entered:




1.3 **Button [1.3 – Fractions]**

A sequence of terms made up of fractions must be added until the sum of the terms just exceeds the value of 4.

The format of the sequence is as follows:

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\cdots+n$$

The first term in the sequence must have:

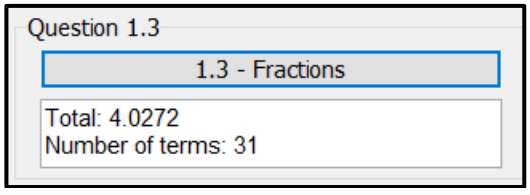
- A numerator of 1 (the top part of the fraction)
- A denominator of 1 (the bottom part of the fraction)

The pattern is continued by incrementing the denominator by the value of 1.

Write code that uses a **conditional loop** to do the following:

- Add the terms in the sequence until the sum just exceeds the value of 4.
- Display the sum of the terms in the **redQ1_3** output area, formatted to four decimal places.
- Display the number of terms in the sequence in the **redQ1_3** output area.

Example of output:




(10)

1.4 The user must enter a sentence in the provided edit box **edtQ1_4**. Write code to do the tasks described in QUESTION 1.4.1 and QUESTION 1.4.2.

Code has been provided in each button to extract the sentence from the **edtQ1_4** edit box.

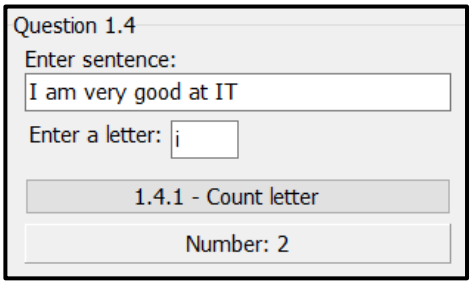
1.4.1 **Button [1.4.1 – Count letter]**

The user must enter a letter in the edit box **edtQ1_4_1**.

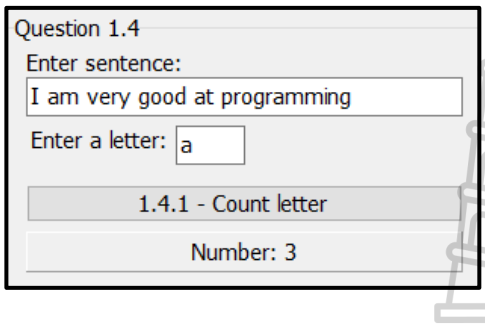
Write code to do the following:

- Extract the letter that was entered from the edit box **edtQ1_4_1**.
- Count the number of times the letter appears in the sentence that was entered in the edit box **edtQ1_4**, irrespective of whether the letter that was entered is a small or capital letter.
- Display the result on the panel **pnlQ1_4_1**, as shown in the example of output below.

Example of output if the sentence is 'I am very good at IT' and the letter 'i' is entered:




Example of output if the sentence is 'I am very good at programming' and the letter 'a' is entered:



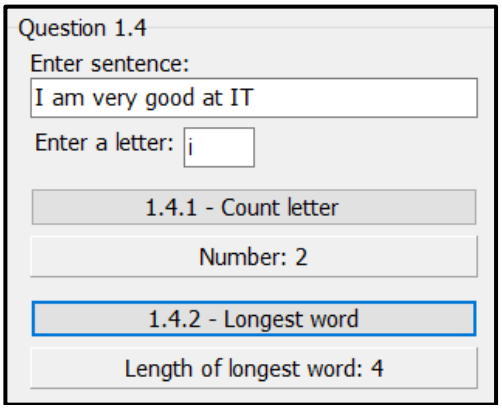

(8)

1.4.2 **Button [1.4.2 – Longest word]**

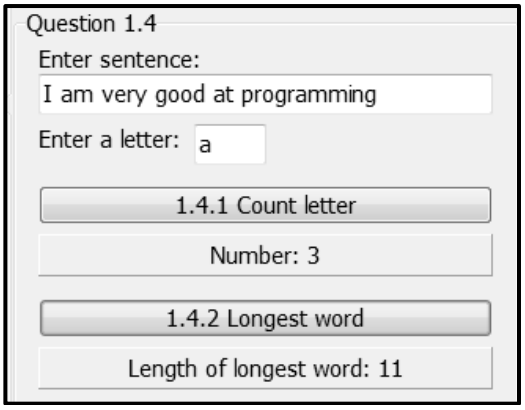
Write code to do the following:

- Determine the length of the longest word that appears in the sentence that was entered in the edit box **edtQ1_4**.
- Display the length of the longest word on the panel **pnlQ1_4_2**, as shown in the example of output that follows.

Example of output if the sentence 'I am very good at IT' is entered:




Example of output if the sentence 'I am very good at programming' is entered:




(8)

- 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

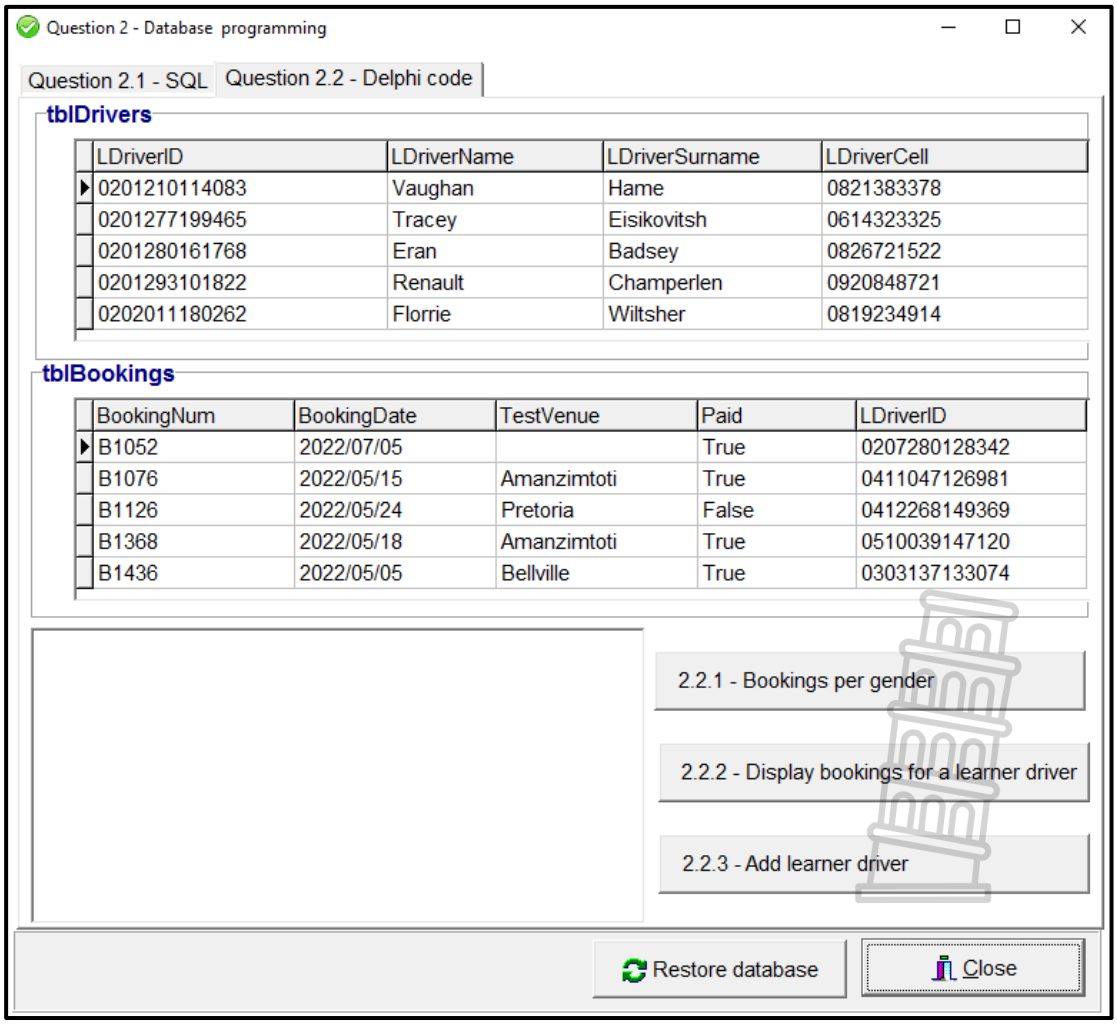
## QUESTION 2: SQL AND DATABASE PROGRAMMING

Chikoro driving school uses a database to manage bookings of learner drivers.

The data pages attached at the end of the question paper provide information on the design of the 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 that follow 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.

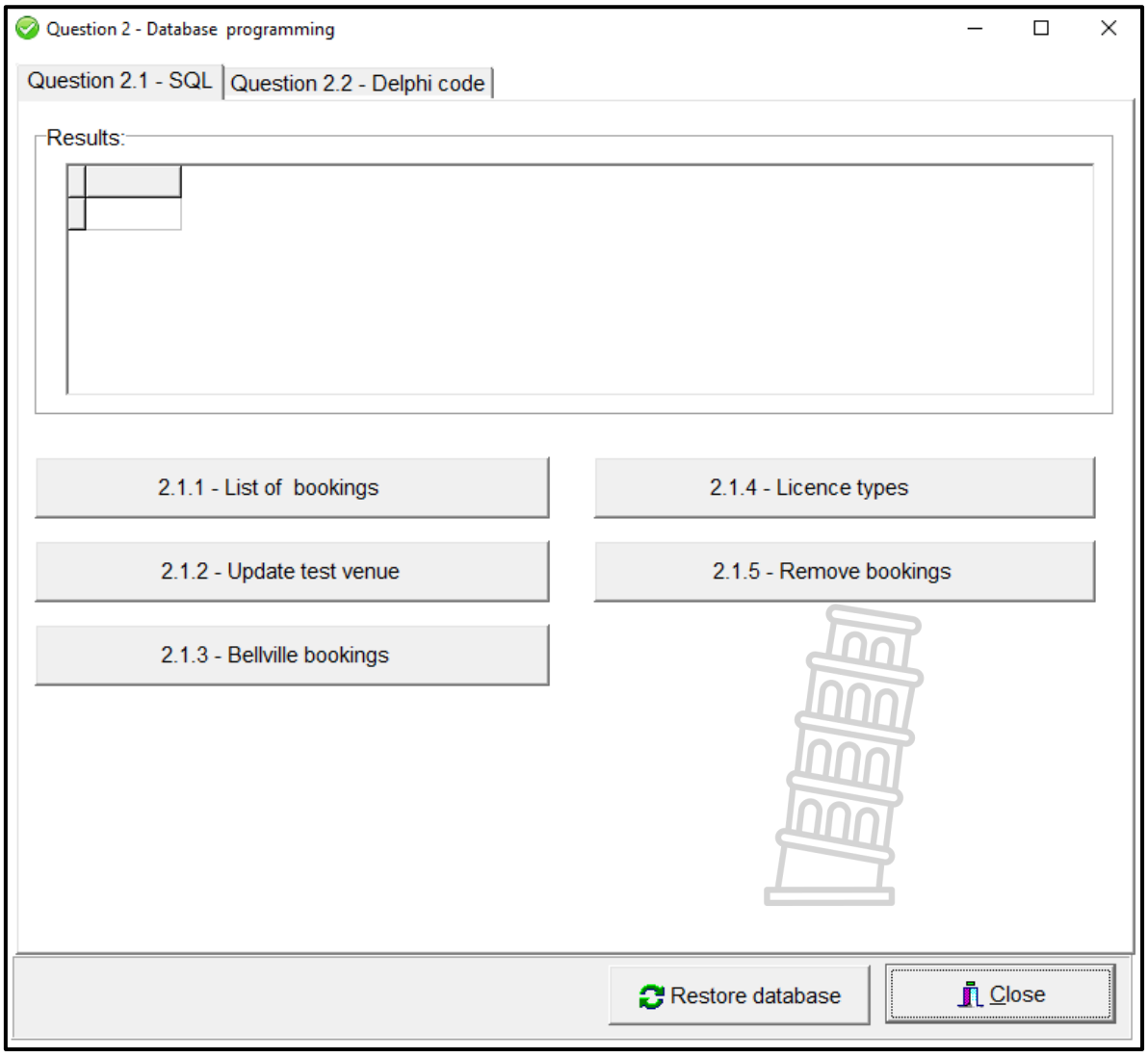
**NOTE:**

- The 'Restore database' button is provided to restore the data contained in the database to the original content.
- Code is provided to link the GUI components to the database. Do NOT change any of the code provided.
- TWO variables are declared as public variables as described in the table below.

| Variable | Data type | Description |
|---|---|---|
| `tblLDrivers` | `TAdoTable` | Stores the information of learner drivers |
| `tblBookings` | `TAdoTable` | Stores bookings for drivers' licence tests |

## 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 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 - List of bookings]**

Display the booking numbers and booking dates in the **tblBookings** table sorted according to booking dates.

Example of output:

| BookingNum | BookingDate |
|---|---|
| B5743 | 2021/05/16 |
| B3134 | 2021/06/02 |
| C4214 | 2022/05/04 |
| T9047 | 2022/05/04 |
| C7281 | 2022/05/04 |
| T3232 | 2022/05/04 |

(3)

2.1.2 **Button [2.1.2 - Update test venue]**

A few of the bookings in table **tblBookings** have not been allocated a test venue. Update the table so that 'Headquarters' is the test venue for these bookings.

Example of output before update is done:

| BookingNum | BookingDate | TestVenue | Paid | LDriverID |
|---|---|---|---|---|
| B1052 | 2022/07/05 | | True | 0207280128342 |
| B1076 | 2022/05/15 | Amanzimtoti | True | 0411047126981 |
| B1126 | 2022/05/24 | Pretoria | False | 0412268149369 |
| B1368 | 2022/05/18 | Amanzimtoti | True | 0510039147120 |
| B1436 | 2022/05/05 | Bellville | True | 0303137133074 |

Example of output after update is done:

| BookingNum | BookingDate | TestVenue | Paid | LDriverID |
|---|---|---|---|---|
| B1052 | 2022/07/05 | Headquarters | True | 0207280128342 |
| B1076 | 2022/05/15 | Amanzimtoti | True | 0411047126981 |
| B1126 | 2022/05/24 | Pretoria | False | 0412268149369 |
| B1368 | 2022/05/18 | Amanzimtoti | True | 0510039147120 |
| B1436 | 2022/05/05 | Bellville | True | 0303137133074 |
| B1602 | 2022/05/25 | Bellville | True | 0211218127327 |

(4)

2.1.3 **Button [2.1.3 - Bellville bookings]**

Display the booking number and the learner driver name and surname of all bookings with Bellville as test venue.

Example of output of the first five records:

| BookingNum | IDriverName | IDriverSurname |
|---|---|---|
| C4214 | Kai | Witherby |
| T9047 | Ansell | Archley |
| C7281 | Marcie | Caps |
| T5120 | Ingmar | Itzik |
| C4431 | Kippar | Murrell |
| B1436 | Alano | Troppmann |

(4)

2.1.4 **Button [2.1.4 - Licence types]**

The first character of the booking number in **tblBookings** represents the type of license. The types of licences are:

B: Bicycle licence
C: Car licence
T: Truck licence

Count and display the number of bookings for each type of licence in the table **tblBookings**.

Example of output:

| LicenceTypes | Number |
|---|---|
| B | 121 |
| C | 119 |
| T | 111 |

(5)

2.1.5 **Button [2.1.5 – Remove bookings]**

Due to maintenance at the Pretoria training centre, all the bookings from 18 May 2022 until 25 May 2022 have been cancelled. Remove all these bookings from the **tblBookings** table.

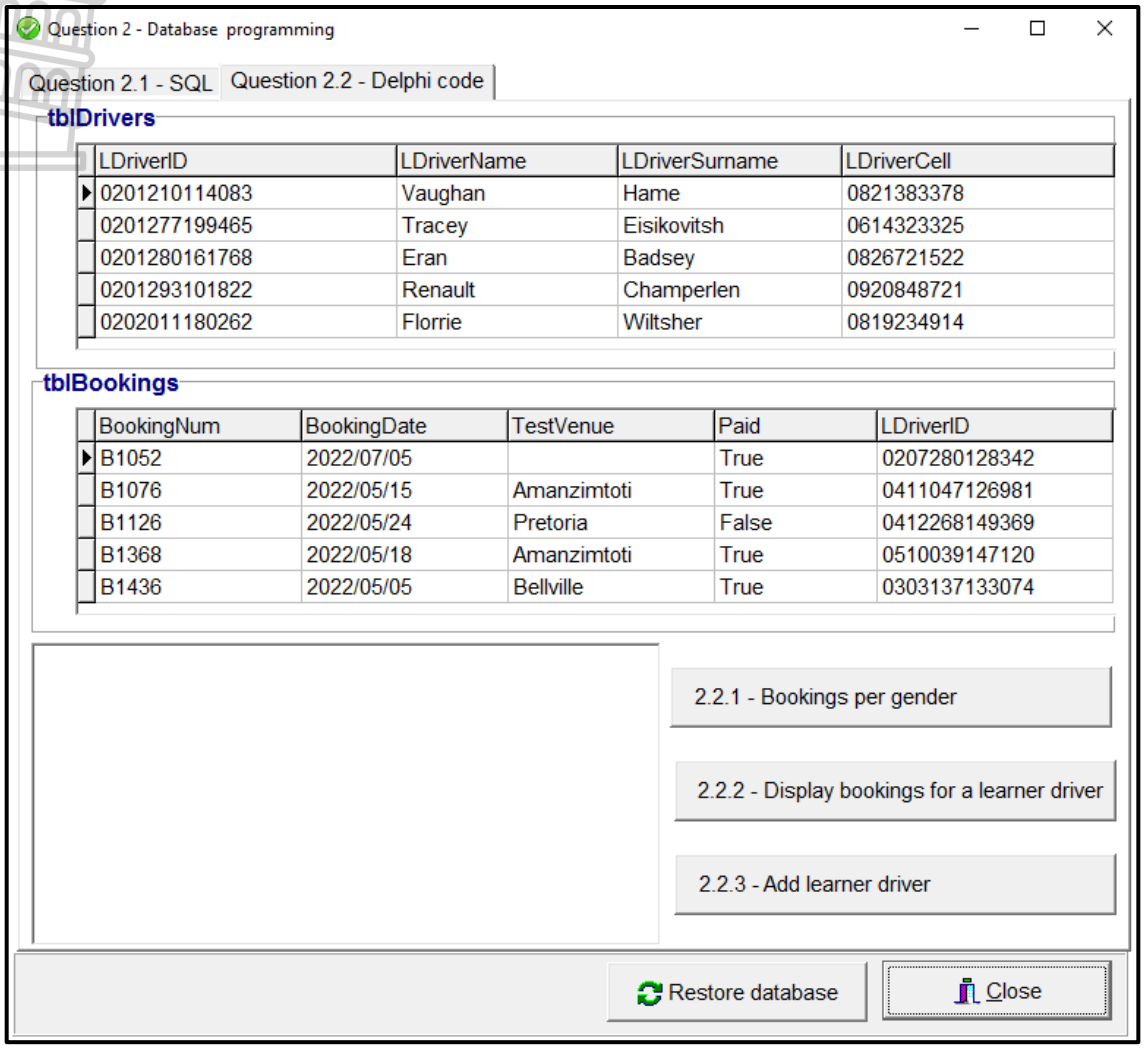
Example of output of the first six records in the table after the relevant records have been removed:

| BookingNum | BookingDate | TestVenue | Paid | LDriverID |
|---|---|---|---|---|
| B1052 | 2022/07/05 | Headquarters | True | 0207280128342 |
| B1076 | 2022/05/15 | Amanzimtoti | True | 0411047126981 |
| B1368 | 2022/05/18 | Amanzimtoti | True | 0510039147120 |
| B1436 | 2022/05/05 | Bellville | True | 0303137133074 |
| B1602 | 2022/05/25 | Bellville | True | 0211218127327 |
| B1729 | 2022/05/17 | Polokwane | True | 0311184102217 |

(5)

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

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




**NOTE:**

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

### 2.2.1 Button [2.2.1 - Bookings per gender]

Display the number of learner drivers according to gender.

The seventh digit of the ID number determines the gender:

- 0–4 indicates a female driver, e.g. 031223**3**628081
- 5–9 indicates a male driver, e.g. 010213**8**567087

**NOTE:** Code to initialise and display the number of male and female learner drivers have been provided.

Example of output:

```
Female: 206
Male: 145
```

(8)

2.2.2 **Button [2.2.2 - Display bookings for a learner driver]**

Code has been provided to enter a learner driver's ID number in an input box.

Write code to display all the bookings made for the learner driver with the ID number entered.

Example of output if 0207280128342 was entered as the learner driver's ID number:

```
B1052   2022/07/05
C1024   2022/05/07
T6518   2022/05/09
```

(7)

2.2.3 **Button [2.2.3 - Add learner driver]**

Write code to add a new learner driver to the **tblLDrivers** with the following details:

- ID number: 0405060708091
- Name: Trish
- Surname: Malope
- Cell phone number: 0710810911

Code has been provided to show the message 'Learner driver has been added.' (4)

- 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

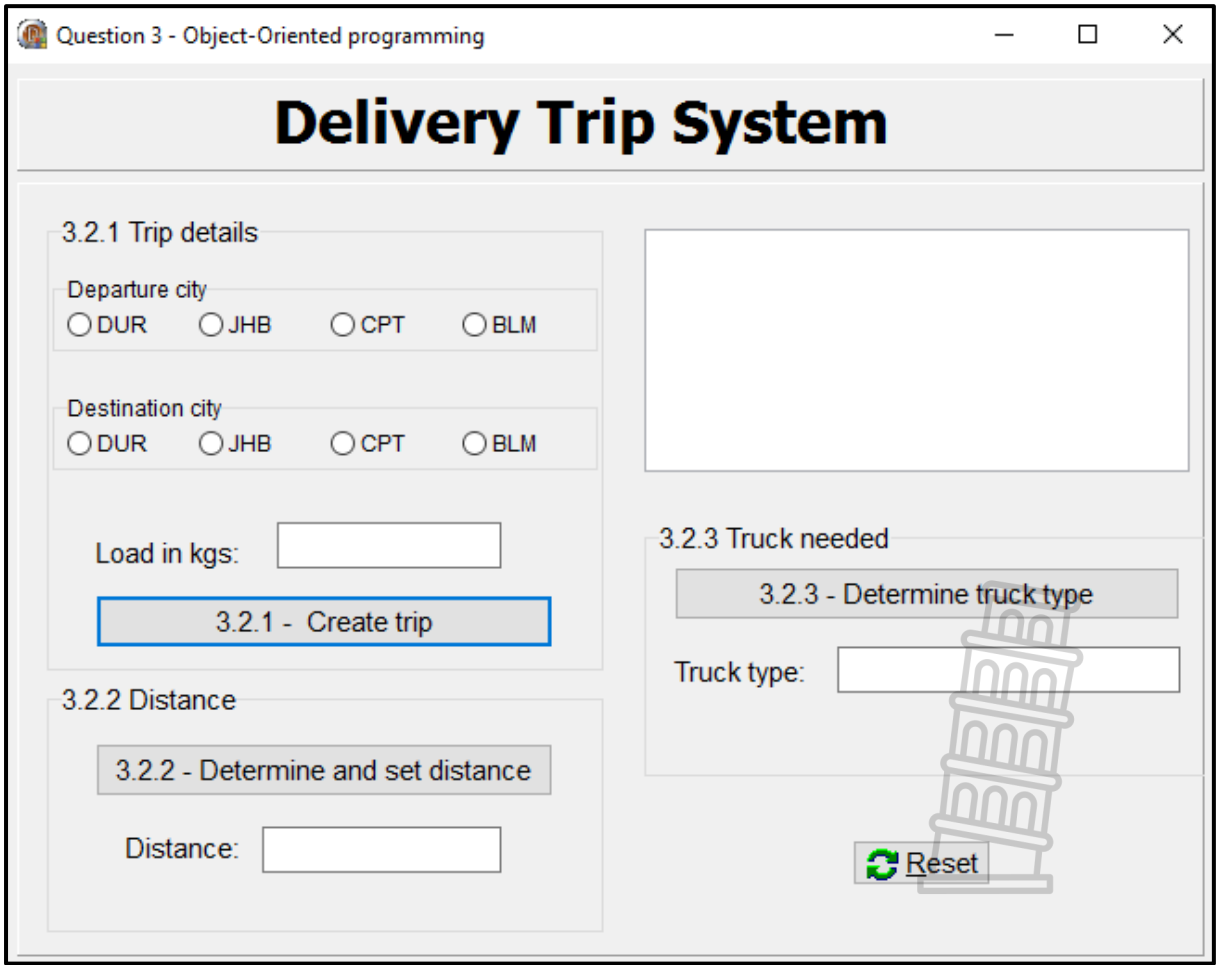
## QUESTION 3: OBJECT-ORIENTED PROGRAMMING

Delivery trips are arranged between different cities in South Africa. The program determines the type of the truck required for a delivery based on the weight of the load.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Open the incomplete object class **DeliveryTrip_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question3_U.pas** and the **DeliveryTrip_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.

3.1 The incomplete object class (**DeliveryTrip**) contains:

- The declaration of five attributes which describe a **Delivery trip** object
- The accessor methods **getDeparture** and **getDestination**
- The **toString** method
- An incomplete private auxiliary method **compileTripNum**

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

| Attribute | Description |
|---|---|
| fTripNumber | A unique number allocated to the trip |
| fDeparture | The city where the delivery trip starts |
| fDestination | The city where the trip ends |
| fLoad | The weight of the load to be delivered in kilograms |
| fDistance | The distance between the two cities |

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

3.1.1 Complete code for the private method called **compileTripNum** that will return a three-digit randomly generated number that does NOT end with a zero. (4)

3.1.2 Write code for a constructor method that will receive the departure city, destination city and the weight of the load to be transported as parameters.

- Assign the parameter values to the corresponding attributes.
- Call the private method **compileTripNum** to assign a value to the **fTripNumber** attribute.
- Set the distance attribute to zero. (5)

3.1.3 Write an accessor method called **getDistance** to return the attribute value for fDistance. (2)

3.1.4 Write code for a method called **setDistance** to receive the distance between the two cities as parameter and assign the value received to the distance attribute. (2)

3.1.5 Write code for a method called **determineTruckType** that will return a one-word description of the type of truck to be used to do the delivery – light, medium or heavy.

Use the **weight of the load** to identity the type of truck to be used, based on the following information:

| Weight of load in kg | Type of truck |
|---|---|
| Load <= 1000 | Light truck |
| Load > 1000 and <= 5000 | Medium truck |
| Load > 5000 | Heavy truck |

(8)

3.2 Write code to perform the tasks described in QUESTION 3.2.1 to QUESTION 3.2.3.

The program contains code for the object class to be accessible and the declaration of an object variable called **objTrip**.

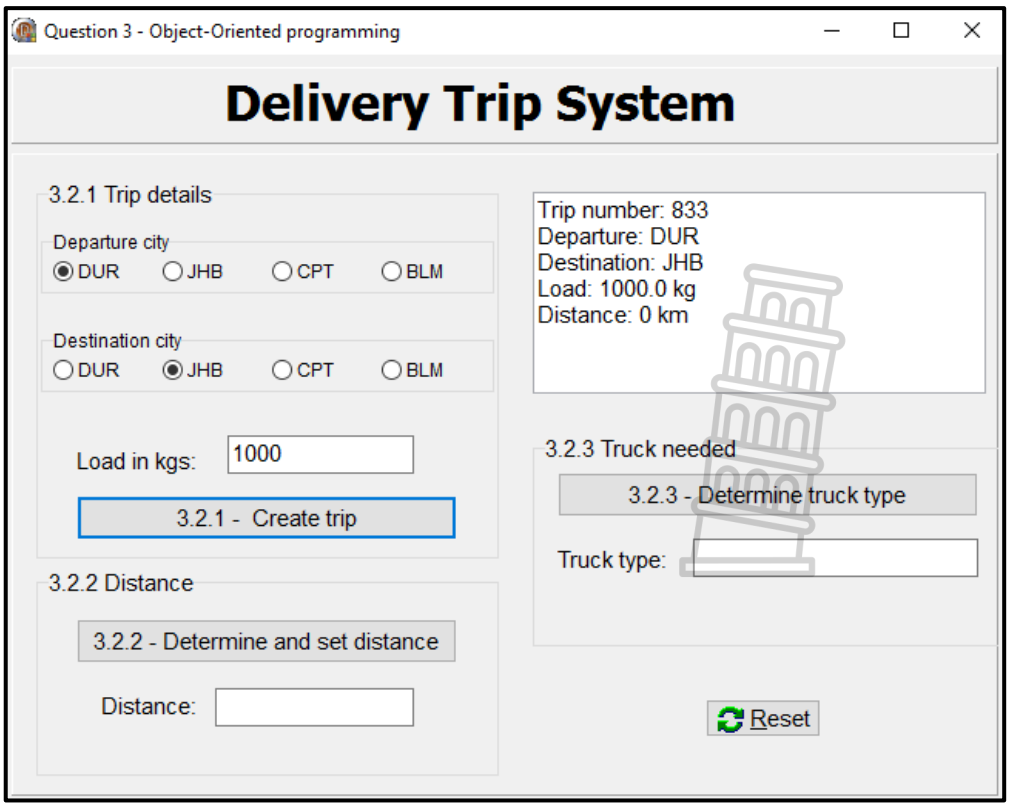
### 3.2.1 Button [3.2.1 - Create trip]

Write code to create the object and display information about the delivery between the two cities.

Write code to do the following:

- Instantiate the **objTrip** object using values obtained from the components:
  - Departure city from radio group **rgpQ3_2_1_Departure**
  - Destination city from radio group **rgpQ3_2_1_Destination**
  - Load in kilograms from the edit box **edtQ3_2_1**
- Display the object information in the output area **redQ3**.

Example of output if Durban (DUR) was selected as the point of departure and Johannesburg (JHB) as the destination and the weight of the load is entered as 1 000 kilograms:




(5)

3.2.2 **Button [3.2.2 - Determine and set distance]**

The information for each city of departure, destination and distance between the cities is stored in a delimited text file **DataQ3.txt**.

The format of each line of text in the text file is:

```
<Departure city>,<Destination city>#<distance between the cities>
```

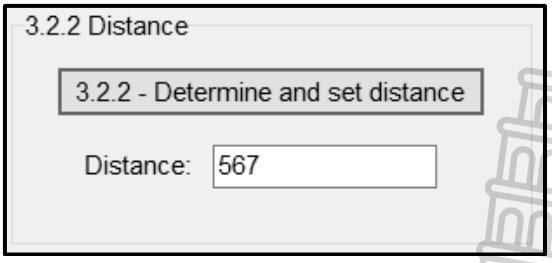
Example of the first five lines of text of the text file:

```
JHB,CPT#1398
JHB,BLM#398
JHB,DUR#567
CPT,JHB#1398
CPT,BLM#1004
```

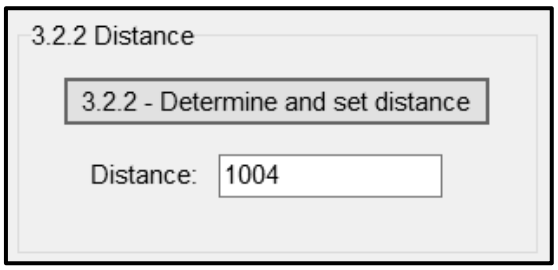
Write code to do the following:

- Extract the distance between the selected departure city and destination city from the text file **DataQ3.txt**.
- Set the distance attribute of the object to the distance extract from the text file.
- Display the distance in the edit box **edtQ3_2_2**.
- Use the toString method to display the updated information of the object in the rich edit **redQ3**.

Example of output in the edit box **edtQ3_2_2** if the departure city is DUR and the destination city is JHB:




Example of output in the edit box **edtQ3_2_2** if the departure city is Cape Town (CPT) and the destination city is Bloemfontein (BLM):



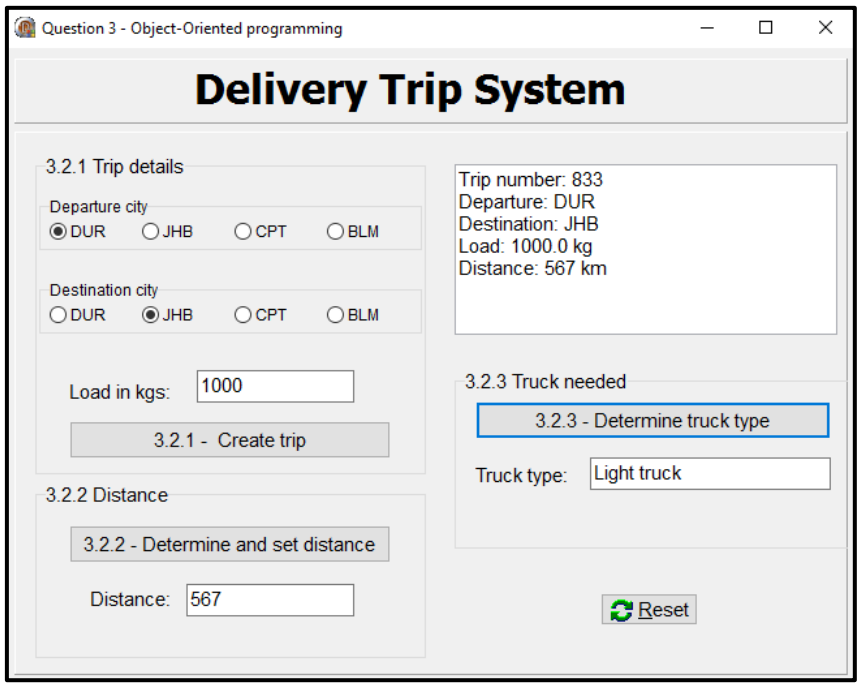

(12)

### 3.2.3 Button [3.2.3 - Determine truck type]

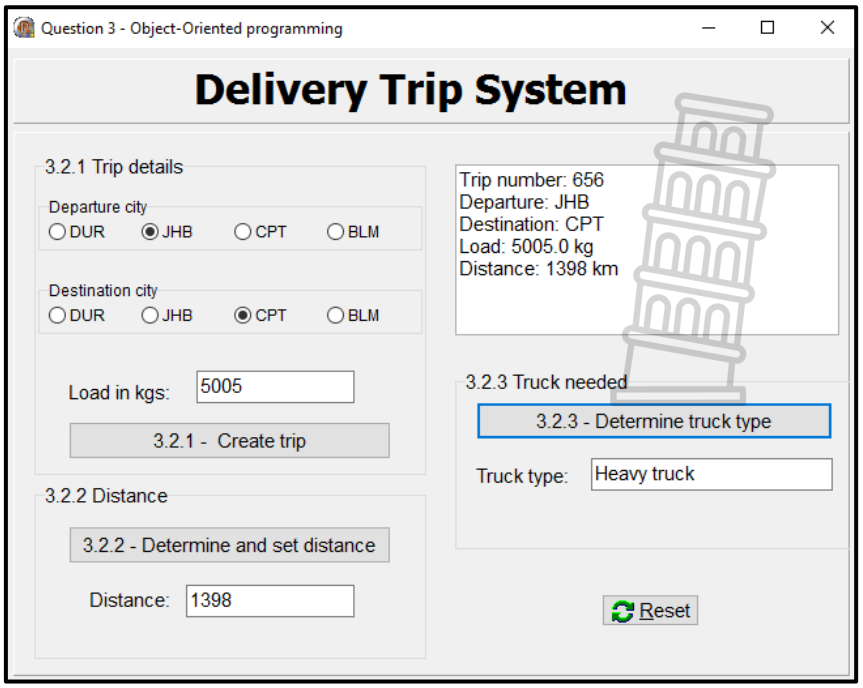
Write code to do the following:

- Call the relevant method to determine the type of truck that will be used.
- Display the result in the edit box **edtQ3_2_3**.

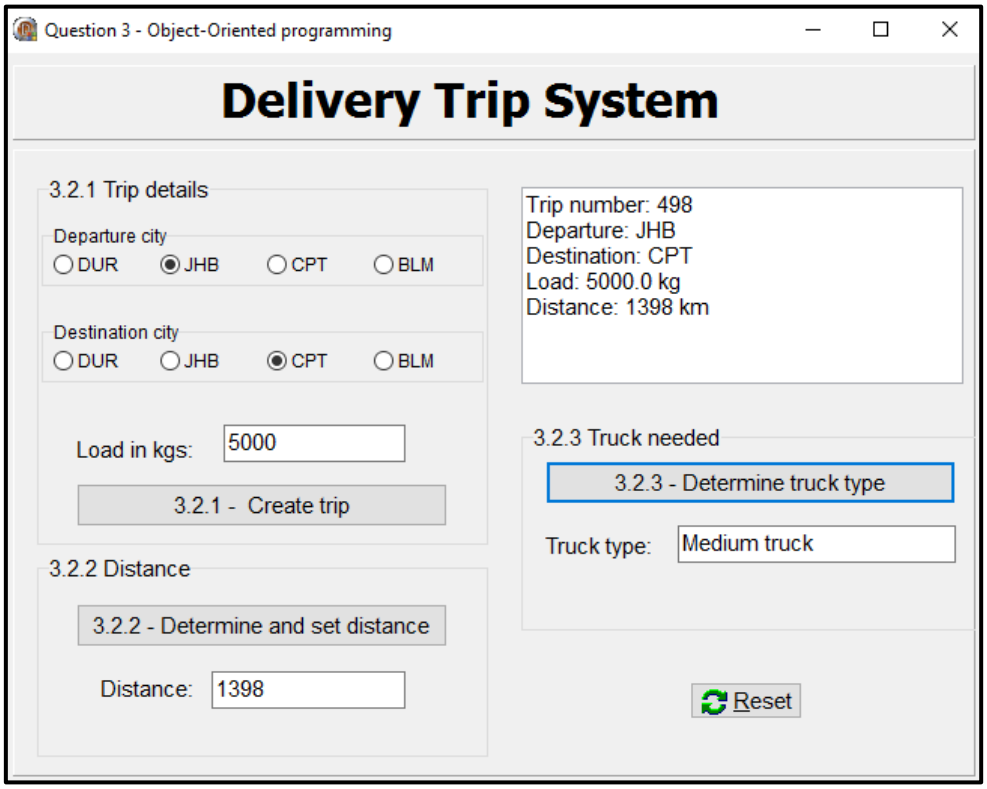
Example of output for a trip from DUR to JHB with a load of 1 000 kg:




Example of output for a trip from JHB to CPT with a load of 5 005 kg:



Example of output for a trip from JHB to CPT with a load of 5 000 kg:




(2)

- 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

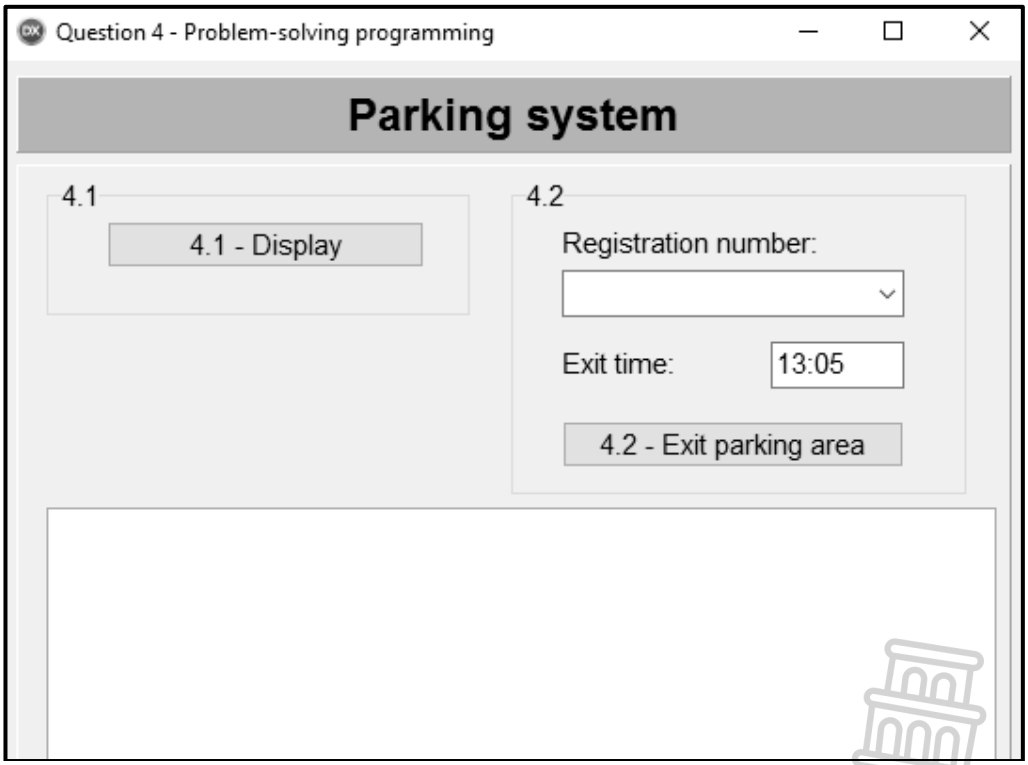
## QUESTION 4: PROBLEM-SOLVING PROGRAMMING

The Parking Group has requested your assistance for the implementation of the calculation of cost when vehicles exit a parking area.

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 graphical user interface (GUI):




The following have been provided in the program:

- Two parallel one-dimensional arrays, **arrRegNumbers** and **arrEntryTimes**, with the registration numbers and entry times of twenty vehicles that entered the parking area.

```
arrRegNumbers: array [1 .. 20] of String =
('CA 123 456', 'NN 21514', 'BBC 123 MP', 'BEC 123 EC', 'XRG
123 L', 'CA JN 912 WP', 'CD 083 027', 'CX 55472', 'BCD 123
MP', 'ND 122 156', '786 ZN', 'SNH 582 GP', 'IXLR8 NM', 'JJO
114 MP', 'OQE 329 GP', 'ALP 439 GP', 'CAA 220 002', 'YTF 871
EC', 'WIL 007 GP', 'CFA 1001');
arrEntryTimes: array [1 .. 20] of String =
   ('08:00', '09:22', '10:11', '10:15', '10:43', '11:03',
'11:34', '12:19', '12:32', '12:45', '12:59', '13:03',
'13:20', '14:24', '14:36', '15:41', '15:51', '16:06',
'16:38', '17:48');
```

- Code that populates the combo box **cmbQ4** with the registration numbers stored in the **arrRegNumbers** array.

Complete the code for each section of QUESTION 4 as described in QUESTION 4.1 to QUESTION 4.2.

4.1 **Button [4.1 - Display]**

Write code to display the number, registration number and time of entry for the vehicles stored in arrays **arrRegNumbers** and **arrEntryTimes** in the rich edit component **redQ4**.

Example of output:

| # | RegNumber | Time In |
|---|---|---|
| 1 | CA 123 456 | 08:00 |
| 2 | NN 21514 | 09:22 |
| 3 | BBC 123 MP | 10:11 |
| 4 | BEC 558 EC | 10:15 |
| 5 | XRG 123 L | 10:43 |
| 6 | CA JN 912 WP | 11:03 |
| 7 | CD 083 027 | 11:34 |
| 8 | CX 55472 | 12:19 |
| 9 | BCD 123 MP | 12:32 |
| 10 | ND 122 156 | 12:45 |
| 11 | 786 ZN | 12:59 |
| 12 | SNH 582 GP | 13:03 |
| 13 | IXLR8 NM | 13:20 |
| 14 | JJO 114 MP | 14:24 |
| 15 | OQE 329 GP | 14:36 |
| 16 | ALP 439 GP | 15:41 |
| 17 | CAA 220 002 | 15:51 |
| 18 | YTF 871 EC | 16:06 |
| 19 | WIL 007 GP | 16:38 |
| 20 | CFA 1001 | 17:48 |

(7)

4.2 **Button [4.2 - Exit parking area]**

The amount of time a vehicle spent in the parking area is used to determine the tariff per hour and calculate the cost of parking. The tariff for parking per hour is determined as follows:

| Time in parking area | Tariff |
|---|---|
| First 30 minutes | Free |
| 31 minutes to 2 hours | R50,00 per hour |
| More than 2 hours and less than or equal to 4 hours | R40,00 per hour |
| More than 4 hours | R30,00 per hour |

When a vehicle exits the parking area, the following must be done:

- Select the vehicle's registration number from the combo box **cmbQ4**.
- Enter the exit time of the vehicle in the edit box **edtQ4** in the format hh:mm.

  A message '**Invalid exit time**' must be displayed in the rich edit component **redQ4** when an attempt is made to enter an exit time that is earlier than the entry time for the vehicle selected. If the exit time is valid, processing can proceed.

- The cost of parking must be calculated using the following formula:

$$Cost\ of\ parking\ =\ Tariff * Time\ spent\ in\ parking\ area$$

  **NOTE:** Minutes must be rounded up to the next whole hour when calculating the cost.

- The registration number, entry time, exit time, time spent in the parking area, tariff per hour and cost of parking must be displayed in the **redQ4** rich edit component in the format shown in the example below:

  Example:

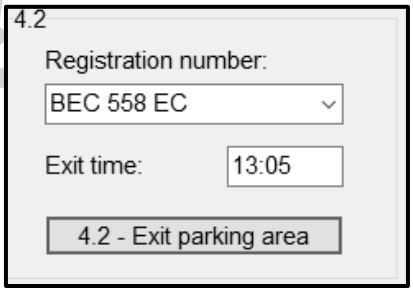
```
Registration number: CA 123 456
Entry time: 08:00
Exit time: 13:05
Time spent: 5 hours 5 min
Tariff per hour: R30.00
Cost of parking: R180.00
```

- The registration number of the selected vehicle and time of entry must be deleted from the relevant arrays.

  **HINT:** Click on the **Display** button to see if the information has been removed from the arrays correctly.

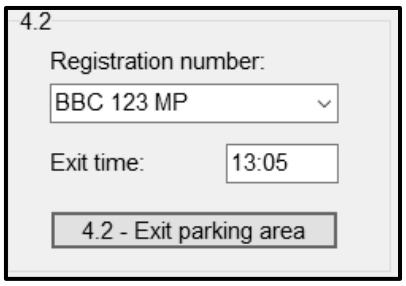
Examples of input and output:

Registration number: **BEC 558 EC**
Entry time: **10:15**
Exit time: **13:05**



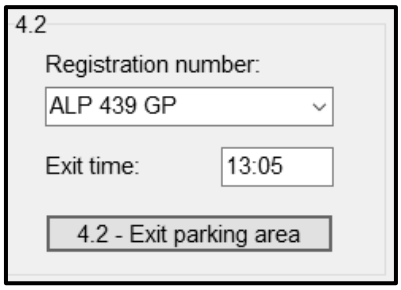

```
Registration number: BEC 558 EC
Entry time: 10:15
Exit time: 13:05
Time spent: 2 hours 50 min
Tariff per hour: R40.00
Cost of parking: R120.00
```

Registration number: **BCD 123 MP**
Entry time: **12:32**
Exit time: **13:05**




```
Registration number: BBC 123 MP
Entry time: 10:11
Exit time: 13:05
Time spent: 2 hours 54 min
Tariff per hour: R40.00
Cost of parking: R120.00
```

Registration number: **ALP 439 GP**
Entry time: **15:41**
Exit time: **13:05**




```
Invalid exit time
```

(23)

- 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 design of the database tables is as follows:

Table: **tblBookings**

This table contains the details of bookings for driver's licence tests at a driving school.

| Field name | Data type | Description |
|---|---|---|
| BookingNum | Short Text (5) | A unique booking number for each booking<br>The first character represents the type of license<br>B: Bicycle licence<br>C: Car licence<br>T: Truck licence |
| BookingDate | Date/Time | Date of driver's license test |
| TestVenue | Short Text (30) | The licence venue where the test will take place |
| Paid | Yes/No | Indicates whether the booking has been paid for or not |
| LDriverID | Short Text (13) | Driver's national ID number |

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

| BookingNum | BookingDate | TestVenue | Paid | LDriverID |
|---|---|---|---|---|
| B1052 | 2022/07/05 | | ☑ | 0207280128342 |
| B1076 | 2022/05/15 | Amanzimtoti | ☑ | 0411047126981 |
| B1126 | 2022/05/24 | Pretoria | ☐ | 0412268149369 |
| B1368 | 2022/05/18 | Amanzimtoti | ☑ | 0510039147120 |
| B1436 | 2022/05/05 | Bellville | ☑ | 0303137133074 |
| B1602 | 2022/05/25 | Bellville | ☑ | 0211218127327 |
| B1729 | 2022/05/17 | Polokwane | ☑ | 0311184102217 |
| B1742 | 2022/05/07 | | ☑ | 0211040181040 |
| B1776 | 2022/05/07 | Polokwane | ☑ | 0212293175654 |
| B1810 | 2022/05/15 | | ☑ | 0211191174956 |

Table: **tblLDrivers**

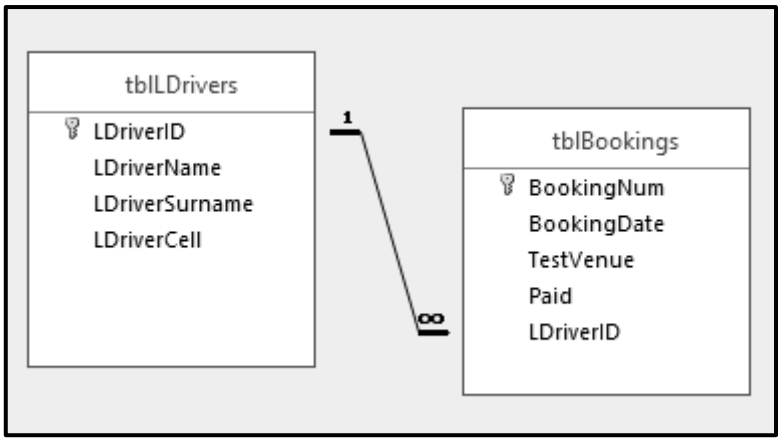
The table contains the information of learner drivers who booked driver's licence tests.

| Field name | Data type | Description |
|---|---|---|
| LDriverID | Short Text(13) | Learner driver's national ID number |
| LDriverName | Short Text(20) | Learner driver's name |
| LDriverSurname | Short Text(20) | Learner driver's surname |
| LDriverCell | Short Text(10) | Learner driver's cell number |

Example of the first ten records of the **tblLDrivers** table:

| LDriverID | LDriverName | LDriverSurname | LDriverCell |
|---|---|---|---|
| 0201210114083 | Vaughan | Hame | 0821383378 |
| 0201277199465 | Tracey | Eisikovitsh | 0614323325 |
| 0201280161768 | Eran | Badsey | 0826721522 |
| 0201293101822 | Renault | Champerlen | 0920848721 |
| 0202011180262 | Florrie | Wiltsher | 0819234914 |
| 0202032166446 | Desiree | Tiley | 0735251090 |
| 0202109119691 | Tanhya | Farans | 0939969302 |
| 0202147110371 | Austin | Vandrill | 0739028024 |
| 0203117133900 | Beau | Poleykett | 0824887784 |
| 0203220127964 | Winnah | Bon | 0829784704 |

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 EXAMINATIONS/
# NATIONAL SENIOR CERTIFICATE EXAMINATIONS

## INFORMATION TECHNOLOGY P1

## 2022

## 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 to 10) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 11 to 25) contain examples of solutions for Questions 1 to 4 in programming code.

- Copies of **Annexures A, B, C** and **D** (pages 3 to 10) 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 – Add device]**<br>Add the 'Tablet' item to rgpQ1_1 ✓<br>Change colour of rgpQ1_1 to cream ✓<br>Set item index to 0 ✓ | | **3** | |
| 1.2 | **Button [1.2 – Apply]** | | | |
| | 1.2.1 | Retrieve age from **edtQ1_2** ✓ and convert to integer ✓ | **2** | |
| | 1.2.2 | If (checkbox not ticked) ✓ then<br>Display suitable message using<br>ShowMessageDialog ✓ | **2** | |
| | 1.2.3 | if (age < 16) ✓<br>YearToApply = CURRENT_YEAR + (16 – iAge) ✓<br>Display suitable message using<br>ShowMessageDialog in first line of text ✓<br>Showing year to apply in next line of text ✓<br>converted value from int to string ✓ | **5** | |
| | 1.2.4 | if (checkbox is checked and age >= 16) ✓<br>Change the text of **btnQ1_2** to 'SUCCESSFUL' ✓<br><br>Note: Also accept **Else** with correct nested with both conditions | **2** | |

| | | | |
|---|---|---|---|
| 1.3 | **Button [1.3 – Fractions]**<br><br>Initialise all three variables ✓<br>Total = 0<br>Top = 1<br>Bottom = 1 // check that division by 0 is not possible – loose mark at increment<br>While loop ✓ condition: sum of terms <= 4 ✓<br>Term = Top ✓ / bottom ✓<br>Increment bottom ✓<br>Add term to Total ✓<br>Display total in **redQ1_3** converted to string ✓<br>and 4 decimal places ✓<br>Display number of terms in **redQ1_3** converted to string ✓<br><br>Alternative to while loop:<br>Repeat (1)<br>Total := Total (1) + (Top (1) / Bottom); (1)<br>Inc(Bottom); (1)<br>Until Total > 4; (1) | **10** | |

| | | | |
|---|---|---|---|
| 1.4.1 | **Button [1.4.1 – Count letter]**<br><br>Extract letter from **edtQ1_4_1** ✓ and convert sentence and letter to either uppercase/lowercase ✓<br><br>Initialise counter variable to 0 ✓<br><br>Loop from 1~~0~~ to Length of sentence ✓<br>Check if character in sentence ✓ = letter ✓<br>Increment counter ✓<br>Display the count on the panel ✓<br><br>Also accept alternative solutions. | **8** | |
| 1.4.2 | **Button [1.4.2 – Longest word]**<br><br>Add space at the end of sentence ✓<br>Initialise iLarge = 0 ✓ // any value < 1<br><br>Loop while position of space in sentence > 0 ✓<br>Get position of space and calculate length of word ✓<br>Test if length of word > iLarge ✓<br>Store length of this word in iLarge ✓<br>Delete characters in sentence up to space ✓<br><br>Display message indicating the length of the longest word ✓<br><br>Concepts:<br><br>Initialise variable for Longest to 1 or less //(1)<br>Loop through sentence //(1)<br>Identify individual words //(1) using spaces //(1)<br>Determine length of word //(1)<br>Test if length of word > Longest //(1)<br>Set Longest to length of word //(1)<br>Display the Longest //(1) | **8** | |
| | **TOTAL SECTION A:** | **40** | |

**ANNEXURE B**

**QUESTION 2: MARKING GRID – SQL AND DATABASE**

| **CENTRE NUMBER:** | **EXAMINATION NUMBER:** | | |
|---|---|---|---|
| **QUESTION** | **DESCRIPTION** | **MAX. MARKS** | **LEARNER'S MARKS** |
| 2.1 | **SQL statements** | | |
| 2.1.1 | **Button [2.1.1 – List of bookings]**<br><br>SELECT BookingNum, BookingDate ✓<br>FROM tblBookings ✓<br>ORDER BY BookingDate ✓ | **3** | |
| 2.1.2 | **Button [2.1.2 – Update test venue]**<br><br>UPDATE tblBookings ✓<br>SET ✓ TestVenue = "Headquarters" ✓<br>WHERE TestVenue IS NULL ✓ | **4** | |
| 2.1.3 | **Button [2.1.3 – Bellville bookings]**<br><br>SELECT BookingNum,LDriverName,LDriverSurname ✓<br>FROM tblBookings,tblLDrivers ✓<br>WHERE tblBookings.LDriverID = tblLDrivers.LDriverID ✓<br>AND tblBookings.TestVenue = "Bellville" ✓// or no table name<br><br>// also accept aliases | **4** | |
| 2.1.4 | **Button [2.1.4 – Licence types]**<br><br>SELECT LEFT(BookingNum,1) ✓<br>AS [LicenceTypes], ✓<br>count(BookingNum) ✓ AS [Number]<br>FROM tblBookings GROUP BY ✓<br>LEFT(BookingNum,1) ✓<br><br>Note:<br>Count can include any field from the tblBookings(including *) | **5** | |

| | | | |
|---|---|---|---|
| 2.1.5 | **Button [2.1.5 – Remove bookings]**<br><br>DELETE FROM tblBookings ✓<br>WHERE TestVenue = "Pretoria" ✓<br>AND ✓ BookingDate BETWEEN ✓<br>#2022/05/18# AND #2022/05/25# ✓<br><br>Alternative:<br>DELETE * FROM tblBookings (1)<br>WHERE TestVenue = "Pretoria" (1)<br>AND (1)<br>(Year(BookingDate) = 2002<br>AND Month(BookingDate) = 5)<br>AND (Day(BookingDate) >=18 (1)<br>AND Day(BookingDate) <=25)) (1) | **5** | |
| | **Subtotal:** | **21** | |

**QUESTION 2: MARKING GRID (CONT.)**

| | | | |
|---|---|---|---|
| 2.2 | **DATABASE MANIPULATION using Delphi code** | | |
| 2.2.1 | **Button - [2.2.1 – Bookings per gender]**<br><br>Go to first record in tblBookings✓<br>Loop until the end of the tblBookings table ✓<br>Test if 7th digit ✓ is less than 5 ✓<br>Increase the female variable ✓<br>Else ✓<br>Increase the male variable ✓<br>Move to the next record ✓<br><br>Display the number of males and females | **8** | |
| 2.2.2 | **Button - [2.2.2 – Display bookings for a learner driver]**<br><br>Go to the first record of tblBookings ✓<br>Loop until the end of the tblBookings table ✓<br>Test if the ID ✓ is same as input ID ✓<br>Display booking number ✓ and date (DateToStr) ✓<br>Move to the next record ✓ | **7** | |

| | | | |
|---|---|---|---|
| 2.2.3 | **Button - [2.2.3 – Add learner driver]**<br><br>tblLDrivers.Insert ✓<br><br>tblLDrivers['LDriverID'] := '0405060708091'<br>tblLDrivers['LDriverName'] := 'Trish'<br>tblLDrivers['LDriverSurname'] := 'Malope'<br>tblLDrivers['LDriverCell'] := '0710810911'<br><br>tblLDrivers.Post ✓<br><br>✓✓ - mark allocation<br>1 mark for any one field assigned correctly<br>1 mark for correctly assigning all other fields | **4** | |
| | **Subtotal:** | **19** | |

| | | | |
|---|---|---|---|
| | **TOTAL SECTION B:** | **40** | |

**ANNEXURE C**

**QUESTION 3: MARKING GRID – OBJECT-ORIENTED PROGRAMMING**

| **CENTRE NUMBER:** | **EXAMINATION NUMBER:** | | |
|---|---|---|---|
| **QUESTION** | **DESCRIPTION** | **MAX. MARKS** | **LEARNER'S MARKS** |
| 3.1.1 | **Function compileTripNum:**<br><br>Repeat ✓<br>Generate a random number ✓ of ~~four /~~ three digits ✓<br>Until the last digit is NOT zero ✓<br><br>Alternative:<br>sTripNum =<br>IntToStr(Random(10)) + // (1)<br>IntToStr(Random(10)) + // (1)<br>IntToStr(Random(9) + 1) // (1),last character is not a 0 // (1)<br><br>Also accept other alternatives | **4** | |
| 3.1.2 | **Constructor Create:**<br>Header with correct parameters✓ of the correct type ✓<br>Assign `fDeparture`, `fDestination` and `fLoad` to correct parameters ✓<br>`fTripNumber` = compileTripNum ✓<br>`fDistance` = 0 ✓ | **5** | |
| 3.1.3 | **Function getDistance:**<br><br>Function heading with correct Integer return type ✓<br>Result = fDistance ✓ | **2** | |
| 3.1.4 | **Procedure setDistance:**<br><br>Procedure heading with correct parameter ✓<br>fDistance = parameter value ✓ | **2** | |

| | | | |
|---|---|---|---|
| 3.1.5 | **Function determineTruckType:**<br><br>Function heading with correct string return type✓<br><br>Test if distance =0, ✓set result to 'No distance' ✓<br>Test if (fLoad <= 1000) ✓ then<br>sTruck = 'Light truck' ✓<br>else if fLoad <= 5000 then ✓<br>sTruck = 'Medium truck'✓<br>else ✓ sTruck = 'Heavy truck' ✓<br>Result = sTruck ✓<br><br>Alternative:<br>Function heading with correct string return type // (1)<br>`if fLoad <= 1000 then //(1)`<br>`sTruck = 'Light Truck' //(1)`<br>`if (fLoad > 1000)`<br>`AND (fLoad <= 5000) then //(1)`<br>`sTruck = 'Medium Truck' //(1)`<br>`if fLoad > 5000 then //(1)`<br>`sTruck = 'Heavy Truck' //(1)`<br>`result = sTruck (1)` | **8** | |
| | **Subtotal:** | **21** | |

**QUESTION 3: MARKING GRID (CONT.)**

| | | | |
|---|---|---|---|
| 3.2.1 | **Button [3.2.1 – Create trip]**<br><br>Extract values from components ✓<br>Instantiate the object with the provided values<br>objTrip:=<br>TDeliveryTrip.create ✓ (sDepartureCity ,sDestinationCity, rLoad)<br>Correct number of parameters ✓<br>Correct order of parameters ✓<br><br>Display the object details in the rich edit using the toString method ✓ | **5** | |
| 3.2.2 | **Button [3.2.2 – Determine and set distance]**<br><br>Assign and reset file ✓<br>Loop through the text file ✓<br>Read line from text file ✓<br><br>Extract the departure city from the text file ✓<br>Extract the destination city from the text file ✓<br><br>Test if the departure ✓ and destination cities ✓ of the text file is the same as the departure and destination cities in the object<br>Determine the position of the delimiter # ✓<br>Extract the distance from the text file ✓<br>Call the setDistance method with distance as the argument ✓<br>Display the distance between the cities in the edit box **edtQ3_2_2** ✓<br>Use the toString method to show the updated information in the rich edit ✓ | **12** | |
| | | | |
| 3.2.3 | **Button [3.2.3 – Determine truck type]**<br><br>Call the determineTruckType method ✓<br>Display the truck type in the edit box **edtQ3_2_3** ✓ | **2** | |
| | **Subtotal:** | **19** | |

| | | | |
|---|---|---|---|
| | **TOTAL SECTION C:** | **40** | |

**ANNEXURE D**

**QUESTION 4: MARKING GRID – PROBLEM-SOLVING PROGRAMMING**

| **CENTRE NUMBER:** | **EXAMINATION NUMBER:** | | |
|---|---|---|---|
| **SECTION** | **DESCRIPTION** | **MAX. MARKS** | **LEARNER'S MARKS** |
| 4.1 | **Button [4.1 - Display]**<br>Display Heading<br>Loop ✓from 1 to number of elements ✓<br>Display loop counter,✓ converted to string ✓,<br>arrRegNumbers[k] ✓and arrEntryTime [k] ✓<br>In neat columns ✓ (e.g. #9) | **7** | |
| 4.2 | **Button [4.2 – Exit parking area]**<br><br>Extract regNum from cmbQ4 ✓<br>Extract exit time from edtQ4 ✓<br><br>**Extract time from arrEntryTime**<br>Loop 1 to number of elements in array ✓<br>If arrRegNumbers[k] = regnum ✓<br>Get inTime from arrEntryTime ✓<br>Get index for specific registration number ✓<br><br>**Determine time spent in parking area**<br>Extract and convert HourIn to integer ✓<br>HourInMin = HourIn * 60 ✓<br>Extract and convert MinIn to integer ✓<br>MinutesIn = HourInMin + MinIn ✓<br><br>Extract and convert HourOut to integer<br>HourOutMin = HourOut * 60 ✓<br>Extract and convert MinOut to integer<br>MinutesOut = HourOutMin + MinOut<br><br>iTimeSpent = MinutesOut – MinutesIn ✓<br><br>if iTimeSpent >= 0✓ then //(valid)<br><br>**Determine tariff**<br>Case / if 0..30 : Tariff = 0<br>Case / if 31 .. 120 : Tariff = 50 ✓✓<br>Case / if 121 .. 240 : Tariff = 40<br>Case / if > 240 : Tariff = 30 | | |

| | | | |
|---|---|---|---|
| | **Determine total**<br>Cost = Tariff * Ceil(iTimeSpent / 60); ✓<br><br>**Move elements up in arrays**<br>Loop from index to length of arrRegNumbers -1 ✓<br>Move element up in arrRegNumbers ✓<br>Move element up in arrEntryTime ✓<br>Decrease counter<br><br>**Output**<br>Display RegNumber, Entry time and Exit time<br>Display hours (iTimeSpent div 60)<br>and minutes (iTimeSpent mod 60)<br>Display tariff in currency<br>Display total cost ✓✓<br><br>**Else**<br>Display Invalid Exit time ✓ | **23** | |
| | **TOTAL SECTION D:**<br>**GRAND TOTAL:** | **30**<br>**150** | |

**SUMMARY OF LEARNER'S MARKS:**

| | SECTION A | SECTION B | SECTION C | SECTION D | |
|---|---|---|---|---|---|
| | QUESTION 1 | QUESTION 2 | QUESTION 3 | QUESTION 4 | GRAND TOTAL |
| MAX. MARKS | 40 | 40 | 40 | 30 | 150 |
| LEARNER'S MARKS | | | | | |

## ANNEXURE E: SOLUTION FOR QUESTION 1

```
// ========================================================================
//                    Question 1.1 – 3 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
    rgpQ1_1.Items.Add('Tablet');
    rgpQ1_1.ItemIndex := 0;
    rgpQ1_1.Color := clCream;
end;
// ========================================================================
//                    Question 1.2 – 11 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
const
  CURRENT_YEAR = 2022; // provided code
var
  iAge, iAppYear: integer;
begin
  iAge := StrToInt(edtQ1_2.Text);
  if (NOT(ckbSACitizen.checked)) then
      ShowMessage('The applicant must be a South African citizen.')
  else
  if iAge < 16 then
  begin
    iAppYear := CURRENT_YEAR + (16 - iAge);
    ShowMessage('The applicant is too young.' +  #13 +
                'Can apply in the year ' + IntToStr(iAppYear));
  end
  else
     btnQ1_2.Caption := 'SUCCESSFUL';
end;
// ========================================================================
//                    Question 1.3 – 10 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  iBottom, iTop : integer;
  rTerm, rTotal : real;
begin
  iBottom := 0;
  iTop := 1;
  rTotal := 0;
  while rTotal <= 4 do
     begin
     inc(iBottom);
     rTerm := iTop / iBottom;
     rTotal := rTotal + rTerm;
     end;
  redQ1_3.Lines.Add('Amount of terms: ' + IntToStr(iBottom));
  redQ1_3.Lines.Add('Total: ' + FloatToStrF(rTotal, ffFixed, 10, 4));
end;
```

```
// ========================================================================
                      Question 1.4.1 – 8 marks
// ========================================================================
procedure TfrmQuestion1.btn1_4_1Click(Sender: TObject);
var
  sSentence : String ;
  iCount, k  : integer ;
  cLetter : char;
begin
  sSentence := UpperCase(edtQ1_4.Text);
  cLetter := Upcase(edtQ1_4_1.Text[1]);
  iCount := 0;
  for k := 0 to Length(sSentence) do
  begin
      if (sSentence[k] = cLetter) then
         iCount := iCount + 1;
  end;
  pnlQ1_4_1.Caption := 'Number: ' + IntToStr(iCount);
end;

// ========================================================================
                      Question 1.4.2 – 8 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_4_2Click(Sender: TObject);
var
  sSentence : String ;
  iLarge, k, iLength : integer ;
begin
  sSentence := edtQ1_4.Text + ' ';
  iLarge := 0;
  while pos(' ', sSentence) > 0 do
   begin
    iLength := pos(' ',sSentence)- 1;
    if iLength > iLarge then
      iLarge := iLength;

     Delete(sSentence,1,iLength + 1);
   end;
   pnlQ1_4_2.Caption := 'Length of longest word: ' + IntToStr(iLarge);

 { //Alternative
  sSentence := edtQ1_4.Text + ' ';

  sLargest := '';
  iStart := 1;

  for k := 1 to Length(sSentence) do
    begin
      if sSentence[k] = ' ' then
        begin
          iEnd := k;
          sWord := Copy(sSentence, iStart, iEnd - iStart);
          if iLarge < Length(sWord) then
              iLarge := Length(sWord);
          iStart := iEnd + 1 ;
```

```
        end;
    end;
  redQ1_4_2.Lines.Add('Length of longest word: ' + IntToStr(iLarge));
end;    }
end.
```

## ANNEXURE F: SOLUTION FOR QUESTION 2

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

  // --- Global variables provided ---
  tblLDrivers, tblBookings: TADOTable;
  qryDB: TADOQuery;

implementation

{$R *.dfm}
{$R+}

// Question 2.1 - SQL section

// ============================================================================
//                    Question 2.1.1 – 3 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
  sSQL1 := 'SELECT BookingNum, BookingDate FROM tblBookings ' +
  ' ORDER BY BookingDate';

  // Provided code - do not change
  dbCONN.RunSQL(sSQL1, dbgSQL);
end;

// ============================================================================
//                    Question 2.1.2 – 4 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
  bChange: Boolean;
begin
  sSQL2 := 'UPDATE tblBookings SET TestVenue = "Headquarters" WHERE
            TestVenue IS NULL';

   // Provided code - do not change
  dbCONN.ExecuteSQL(sSQL2, bChange);
  if bChange then
    begin
      MessageDlg('Database updated', mtInformation, [mbOK], 0);
    end;
end;
```

```
// ============================================================================
//                      Question 2.1.3 – 4 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3: String;
begin
  sSQL3 := 'SELECT BookingNum,LDriverName,LDriverSurname FROM
           tblBookings,tblLDrivers WHERE tblBookings.LDriverID =
           tblLDrivers.LDriverID AND TestVenue = "Bellville"';

    // Provided code - do not change
  dbCONN.RunSQL(sSQL3,dbgSQL);
end;
// ============================================================================
//                      Question 2.1.4 – 5 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
begin
  sSQL4 := 'SELECT LEFT(BookingNum,1) AS [LicenceTypes]
            Count(BookingNum) AS [Number] FROM tblBookings
            GROUP BY LEFT(BookingNum,1)';

    // Provided code - do not change
     dbCONN.RunSQL(sSQL4,dbgSQL);

end;

// ============================================================================
//                      Question 2.1.5 – 5 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5: String;
  bChange: Boolean;
begin
  sSQL5 := DELETE FROM tblBookings WHERE TestVenue = "Pretoria" AND
           BookingDate BETWEEN #2022/05/18# AND #2022/05/25# ';

// Provided code - do not change
  if dbCONN.ExecuteSQL(sSQL5, bChange) then
    begin
      MessageDlg('Database updated', mtInformation, [mbOK], 0);
    end;
end;
```

```
// Question 2.2 - Delphi section

// ========================================================================
//                    Question 2.2.1 – 8 marks
// ========================================================================
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  iMaleCount, iFemaleCount : Integer;
begin
//Provided code
  iMaleCount := 0;
  iFemaleCount := 0;

//Question 2.2.1
  tblBookings.First;
  while not tblBookings.Eof do
    begin
      if COPY(tblBookings['LDriverID'],7,1) IN ['0'.. '4'] then
           Inc(iFemaleCount)
      else
           Inc(iMaleCount);

      tblBookings.Next;
    end;

//Provided code
 redQ2.Lines.Add('Female: ' + IntToStr(iFemaleCount));
 redQ2.Lines.Add('Male: ' + IntToStr(iMaleCount));
end;

// ========================================================================
//                    Question 2.2.2 – 7 marks
// ========================================================================
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  sID : String;
  bFound : Boolean;
  sOut : String;
begin
//Provided code
redQ2.clear;
tblLDrivers.First;
sID := InputBox('Enter learner driver`s ID','','0207280128342');

 //Question 2.2.2

tblBookings.First;
while not tblBookings.eof do
   begin
    if tblBookings['LDriverID'] = sID then
      redQ2.Lines.Add(tblBookings['BookingNum'] +#9 +
                DateToStr(tblBookings['BookingDate']));
    tblBookings.Next;
   end;
end;
// ========================================================================
```

```
//                    Question 2.2.3 – 4 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_2_3Click(Sender: TObject);
begin
//Question 2.2.3
  tblLDrivers.Insert;
  tblLDrivers['LDriverID']:= '0405060708091';
  tblLDrivers['LDriverName'] :='Trish';
  tblLDrivers['LDriverSurname'] := 'Malope';
  tblLDrivers['LDriverCell']  :='0710810911';
  tblLDrivers.Post;

//Provided code
  ShowMessage('Learner driver has been added.');
end;

{$REGION DB CONNECTION}
//Setup DB connections - DO NOT CHANGE!

// ============================================================================
procedure TfrmQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  // Restores the Database
  dbCONN.RestoreDatabase;
  dbCONN.setupGrids(dbgBookings, dbgSQL);
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
  //Format rich edit
  redQ2.Paragraph.TabCount := 2;
  redQ2.Paragraph.Tab[0] := 70;
  redQ2.Paragraph.Tab[1] := 100;
  // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblLDrivers := dbCONN.tblOne;
  tblBookings := dbCONN.tblMany;
  dbCONN.SetupGrids(dbgBookings, dbgSQL);
  pgcTabs.ActivePageIndex := 0;

end;
// ============================================================================

{$ENDREGION}
end.
```

## ANNEXURE G: SOLUTION FOR QUESTION 3

### Object class

```
// ========================================================================
//                    Question 3.1.1 – 4 marks
// ========================================================================
function TDeliveryTrip.compileTripNum: Integer;
var
  iTripNum : Integer;
begin
 //Question 3.1.1
  repeat
    iTripNum := randomRange(100,1000);
  until iTripNum MOD 10 <> 0;
  result := iTripNum;
end;
// ========================================================================
//                    Question 3.1.2 – 5 marks
// ========================================================================
constructor TDeliveryTrip.create(sDeparture, sDestination:String;
rLoad:Real);
begin
    fDeparture := sDeparture;
    fDestination:= sDestination;
    fLoad := rLoad;
    fDistance := 0;
    fTripNumber := compileTripNum;
end;
// ========================================================================
//                    Question 3.1.3 – 2 marks
// ========================================================================
function TDeliveryTrip.getDistance: integer;
begin
    result := fDistance;
end;
// ========================================================================
//                    Question 3.1.4 – 2 marks
// ========================================================================
procedure TDeliveryTrip.setDistance(iDist: integer);
begin
   fDistance := iDist;
end;
// ========================================================================
//                    Question 3.1.5 – 8 marks
// ========================================================================
function TDeliveryTrip.determineTruckType: string;
begin
   if fDistance = 0 then
      result := 'No distance'
     else
         if (fLoad <= 1000) then
           result := 'Light truck'
          else if fLoad <= 5000 then
                  result := 'Medium truck'
                 else result := 'Heavy truck';
end;
```

```
// ============================================================================
//                      Provided Code
// ============================================================================

function TDeliveryTrip.getDestination: String;
begin
  Result := fDestination;
end;

function TDeliveryTrip.getDeparture: String;
begin
  Result := fDeparture;
end;

function TDeliveryTrip.toString: String;
begin
 Result :='Trip number: '+ IntToStr(fTripNumber) + #13 +
          'Departure: ' + fDeparture + #13 +
          'Destination: '+ fDestination + #13 +
          'Load: '+ FloatToStrF(fLoad,ffFixed,5,1) +' kg'+ #13 +
          'Distance: ' + IntToStr(fDistance) + ' km' + #11+ '';
end;

//end of provided code

end.
// ============================================================================
```

## Main Form Unit

```
// ========================================================================
//                    Question 3.2.1 – 5 marks
// ========================================================================
procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
    sDepart, sDestination : String;
    rLoad: real;

begin
 //Provided code
  redQ3.Clear;

 //Code Question 3.1 here
  sDepart := rgpQ3_2_1_Departure.Items[rgpQ3_2_1_Departure.ItemIndex];
  sDestination :=
      rgpQ3_2_1_Destination.Items[rgpQ3_2_1_Destination.ItemIndex];
  rLoad := StrToFloat(edtQ3_2_1.Text);

  objTrip:= TDeliveryTrip.create(sDepart, sDestination, rLoad);
  redQ3.Lines.Add(objTrip.toString);
end;
// ========================================================================
//                    Question 3.2.2 – 12 marks
// ========================================================================
procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  sLine, sTown, sDep, sDest : String;
  iPos : Integer;
  iDistance: integer;
  myFile : TextFile;

begin
 //Provided code
  redQ3.Clear;
  if FileExists('DataQ3.txt') <> True then
    begin
      ShowMessage('File Does not Exist');
      Exit;
    end
  else

  //Code Question 3.2 here
  AssignFile(myFile,'DataQ3.txt');
  Reset(myFile);
  while not EOF(myFile) do
  begin
    readln(myFile,sLine);
    sDep := objTrip.getDeparture;
    sDest := objTrip.getDestination;
    if (Pos(sDest, sLine) > 0) AND (Pos(sDep, sLine) > 0) then
        begin
          iPos := pos('#',sLine);

          objTrip.setDistance(StrToInt(copy(sLine, iPos + 1)));
```

```
            edtQ3_2_2.Text := IntToStr(objTrip.getDistance);
        end;
    end;
    CloseFile(myFile);
    redQ3.Lines.Add(objTrip.toString);
end;

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

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
begin
    //Code Question 3.2.3 here
    edtQ3_2_3.Text := objTrip.determineTruckType;
end;

end.
```

## ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_u;

interface

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

type
  TfrmQuestion4 = class(TForm)
    pnlQ4: TPanel;
    Panel2: TPanel;
    redQ4: TRichEdit;
    edtQ4: TEdit;
    cmbQ4: TComboBox;
    lblQ4_2_2: TLabel;
    grbQ4_2: TGroupBox;
    lblQ4_2_1: TLabel;
    btnQ4_1: TButton;
    GroupBox2: TGroupBox;
    btnQ4_2: TButton;
    procedure btnQ4_1Click(Sender: TObject);
    procedure cmbQ4Enter(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion4: TfrmQuestion4;


  arrRegNumbers: array [1 .. 20] of String = (
    'CA 123 456','NN 21514',
    'BBC 123 MP','BEC 558 EC',
    'XRG 123 L', 'CA JN 912 WP',
    'CD 083 027','CX 55472',
    'BCD 123 MP','ND 122 156',
    '786 ZN',    'SNH 582 GP',
    'IXLR8 NM',  'JJO 114 MP',
    'OQE 329 GP','ALP 439 GP',
    'CAA 220 002','YTF 871 EC',
    'WIL 007 GP','CFA 1001'
  );

  arrEntryTimes: array [1 .. 20] of String = (
    '08:00',  '09:22',
    '10:11',  '10:15',
    '10:43',  '11:03',
    '11:34',  '12:19',
    '12:32',  '12:45',
```

```
    '12:59',  '13:03',
   '13:20', '14:24',
    '14:36',  '15:41',
    '15:51',  '16:06',
    '16:38',  '17:48'
  );

iCounter:Integer;
implementation

{$R *.dfm}
// ============================================================================
//                      Question 4.1 – 7 Marks
// ============================================================================

procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  sLine: String;
  I: integer;
  J: integer;
begin
//Provided code
 redQ4.Clear;
 redQ4.Lines.Add('#'+ #9 + 'RegNumber' + #9 + 'Time In');

// Code Question 4.1 here
 for I:= 1 to iCounter do
   redQ4.Lines.Add(IntToStr(I) + #9 + arrRegNumbers[I] + #9+
arrEntryTimes[I]);

 {Alternative
 redQ4.Lines.Add(format('%-5s%-15s%10s', ['#', 'Number Plate', 'Time
In']));
 for I := 1 to iCounter do
   redQ4.Lines.Add(format('%-5d%-15s%10s',
   [I, arrNumberPlates[I], arrEntryTimes[I]])); }
end;

// ============================================================================
//                      Question 4.2 – 23 Marks
// ============================================================================

procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  sLine, sRegNum, sTimeIn, sTimeOut: String;
  iIndex, iTimeInInMins, iTimeOutInMins, iTimeSpent: integer;
  iHoursMin, iMinutes, iPosColon : Integer;
  I, iCounter: integer;
  rTariff, rCost:real;
begin
 redQ4.Clear;
 for I := 1 to iCounter do
  if cmbQ4.Text = arrRegNumbers[I] then
    begin
      iIndex := I;
      sRegNum := arrRegNumbers[I];
      sTimeIn := arrEntryTime[I];
    end;
```

```
    sTimeOut := edtQ4.Text;
    iPosColon := pos(':',sTimeOut);
    iHoursMin := StrToInt(copy(sTimeOut, 1, 2)) * 60;
    iMinutes := StrToInt(copy(sTimeOut, iPosColon + 1, 2));
    iTimeOutInMins := iHoursMin + iMinutes;

    iPosColon := pos(':',sTimeIn);
    iHoursMin := StrToInt(copy(sTimeIn, 1, 2)) * 60;
    iMinutes := StrToInt(copy(sTimeIn, iPosColon + 1, 2));
    iTimeInInMins := iHoursMin + iMinutes;

    iTimeSpent := iTimeOutInMins - iTimeInInMins;

    if iTimeSpent > 0 then
     begin

     case iTimeSpent of
      0 .. 30:    rTariff := 0;
      31 .. 120:  rTariff := 50;
      121 .. 240: rTariff := 40;
     else
      rTariff := 30;
   end;

 rCost := rTariff * Ceil(iTimeSpent / 60);

 for I := iIndex to length(arrRegNumbers) - 1 do
  begin
    arrRegNumbers[I] := arrRegNumbers[I + 1];
    arrEntryTime[I] := arrEntryTime[I + 1];
  end;
Dec(iCounter);

 redQ4.Lines.Add('Registration number: ' + sRegNum);
 redQ4.Lines.Add('Entry time: ' + sTimeIn);
 redQ4.Lines.Add('Exit time: ' + sTimeOut);
 redQ4.Lines.Add('Time spent: ' + IntToStr(iTimeSpent div 60) + ' hours ' +
                              IntToStr(iTimeSpent mod 60) + ' min');
 redQ4.Lines.Add('Tariff per hour: ' +
                 floatToStrF(rTariff, ffCurrency, 10, 2));
 redQ4.Lines.Add('Cost of parking: ' +
                 floatToStrF(rCost, ffCurrency, 10, 2));
end
else
    redQ4.Lines.Add('Invalid exit time');
end;

//PROVIDED CODE - DO NOT CHANGE
procedure TfrmQuestion4.cmbQ4Enter(Sender: TObject);
var
  I: integer;
begin
  cmbQ4.Clear;
  for I := 1 to length(arrRegNumbers) do
  begin
    cmbQ4.Items.Add(arrRegNumbers[I]);
  end;

end;
```

```
procedure TfrmQuestion4.FormActivate(Sender: TObject);
begin
  redQ4.Paragraph.TabCount:= 2;
  redQ4.Paragraph.Tab[0] := 50;
  redQ4.Paragraph.Tab[1] := 150;
end;

end.
```