



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/ NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

2022(2)

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:
DataENGJuneBackUp2022.exe.
- Click on the 'Extract' button.
- Enter the following password: **#locust*1**

Once extracted, the following list of files will be available in the folder **DataENGJuneBackUp2022:**

Question 1:

Bee.txt
Question1_P.dpr
Question1_P.dproj
Question1_P.res
Question1_U.dfm
Question1_U.pas

Question 2:

ConnectDB_U.pas
InsectMuseum.mdb
InsectMuseum - Copy.mdb
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question 3:

Inspection_U.pas
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas
Results.txt

Question 4:

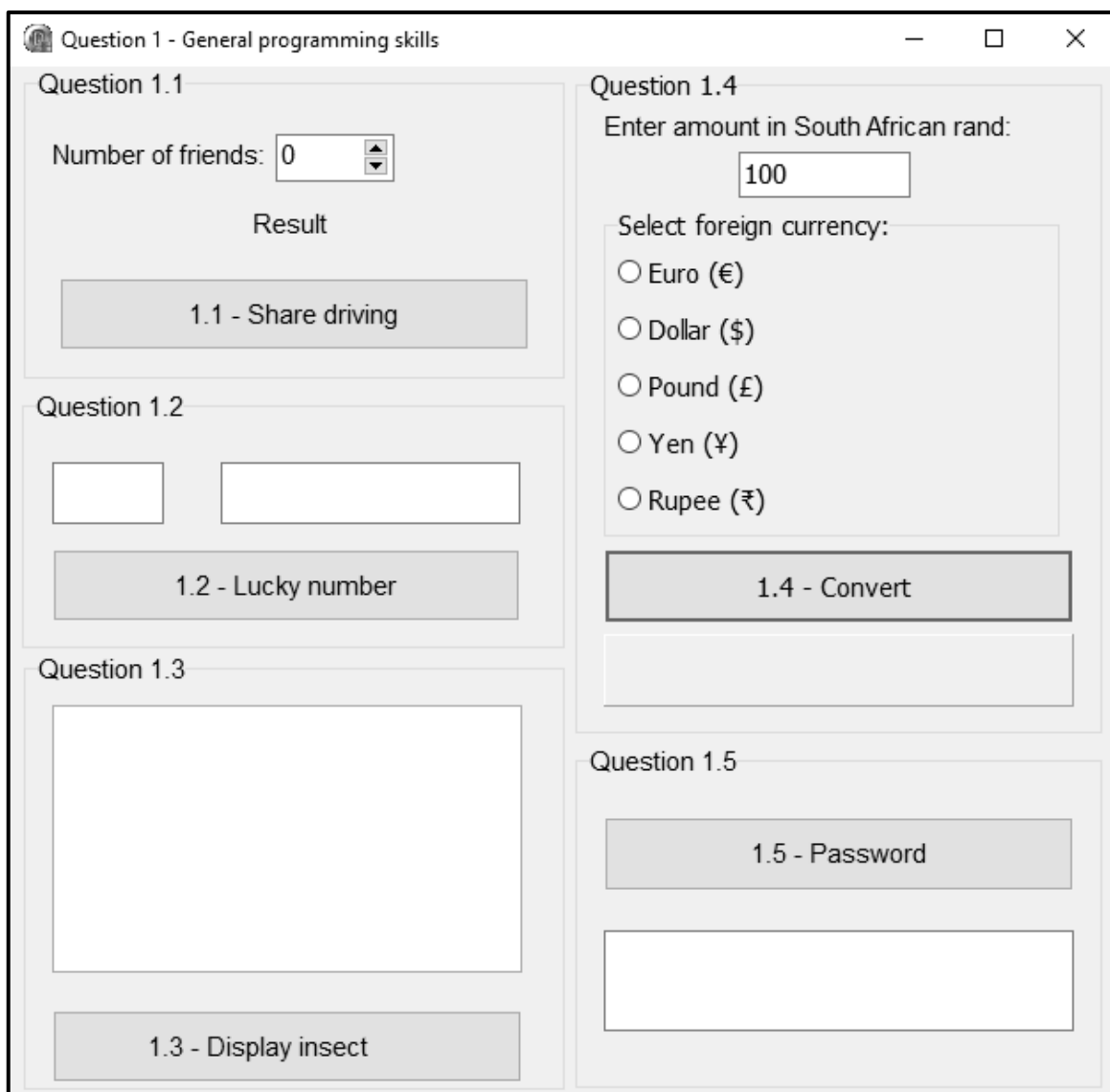
butterfly.jpg
caterpillar.jpg
dragonfly.jpg
ladybug.jpg
Question4_P.dpr
Question4_P.dproj
Question4_P.res
Question4_U.dfm
Question4_U.pas

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 1.5 that follow.

1.1 Button [1.1 – Share driving]


A group of university friends wants to take turns to drive a distance of 2 356,85 km to their destination. A constant variable **rDISTANCE** has been declared to store the value of 2356.85.

The user must select the number of friends in the **spnQ1_1** spin edit component.

Write code to do the following:

- Extract the number of friends that was selected from the **spnQ1_1** spin edit componen.
- Calculate the number of kilometres each friend will drive if the number of kilometres to drive is shared equally. Use the provided constant variable **rDISTANCE** in the calculation.
- Use the label **lblQ1_1** to display the number of kilometres each friend will drive, rounded to the nearest integer.

Example of output if the number of friends selected is 4:



(4)

1.2 Button [1.2 – Lucky number]

A person can gain free entry to an exhibition if their ticket number is a perfect square. A number is a perfect square if the square root of the number is a whole number.

Write code to do the following:

- Assign a random value in the range from 1 to 20 (inclusive) to the provided variable **iTicketNumber** as the ticket number.
- Display the ticket number in the edit box **edtQ1_2**.
- Test whether the ticket number is a perfect square or not.

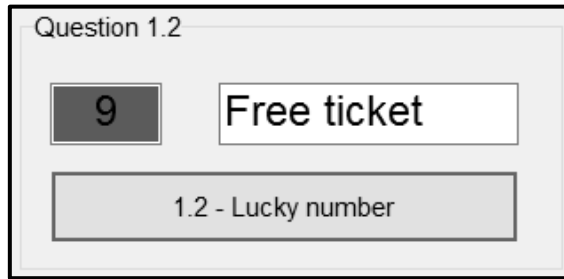
If the number is a perfect square, do the following:

- Change the colour of the edit box **edtQ1_2** to green.
- Display the message 'Free ticket' in the **edtQ1_2Ticket** edit box.

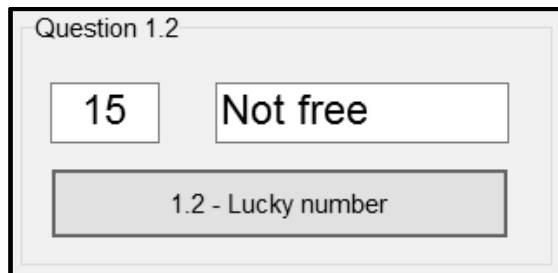
If the number is NOT a perfect square, do the following:

- Change the colour of the edit box **edtQ1_2** to white.
- Display the message 'Not free' in the **edtQ1_2Ticket** edit box.

Example of output if the value of 9 was generated as the ticket number:



Example of output if the value of 15 was generated as the ticket number:



(8)

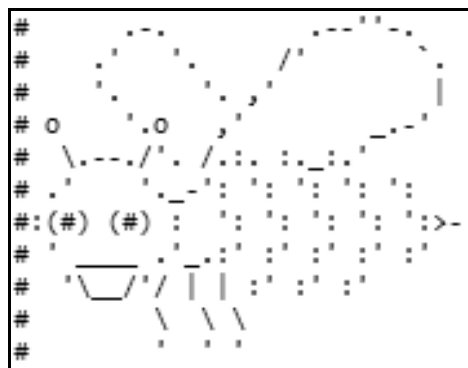
1.3 Button [1.3 – Display insect]

A picture has been compiled using various characters to form a bee-like figure. The lines of characters have been saved in the provided text file **Bee.txt**.

Write code to display the contents of the text file **Bee.txt** in the **redQ1_3** rich edit with a hash character (#) at the beginning of each line.

NOTE: NO marks will be awarded for the use of the LoadFromFile function.

Example of output:



(8)

1.4 Button [1.4 – Convert]

South Africans travelling abroad usually need to convert South African rand into other foreign currencies.

The user must do the following:

- Use the edit box **edtQ1_4_1** to enter the amount to be converted into South African rand.
- Use the radio group **rgpQ1_4** to select the foreign currency that the amount must be converted into.

Write code to do the following:

- Declare suitable variables to store the amount that was entered, the exchange rate and the converted amount.
- Extract the amount that was entered from the edit box **edtQ1_4_1**.
- Determine the exchange rate using the following table:

FOREIGN CURRENCY	EXCHANGE RATE OF ONE SOUTH AFRICAN RAND
Euro	16.90
Dollar	15.35
Pound	20.42
Yen	0.13
Rupee	0.20

- Convert the amount that was entered into the selected foreign currency.
- Display the converted amount on the panel **pnlQ1_4**, formatted to TWO decimal places.
- The display must include the currency from the option selected in the radio group.

Example of output if an amount of 100 South African rand was entered and dollar (\$) is selected (left-hand side) and rupee (₹) is selected (right-hand side).

Question 1.4

Enter amount in South African rand:

Select foreign currency:

Euro (€)

Dollar (\$)

Pound (£)

Yen (¥)

Rupee (₹)

1.4 - Convert

6.51 Dollar (\$)

Question 1.4

Enter amount in South African rand:

Select foreign currency:

Euro (€)

Dollar (\$)

Pound (£)

Yen (¥)

Rupee (₹)

1.4 - Convert

500.00 Rupee (₹)

(10)

1.5 Button [1.5 – Password]

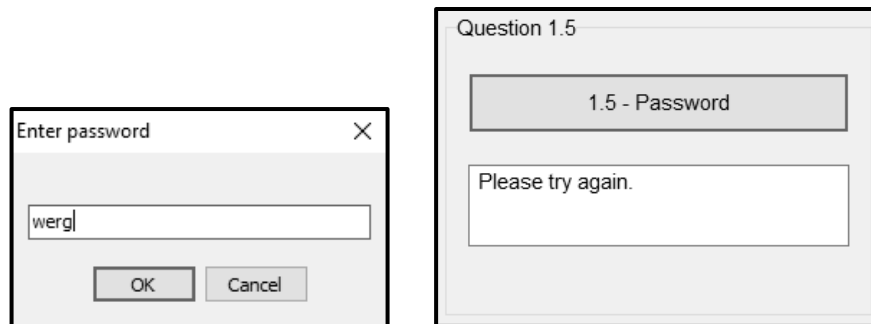
A strong password must include at least one special character (! @ # \$ %) and must consist of more than eight characters.

An input box must be used to enter a password.

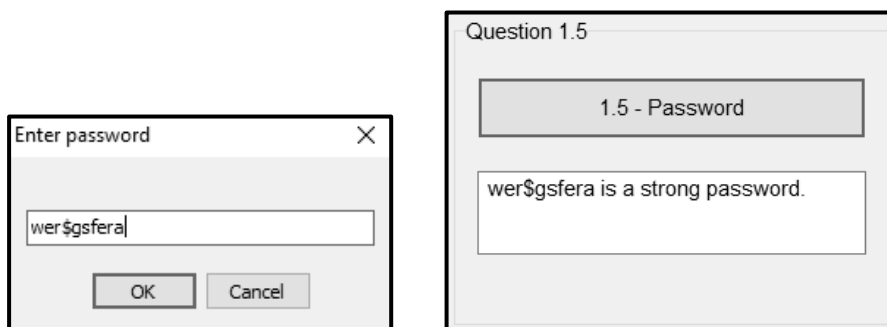
Write code to do the following:

- Save the password entered in the input box in a variable.
- Test whether the password is a strong password or not.
- If the password is not a strong password, display a suitable message and keep on asking the user to enter another password until a strong password is entered.

Example of input and output if a weak password was entered:



Example of input and output if a strong password was entered:



(10)

- 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**

An insectarium is a place where a collection of insects is housed and exhibited. The insectarium at a local museum has developed a database called **InsectMuseum.mdb**, which contains information about different insects and also the rooms in which they are exhibited. The database contains two tables called **tblExhibitRooms** and **tblInsects**. The manager at the insectarium requires your assistance with retrieving information from the database and its maintenance.

The data pages attached at the end of the question paper provide information on the design of the **InsectMuseum.mdb** database and its contents.

Do the following:

- Open the incomplete project file called **Question2_P.dpr** in the **Question 2** folder.
- Add 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**.

ExhibitRoomID	MaxExhibits	Category	LiveSpecimens
EX01	40	Beetles	False
EX02	30	Dragonflies	False
EX03	45	Crickets	False
EX04	15	Cockroaches	True

InsectCode	ScientificName	Venomous	Weight	Habitat	ExhibitRoomID
BAW01	Goliathus goliatus	False	285	Forest	EX01
BAW02	Megasoma elephas	True	160	Salt water	EX01
BAW03	Dynastes hercules	False	250	Forest	EX01
BAW04	Chalcosoma atlas	False	200	Desert	EX01

- 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.

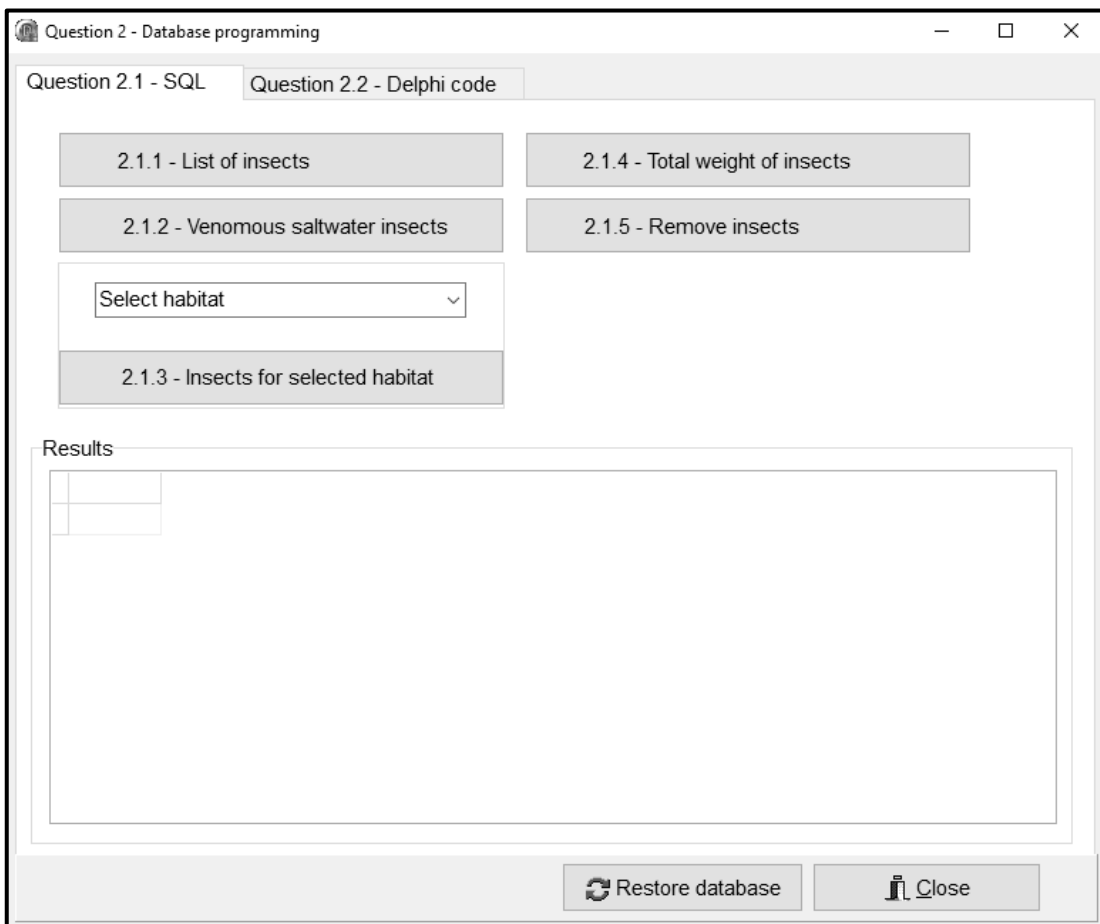
NOTE:

- 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 provided code.
- TWO variables are declared as public variables as described in the table below.

Variable	Data type	Description
tblExhibitRooms	TADOTable	Refers to the table tblExhibitRooms
tblInsects	TADOTable	Refers to the table tblInsects

2.1 Tab sheet [Question 2.1 – SQL]

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



NOTE: Code to execute the SQL statements and display the results of the queries are 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 insects]**

Display all details of the insects in the **tblInsects** table, sorted in ascending order of habitat and in descending order of the weight of the insects.

Example of output of the first five records:

InsectCode	ScientificName	Venomous	Weight	Habitat	ExhibitRoomID
BAW04	Chalcosoma atlas	False	200	Desert	EX01
UNK03	Stomaphis quercus	True	143	Desert	EX08
UNK04	Ledromorpha planirostris	False	116	Desert	EX08
STC03	Ctenomorpha gargantua	False	40	Desert	EX07
WAS02	Vespa mandarinia	False	32	Desert	EX06

(3)

2.1.2 **Button [2.1.2 – Venomous saltwater insects]**

Display the insect code and scientific name in the **tblInsects** table of all venomous insects that live in salt water.

Example of output:

InsectCode	ScientificName
UNK01	Andalucia
BAW02	Megasoma elephas
COC05	Grylloblatta campode
PRA04	Hierodula

(4)

2.1.3 **Button [2.1.3 – Insects for selected habitat]**

The user must select a habitat from the **cmbQ2_1_3** combo box. Display the **InsectCode** field and **ScientificName** field of all the insects in the habitat selected by the user.

Code has been provided to extract and save the selected habitat in a variable called **sHabitat**.

Example of output if 'Desert' was selected as habitat:

InsectCode	ScientificName
WAS02	Vespa mandarinia
BAW04	Chalcosoma atlas
STC03	Ctenomorpha gargantua
UNK03	Stomaphis quercus
UNK04	Ledromorpha planirostris

(4)

2.1.4 **Button [2.1.4 – Total weight of insects]**

Display the **ExhibitRoomID** and the total weight of the insects in each room in kilograms only if the total weight is more than one kilogram.

Use **Total Weight** as the name of the calculated field for the total weight of the insects.

Example of output:

ExhibitRoomID	Total Weight
EX01	1.075
EX04	1.185

(9)

2.1.5 **Button [2.1.5 – Remove insects]**

Exhibition room **EX08** will be renovated and all the insect exhibits must be moved from this room into storage.

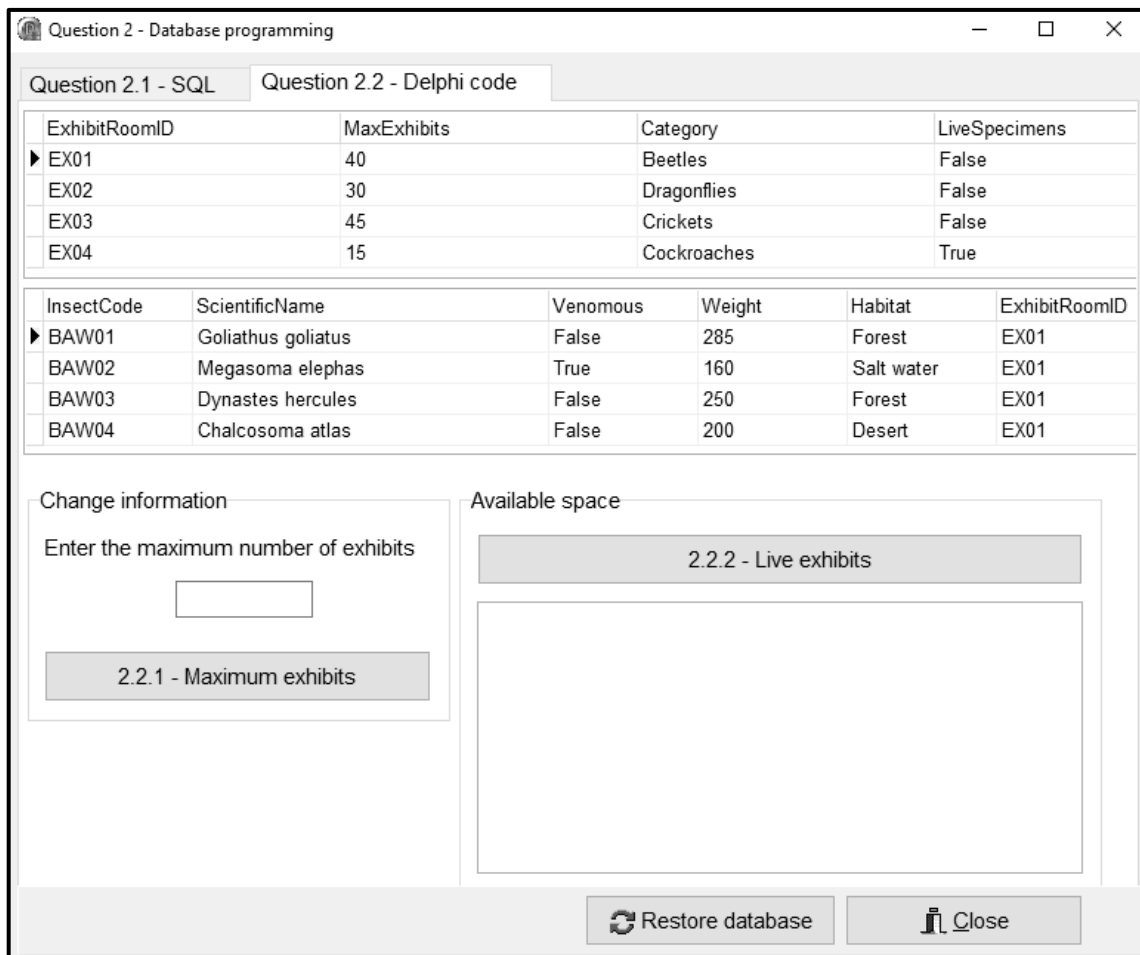
Remove all the records in the **tblInsects** table where the **ExhibitRoomID** is **EX08**.

Code has been provided to display a message to indicate that the content of the database has been changed.

(3)

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 2.2.1 and QUESTION 2.2.2.
- NO marks will be awarded for SQL statements in QUESTION 2.2.

2.2.1 **Button [2.2.1 – Maximum exhibits]**

Sometimes exhibitions are moved to other rooms, which means that the maximum number of exhibits allowed in an exhibition room may change.

In this instance, the user must select the **ExhibitRoomID** of the room from the **tblExhibitRooms** table in the dbgrid (dbgRooms) and enter the adjusted maximum number of exhibits in the **edtQ2_2_1** edit box.

Code has been provided to extract and save the adjusted maximum number of exhibits from the **edtQ2_2_1** component.

Write code to change the maximum number of exhibits in the selected exhibition room to the value that was entered.

Example of the result if the exhibition room with the code **EX01** was selected and a value of 60 was entered:

The maximum number of exhibits allowed in room **EX01** before the change:

ExhibitRoomID	MaxExhibits	Category	LiveSpecimens
EX01	40	Beetles	False

The maximum number of exhibits allowed in room **EX01** after the change:

ExhibitRoomID	MaxExhibits	Category	LiveSpecimens
EX01	60	Beetles	False

(4)

2.2.2 **Button [2.2.2 – Live exhibits]**

A report that displays the code and weight of the insects in each room accommodating live exhibits has been requested. The total number of insects in each of these rooms and the number of spaces still left for more insects must also be calculated and displayed.

Write code to do the following:

Do the following for each room with live exhibits:

- Display the **ExhibitRoomID** of the room as a heading in the rich edit component **redQ2_2_2**.
- Display the **InsectCode** and **Weight** of all the insects in the room.
- Determine and display the total number of insects in the room.
- Determine and display the number of spaces left to accommodate more insects in the room.

Example of output of the first room in the **tblInsects** table that accommodates live insect exhibits:

Available space	
2.2.2 - Live exhibits	
Insects in exhibit room: EX04	
Insect code	Weight
COC01	120 g
COC02	230 g
COC03	275 g
COC04	240 g
COC05	320 g
Total insects: 5	
Number of spaces for more insects: 10	

(13)

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

TOTAL SECTION B: 40

SECTION C**QUESTION 3: OBJECT-ORIENTATED PROGRAMMING**

South Africa is home to locust species that regularly swarm and damage crops. A reliant inspection system has been set up to determine whether a reported locust outbreak requires a farm to be treated immediately, or not.

The incomplete program provided must be completed and used to determine whether an identified swarm of locusts poses a threat and requires treatment, or not.

Do the following:

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

Example of graphical user interface (GUI):

The screenshot shows a graphical user interface for a locust inspection system. The window title is "Question 3 - Object-oriented programming". The main heading is "Locust inspections". The interface includes several input fields and buttons:

- Details of the locust inspection:**
 - Farm name:
 - Inspection date:
 - Farm size:
 - Locust development stage:
- Display area:** A large empty rectangular area on the right side of the window.
- Duration and pesticide:**
 - 3.2.2 - Duration of outbreak:
 - 3.2.3 - Amount of pesticide:
- Treatment required:**
 - Infected area of farm in hectares:
 - 3.2.4 - Write to file:

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

- 3.1 The provided incomplete object class (**TInspection**) contains the declaration of four attributes which are used to describe an **Inspection** object and methods.

The following have been provided in the object class:

- The declaration of the following attributes for a **TInspection** object:

Attribute	Description
fFarmName	The name of a farm
fInspectionDate	The date of the inspection (dd/mm/yyyy)
fFarmSize	The total size of the farm measured in hectares
fLocustStage	The current development stage of a locust (egg, nymph, hopper or adult locust)

- A completed constructor named 'Create'
- A completed **toString** method that displays the details of an inspection in the following format:

```
Farm name: <Farm name>
Inspection date: <Inspection date>
Farm size: <Farm size> hectares
Locust stage: <Locust development stage>
```

Complete the code in the object class as described in QUESTION 3.1.1 to QUESTION 3.1.4 below.

- 3.1.1 Write an accessor method called **getFarmName** to return the name of the farm. (2)

- 3.1.2 Write a method called **calcPesticideRequirement** to calculate and return the amount of pesticide required in litres to treat the farm. The return value must be rounded to the next integer value.

NOTE: A litre of pesticide is used to treat a maximum of 0,85 hectares. (3)

- 3.1.3 The development stage of the locust indicates the duration of the outbreak in weeks.

Use the information in the table below and write a method called **determineDuration** to determine the duration of the outbreak and to return a string that describes the duration of the outbreak.

Locust stage of development	Duration of the outbreak
Nymph	Less than 2 weeks
Hopper	2 to 8 weeks
Adult	More than 8 weeks

(7)

- 3.1.4 Write a method called **treatmentRequired** that receives the size of the infected area of the farm, in hectares, as a parameter.

Return the string 'Treatment required' if ONE of the following conditions is met:

- The development stage of the locust is 'Adult'.
- OR**
- The infected area is more than 25% of the total size of the farm.

Return the string 'Treatment not required' if none of the conditions above is met.

(7)

- 3.2 An incomplete program, **Question3_P**, has been supplied in the **Question 3** folder. The program contains code for the object class to be accessible and declares an object variable called **objInspection**.

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

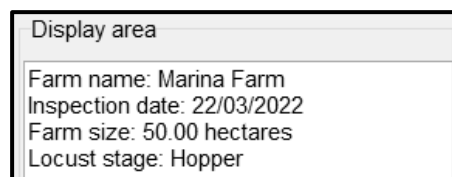
3.2.1 **Button [3.2.1 – Instantiate and display object details]**

Code has been provided to extract the farm name, inspection date and the farm size from the relevant edit boxes.

Write code to do the following:

- Extract the development stage of the locusts from the combo box **cmbQ3_1**.
- Instantiate a new inspection object.
- Use the **toString** method to display the details of the object in the rich edit **redQ3**.

Example of output:

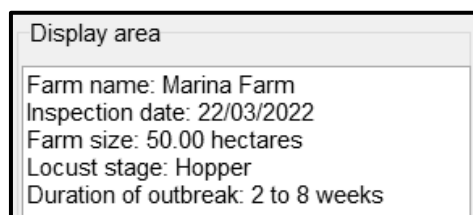


(5)

3.2.2 **Button [3.2.2 – Duration of outbreak]**

Write code to call the relevant method to determine the duration of the outbreak and display the result in the rich edit **redQ3**.

Example of output if the development stage of the locusts is 'hopper':

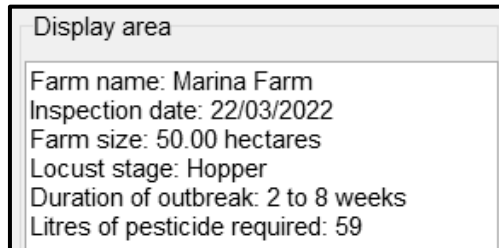


(3)

3.2.3 Button [3.2.3 – Amount of pesticide]

Write code to call the relevant method to calculate the litres of pesticide required to treat the infected area and display the result in the rich edit **redQ3**.

Example of output with the details of the farm, the outbreak and the amount of pesticide required:



(4)

3.2.4 Button [3.2.4 – Write to file]

The user must enter the size of the infected area of the farm in hectares in the **edtQ3_2_4** edit box.

The size of the infected farm area will be used to determine whether the farm requires treatment or not. The outcome of the inspection is added to the data saved in the provided text file **Results.txt** in the following format:

```
<Name of farm>
<Treatment required/not required>
```

Content of the provided text file **Results.txt**:

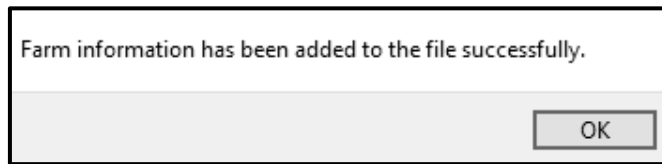
```
Nottingham
Treatment required
Clarens Farm
Treatment not required
```

Code has been provided to extract the size of the infected area of the farm in hectares from the **edtQ3_2_4** edit box.

Write code to do the following:

- Open the text file **Results.txt** for new information to be added.
- Call the **treatmentRequired** method and use the size of the infected area as an argument to determine whether the farm requires treatment or not.
- Write the following information to the text file:
 - The name of the farm
 - Whether or not treatment is required
- Use a dialogue box to indicate that the information has been written to the file.

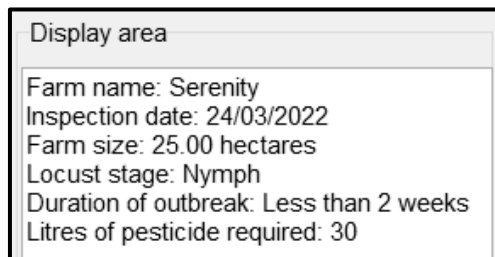
Example of message to be displayed:



Example of the content of the text file **Results.txt** after the information has been added to the file:

```
Nottingham
Treatment required
Clarens Farm
Treatment not required
Marina Farm
Treatment required
```

Example of another inspection that has been conducted with the following information:



Example of the content of text file **Results.txt** after the inspection has been completed and 5 hectares is entered as the infected area:

```
Nottingham
Treatment required
Clarens Farm
Treatment not required
Marina Farm
Treatment required
Serenity
Treatment not required
```

(9)

- 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

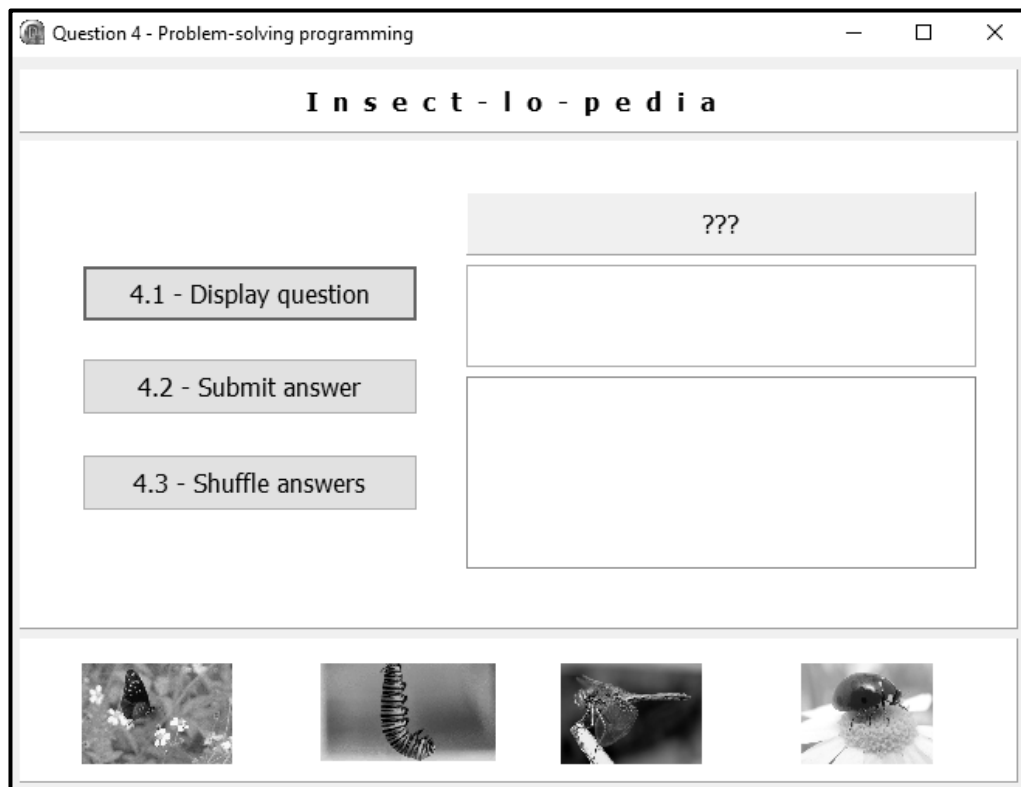
SECTION D**QUESTION 4: PROBLEM-SOLVING PROGRAMMING**

The Insect-lo-pedia is holding their annual quiz competition. You have been asked to assist the organisation in developing an electronic multiple-choice program.

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 code has been provided:

- An array **arrQuesAns** consisting of ten questions with four multiple-choice answers each

The format of the elements in the array is given below.

```
((<Question1>, <answer1>, <*answer2>, <answer3>, <answer4>),
(<Question2>, <*answer1>, <answer2>, <answer3>, <answer4>),
...)
```

The correct answer to a question is indicated with an asterisk (*) as the first character.

Examples of the given code for the first three questions and possible answers are given below.

```
arrQuesAns: array [1..10, 1..5] of String = (
  ('A caterpillar is the ..... stage of a butterfly.',
   'pupal', '*larva', 'adult', 'infant'),
  ('One of the development stages of a locust is a .....',
   '*hopper', 'larva', 'pupal', 'flyer'),
  ('Most insects respire through their .....',
   'skin', 'gills', 'lungs', '*tracheal system'),
  .....
```

- Two global variables **iCounter** and **iScore** have been declared and initialised to zero:

```
iCounter: Integer = 0;
iScore: Integer = 0;
```

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

4.1 **Button [4.1 – Display question]**

Questions and answers from the **arrQuesAns** array must be displayed in sequence from Question 1 to Question 10, one question at a time on the click of **btnQ4_1**.

Write code to do the following:

- Use the **iCounter** variable to display the number of the question in the panel **pnlQ4**.
- Display the corresponding question from the array **arrQuesAns** in the rich edit **redQ4**.
- Display the four possible answers in the list box **lsbQ4**. The asterisk character indicating the correct answer must NOT be displayed.

Code has been provided to do the following:

- Enable the list box **lsbQ4** after the 'Display question' button has been clicked.
- Disable the **btnQ4_1** button after all ten questions have been displayed.

Example of output for the first question in the **arrQuesAns** array:

Question 1
A caterpillar is the stage of a butterfly.
pupal larva adult infant

(7)

4.2 Button [4.2 – Submit answer]

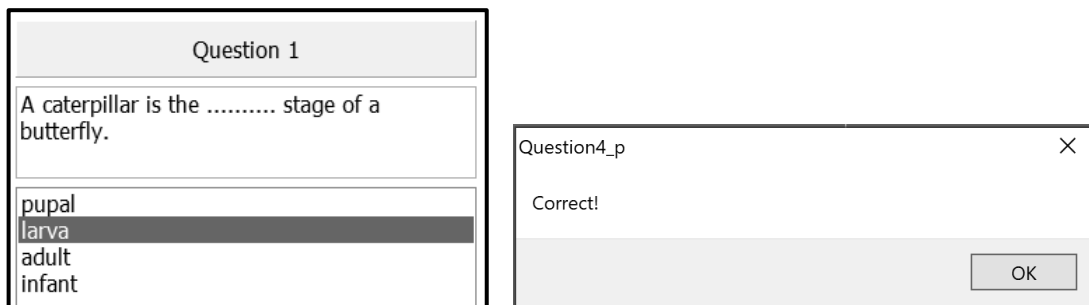
The user is required to select an answer from the list box **IsbQ4** on the display of a question and possible answers.

Write code to do the following:

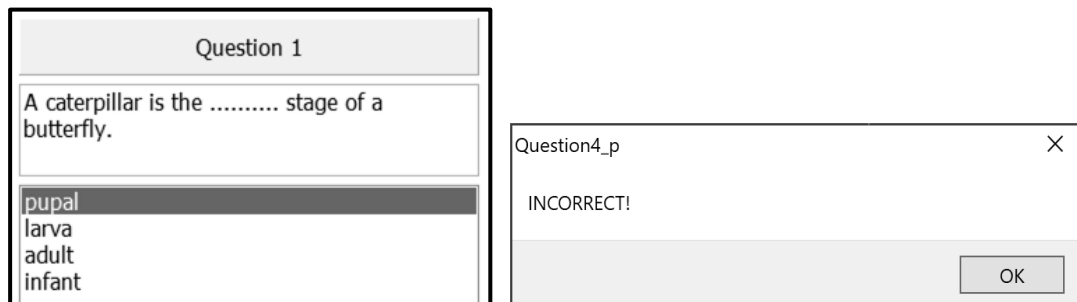
- If the **Submit answer** button has been clicked without an answer being selected from the **IsbQ4** list box, do the following:
 - Use a ShowMessage dialogue box to display the message: 'No answer provided! No score.'
 - Assign the value of 5 to the ItemIndex of the **IsbQ4** list box.
- If an answer has been selected and submitted, do the following:
 - Obtain the selected answer from the list box **IsbQ4**.
 - Test whether the selected answer is correct or not.
 - Use a show message dialogue box to display an appropriate message.
 - If the answer is correct, increment the score by the value of 1.
- Once all the questions have been answered, use a ShowMessage dialogue box to display the final score.

Code has been provided to disable the list box **IsbQ4** after an answer has been selected.

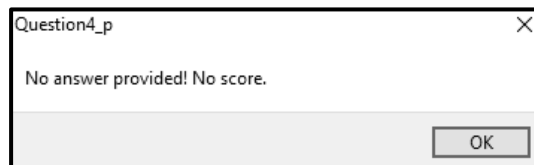
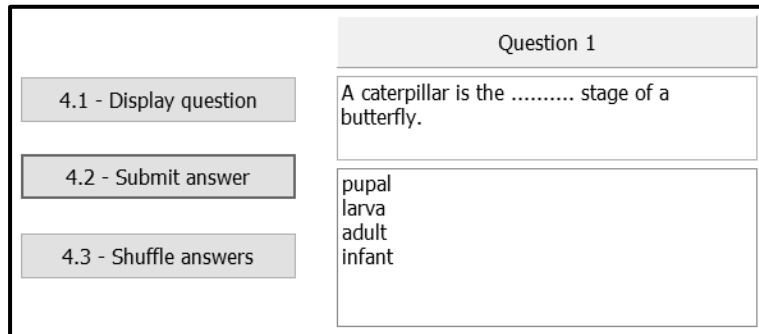
Example of output if the first question has been answered correctly:



Example of output if the first question has been answered incorrectly:



Example of output if **Submit answer** button has been clicked without an answer being selected:



Example of output of the final score once all the questions have been answered:



(10)

4.3 **Button [4.3 – Shuffle answers]**

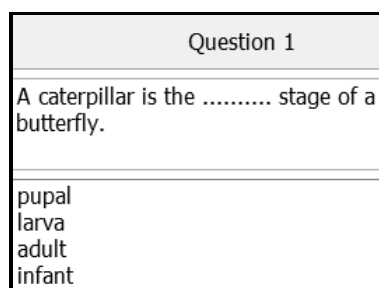
The shuffling of answers is necessary to allow contestants to attempt the same questions again.

Code has been provided to enable button **btnQ4_1** to allow for the display of questions after shuffling the answers.

Write code to do the following:

- Randomly swap the options provided as answers for each question in array **arrQuesAns**.
- Set the **iCounter** variable to zero.
- Use a ShowMessage dialogue box to display a message to indicate that the answers have been shuffled.

Example of output of the original answers for Question 1:



Example of output of possible answers after the shuffling of the answers for Question 1:

Question 1
A caterpillar is the stage of a butterfly.
larva infant pupal adult

(12)

- 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 FOR QUESTION 2:**

The design of the database tables is as follows:

Table: **tblExhibitRooms**

This table contains the details of the rooms where the insects are exhibited.

Field name	Data type	Description
ExhibitRoomID	Text (5)	Unique ID for each exhibit room
MaxExhibits	Number	The maximum number of insects that can be accommodated in a room
Category	Text (30)	The category of insects in the room
LiveSpecimens	Boolean	A true value to indicate whether the specimens in the room are alive or not

Example of the first eight records of the **tblExhibitRooms** table:

ExhibitRoomID	MaxExhibits	Category	LiveSpecimens
EX01	40	Beetles	<input type="checkbox"/>
EX02	30	Dragonflies	<input type="checkbox"/>
EX03	45	Crickets	<input type="checkbox"/>
EX04	15	Cockroaches	<input checked="" type="checkbox"/>
EX05	50	Praying mantises	<input type="checkbox"/>
EX06	50	Wasps	<input checked="" type="checkbox"/>
EX07	35	Stick insects	<input checked="" type="checkbox"/>
EX08	20	Unknown	<input type="checkbox"/>

Table: **tblInsects**

This table contains the information of the insects.

Field name	Data type	Description
InsectCode	Text (5)	Unique code for each insect
ScientificName	Text (24)	Scientific name of the insect
Venomous	Boolean	Boolean value to indicate whether the insect is venomous or not
Weight	Number	The weight of the insect in grams
Habitat	Text (11)	The habitat of each insect (sand, fresh water, desert, mountain, salt water, woodland, orchard)
ExhibitRoomID	Text (5)	Unique ID that indicates the room in which the insect will be exhibited

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

InsectCode	ScientificName	Venomous	Weight	Habitat	ExhibitRoomID
BAW01	Goliathus goliatus	<input type="checkbox"/>	285	Forest	EX01
BAW02	Megasoma elephas	<input checked="" type="checkbox"/>	160	Salt water	EX01
BAW03	Dynastes hercules	<input type="checkbox"/>	250	Forest	EX01
BAW04	Chalcosoma atlas	<input type="checkbox"/>	200	Desert	EX01
BAW05	Xixuthrus heros	<input checked="" type="checkbox"/>	180	Sand	EX01
COC01	Megaloblatta	<input type="checkbox"/>	120	Woodland	EX04
COC02	Blaberus giganteus	<input type="checkbox"/>	230	Woodland	EX04
COC03	Macropanesthia rhino	<input checked="" type="checkbox"/>	275	Forest	EX04
COC04	Lethocerus	<input type="checkbox"/>	240	Fresh water	EX04
COC05	Grylloblatta campode	<input checked="" type="checkbox"/>	320	Salt water	EX04

NOTE:

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

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

