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):
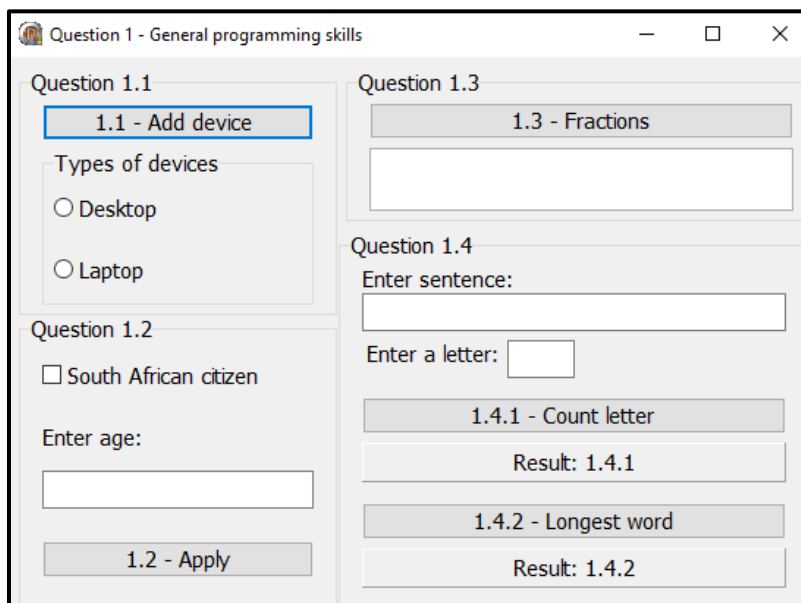
- Complete the code for each section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.4 that follow.
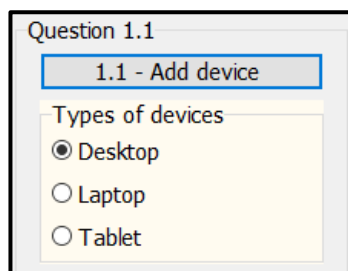
1.1   **Button [1.1 – Add device]**

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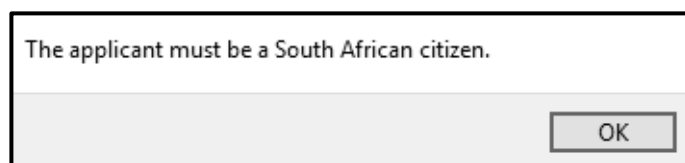
Example of output if the check box for South African citizen is ticked and the value of 13 is entered:

```
The applicant is too young.
Can apply in the year 2025

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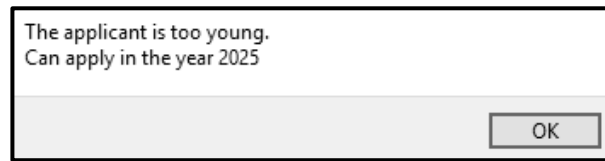
Example of output if the check box for South African citizen is ticked and the value of 16 is entered:

```
Question 1.2
☑ South African citizen

Enter age:
16

            SUCCESSFUL
```
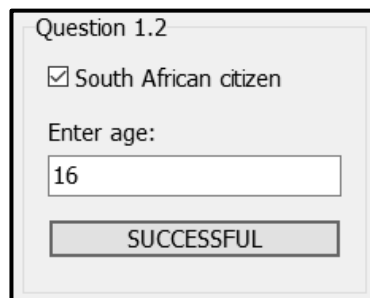
1.3   **Button [1.3 – Fractions]**

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:

```
Question 1.3
              1.3 - Fractions

Total: 4.0272
Number of terms: 31
```
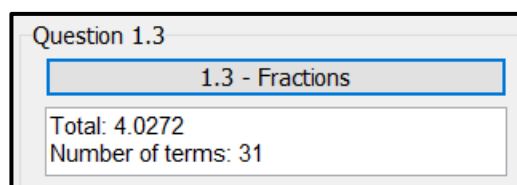
(10)

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:

```
Question 1.4
Enter sentence:
I am very good at IT
Enter a letter: i
        1.4.1 - Count letter
        Number: 2
```

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

```
Question 1.4
Enter sentence:
I am very good at programming
Enter a letter: a
        1.4.1 - Count letter
        Number: 3
```
(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:

```
Question 1.4
Enter sentence:
I am very good at IT

Enter a letter: i

          1.4.1 - Count letter
               Number: 2

          1.4.2 - Longest word
         Length of longest word: 4
```

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

```
Question 1.4
Enter sentence:
I am very good at programming
Enter a letter:  a

          1.4.1 Count letter
               Number: 3

          1.4.2 Longest word
        Length of longest word: 11
```
(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.
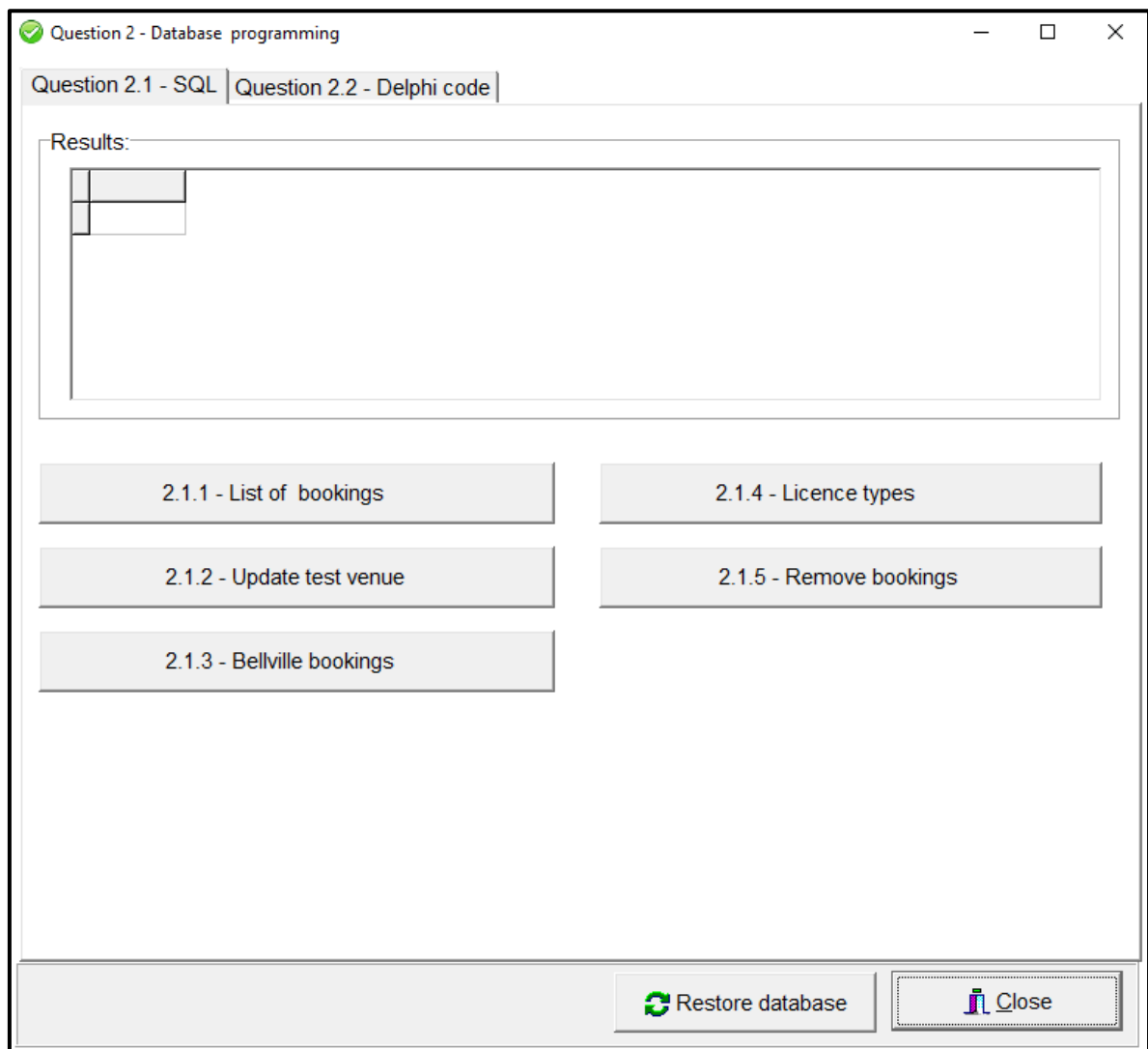
NOTE:

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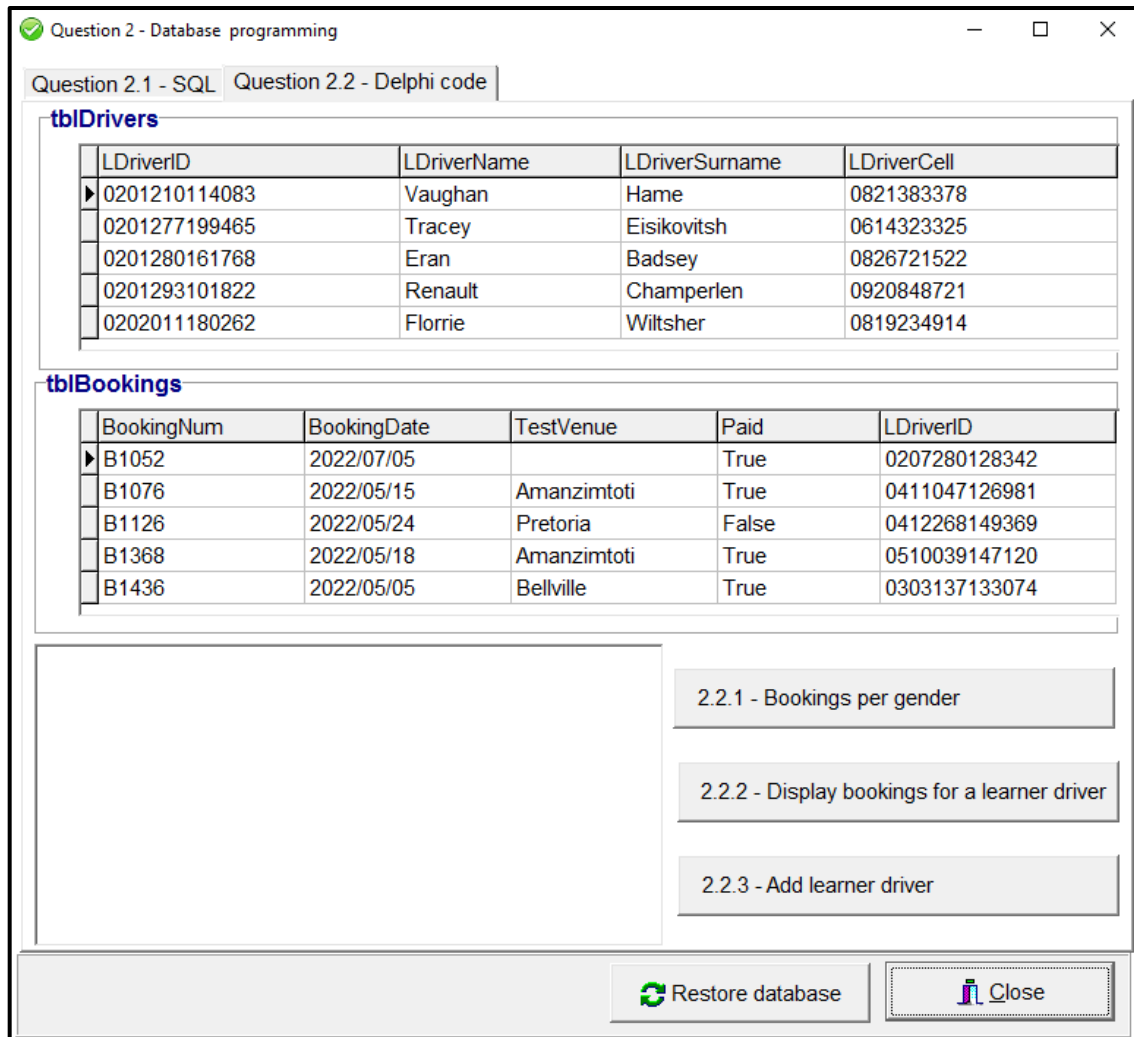
NOTE:
- Use ONLY Delphi programming code to answer QUESTION 2.2.
- NO marks will be awarded for SQL statements in QUESTION 2.2.

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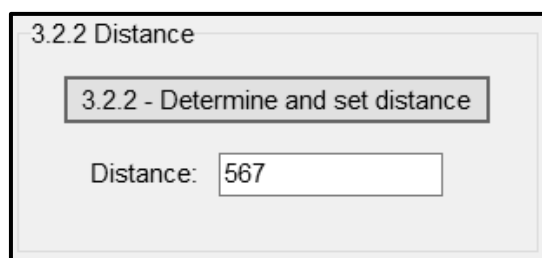
Example of the first five lines of text of the text file:

```
JHB,CPT#1398
JHB,BLM#398
JHB,DUR#567
CPT,JHB#1398
CPT,BLM#1004
```
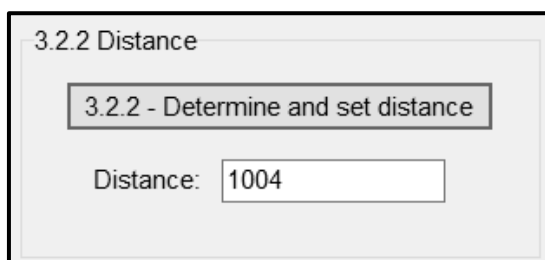
Write code to do the following:

- 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:

```
3.2.2 Distance
    ┌──────────────────────────────────┐
    │  3.2.2 - Determine and set distance │
    └──────────────────────────────────┘
    Distance:  567
```

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):

```
3.2.2 Distance
    ┌──────────────────────────────────┐
    │  3.2.2 - Determine and set distance │
    └──────────────────────────────────┘
    Distance:  1004
```

(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):
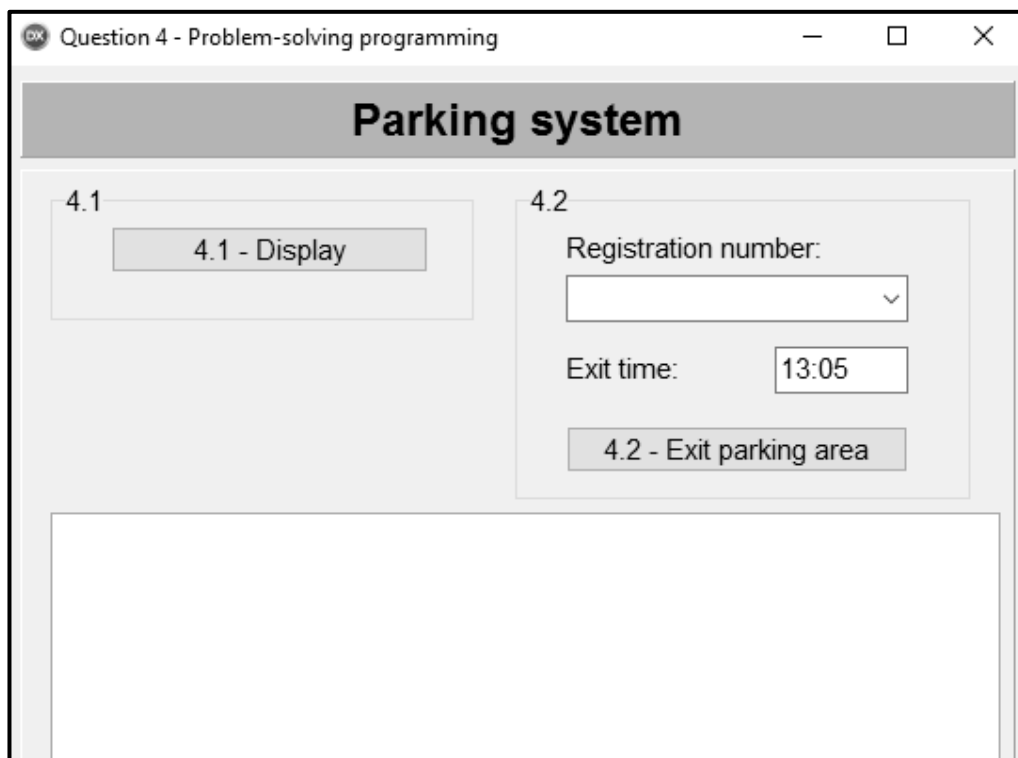
The following have been provided in the program:

- 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**

```
4.2
Registration number:
BEC 558 EC        ∨

Exit time:      13:05

  4.2 - Exit parking area
```

```
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**

```
4.2
Registration number:
BBC 123 MP        ∨

Exit time:      13:05

  4.2 - Exit parking area
```

```
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**

```
4.2
Registration number:
ALP 439 GP        ∨

Exit time:      13:05

  4.2 - Exit parking area
```

```
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 The first character represents the type of license B: Bicycle licence C: Car licence 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: