basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/
NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

2019

MARKS: 150

TIME: 3 hours

This question paper consists of 21 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 the Delphi programming language.

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

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

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

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

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

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

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

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

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

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

NOTE: Candidates must use the file **DataENGJun2019.exe**.

Do the following:

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

Once extracted, the following list of files will be available in the folder **DataENGJun2019**:

SUPPLIED FILES

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

Question 2:
ConnectDB_U.dcu
ConnectDB_U.pas
NurseryDB.mdb
NurseryDB_Backup.mdb
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question 3:
Logbook.txt
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas
star.png
Trainee_U.pas

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

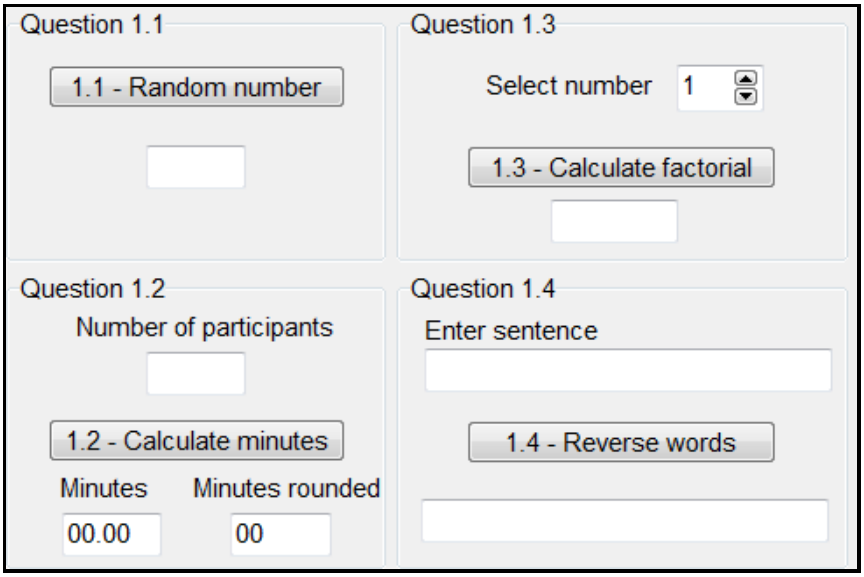
SECTION A

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 user interface displays FOUR sections labelled QUESTION 1.1 to QUESTION 1.4. The program has no functionality currently.

Example of the graphical user interface (GUI):



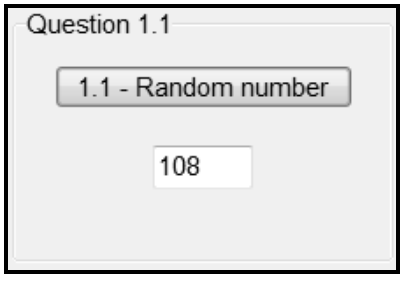
Follow the instructions below to complete the code for EACH section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.4.

1.1 Button [1.1 - Random number]

Write code to do the following:

- Generate a random number in the range 100–120 (inclusive).
- Display the random number in the **edtRandomNumber** edit box.

Example of output if the random number generated is 108:



(4)

1.2 **Button [1.2 - Calculate minutes]**

A school dance has many learners participating. The organisers need to determine how long it will take to introduce all the participants. The time allocated to introduce each participant will depend on the total number of participants. The following criteria will be used:

Number of participants	Number of minutes per participant
Less than or equal to 20	2.5 minutes
More than 20 and less than or equal to 50	2.3 minutes
More than 50	2.0 minutes

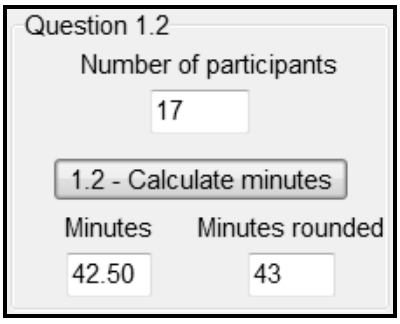
The user must enter the number of participants in the **edtParticipants** edit box. Write code to do the following:

- Extract the number of participants from the **edtParticipants** edit box.
- Calculate the total number of minutes that will be required to introduce ALL the participants. Display the value in the **edtMinutes** edit box to TWO decimal places.
- Round up the calculated number of minutes and display the result in the **edtMinsRounded** edit box.

Use the following test data:

Number of participants	Minutes	Minutes rounded
13	32.50	33
21	48.30	49
50	115.00	115
51	102.00	102

Example of output if the number of participants is 17:



(13)

1.3 **Button [1.3 - Calculate factorial]**

The factorial of a number is the product of multiplying all integers from 1 to the number, e.g.

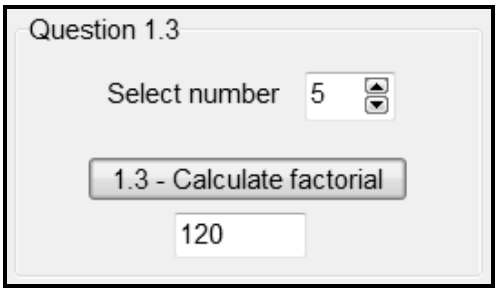
The factorial of 4 = 1 x 2 x 3 x 4
= 24

The factorial of 6 = 1 x 2 x 3 x 4 x 5 x 6
= 720

The user must select a number from the **spnNumber** spin edit box.

Write code to extract the number from the **spnNumber** spin edit box, calculate the factorial of the number and display the result in the **edtFactorial** edit box.

Example of output if the number 5 is selected:



(7)

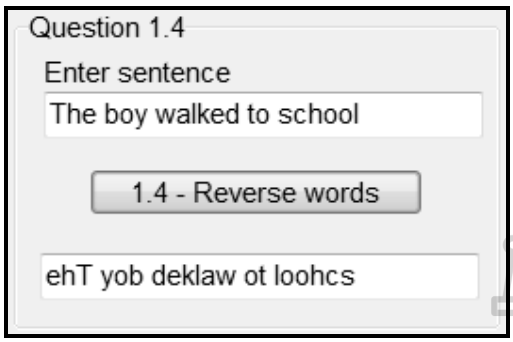
1.4 **Button [1.4 - Reverse words]**

The characters in words in a sentence must be reversed for encryption purposes. The user must enter a sentence in the **edtSentence** edit box.

Write code to do the following:

- Extract the sentence from the **edtSentence** edit box.
- Reverse the characters in EACH word in the sentence.
- Display the result in the **edtReverse** edit box.

Example of input and output:



(16)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION A: 40

SECTION B

QUESTION 2: DATABASE PROGRAMMING

PJB Wholesale Garden Centre supplies plants to a number of nurseries. When a quotation for an order is given, the requested number of plants is checked for availability.

The database **NurseryDB** for PJB Wholesale Garden Centre contains two tables, namely **tblPlants** and **tblOrders**.

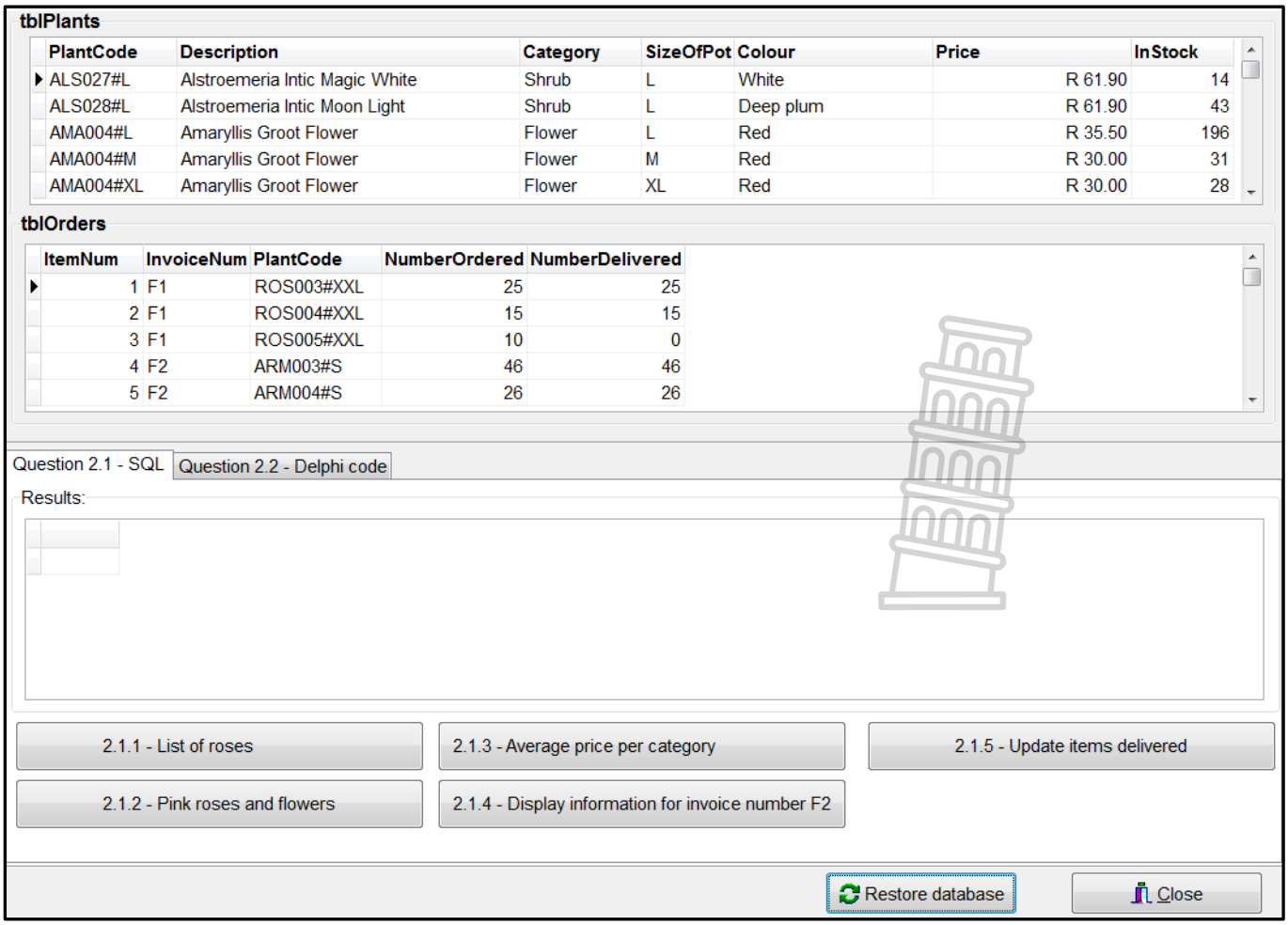
The data pages attached at the end of this question paper provide information on the design of the database and examples of the contents of the tables.

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 currently has no functionality.

NOTE: If the compiler shows an error message with regard to the given CurrencyString statement, remove the statement.

The following user interface is displayed:



- Carry out the following instructions to complete the code for each section, as described in QUESTION 2.1 and QUESTION 2.2.
- Use SQL statements to answer QUESTION 2.1 and Delphi code to answer QUESTION 2.2.

NOTE:

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

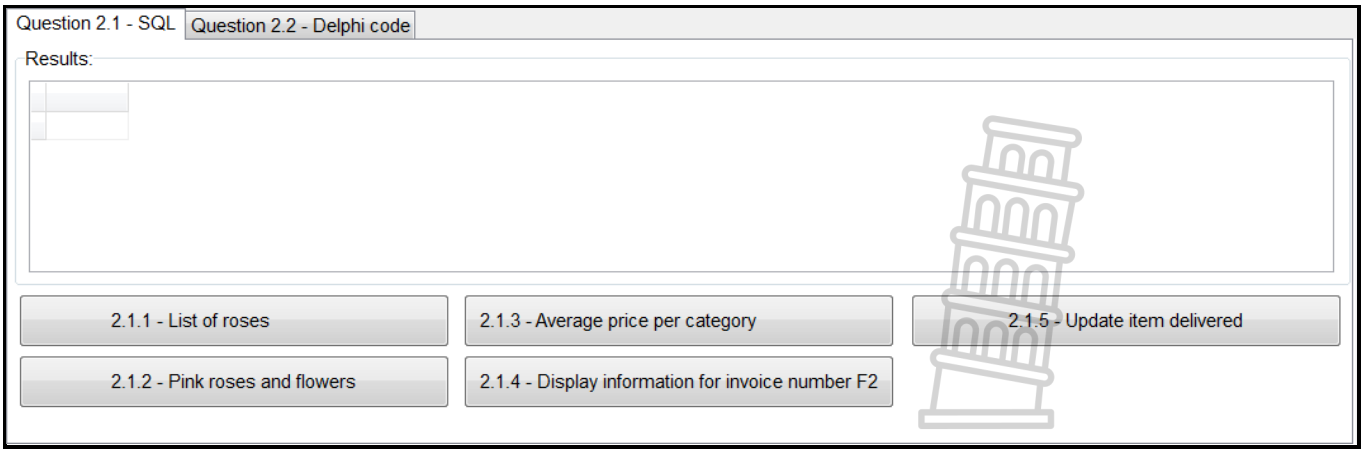
Variable	Data type	Description
`tblPlants`	`TADOTable`	Refers to the table tblPlants
`tblOrders`	`TADOTable`	Refers to the table tblOrders

2.1 Tab sheet [Question 2.1 - SQL]

In this section you may use ONLY SQL statements 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 are incomplete.

The following GUI is displayed:



Do the following:

Enter the SQL statements for QUESTION 2.1.1 to QUESTION 2.1.5 which is assigned to the variables **sSQL1**, **sSQL2**, **sSQL3**, **sSQL4** and **sSQL5** respectively.

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

Display all the details of the plants in the Rose category in the **tblPlants** table.

Example of output of the first five records:

PlantCode	Description	Category	SizeOfPot	Colour	Price	InStock
ROS002#XXL	Pernille Poulsen	Rose	XXL	Salmon pink	66.95	175
ROS003#XXL	Lisa	Rose	XXL	Deep pink	66.95	187
ROS004#XXL	Ester Geldenhuys	Rose	XXL	Coral	66.95	176
ROS005#XXL	Just Joey	Rose	XXL	Shades of copper	66.95	67
ROS007#XXL	Andrea Stelzer	Rose	XXL	Clear pink	66.95	150

(3)

2.1.2 **Button [2.1.2 - Pink roses and flowers]**

Display the PlantCode, Category, Colour and SizeOfPot fields from the **tblPlants** table for all the plants in the **Flower** and **Rose** categories that have any type of **pink** colour.

Example of output of the last five records:

PlantCode	Category	Colour	SizeOfPot
LIS001#L	Flower	Pink	L
LIS001#S	Flower	Pink	S
ROS002#XXL	Rose	Salmon pink	XXL
ROS003#XXL	Rose	Deep pink	XXL
ROS007#XXL	Rose	Clear pink	XXL

(6)

2.1.3 **Button [2.1.3 - Average price per category]**

Use data from the **tblPlants** table to determine the average price for plants per category using **AveragePrice** as the new field name for the calculated field. The average price should be displayed in currency format.

Example of output:

Category	AveragePrice
Conifer	R 54.97
Creeper	R 59.33
Flower	R 28.78
Rose	R 69.20
Shrub	R 47.35

(5)

2.1.4 **Button [2.1.4 - Display information for invoice number F2]**

The invoice number will be the same for all items of a specific order.

Display the InvoiceNum, Description and NumberOrdered fields of all items ordered in the **tblOrders** table for invoice number F2.

Example of output:

InvoiceNum	Description	NumberOrdered
F2	Armeria Ballerina	46
F2	Armeria Ballerina	26
F2	Cyrtanthus Mackenii	85
F2	Cyrtanthus Mackenii	44

(5)

2.1.5 **Button [2.1.5 - Update items delivered]**

The number of items delivered differs sometimes from the number of items ordered in the table **tblOrders**.

Once an outstanding delivery has been made, the user must enter the item number (ItemNum) for the delivery that has been made. Write an SQL statement for variable **sSQL5** to modify the **NumDelivered** field to contain the same value as the **NumberOrdered** field.

NOTE: Code is provided to request the user to enter the item number.

Example of output for item number 6 before the record was modified:

ItemNum	InvoiceNum	PlantCode	NumberOrdered	NumberDelivered
6	F2	CYR001#S	85	45

Example of output for item number 6 after the record had been modified:

ItemNum	InvoiceNum	PlantCode	NumberOrdered	NumberDelivered
6	F2	CYR001#S	85	85

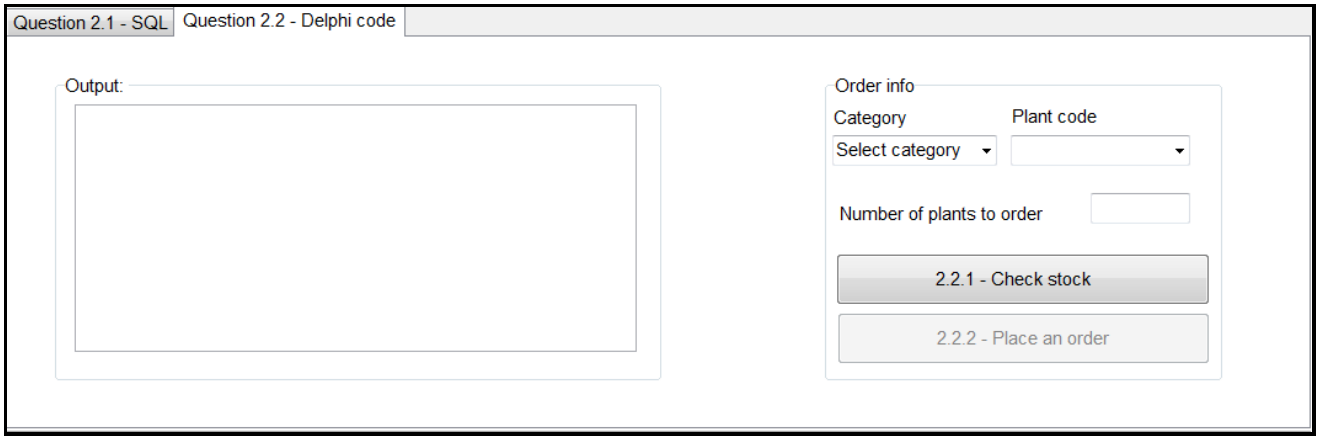
(4)

2.2 Tab sheet [Question 2.2 - Delphi code]

Use only programming code in this section to answer QUESTION 2.2.1 to QUESTION 2.2.2.

NO marks will be awarded for SQL statements in QUESTION 2.2.

The GUI for QUESTION 2.2 is shown below.



NOTE: The variables **iNumOrdered** and **sPlantCode** are provided as global variables. The content of these variables must be used in QUESTION 2.2.1 to check the availability of stock and in QUESTION 2.2.2 to place the order.

2.2.1 Button [2.2.1 - Check stock]

The stock available must be checked before an order can be placed.

The user must do the following:

- Select a category from the **Category** combo box.

Code is provided to populate the combo box (**cmbPlantCode**) containing the plant code associated with the selected category.

- Select a plant code from the **cmbPlantCode** combo box.
- Enter the number of plants to be ordered in the provided edit box.

Code is provided to:

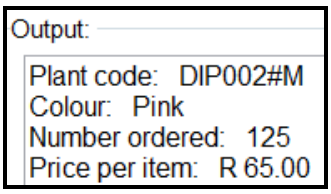
- Extract the plant code selected from the **cmbPlantCode** combo box
- Extract the number of plants entered from the **edtNumPlants** edit box

Write code to determine whether or not there is enough stock available.

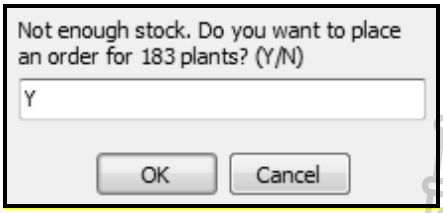
- If there is enough stock, display the plant code, colour, number of plants ordered and the price of the selected item in the output area **redDisplay** and enable the **btnQ2_2_2** button.

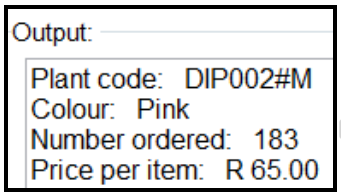
- If there is NOT enough stock available:
  - Display a message showing the number of plants in stock.
  - Ask the user whether he or she wants to continue to place the order for the available stock.
  - If the user wants to place the order:
    - Display the plant code, colour, number of plants ordered and the price of the selected item in the **redDisplay** output area.
    - Enable the **btnQ2_2_2** button.
  - If the user does NOT want to place the order:
    - Display the message 'Order cancelled' in the **redDisplay** output area.
    - Disable the **btnQ2_2_2** button.

Example of output if a request for 125 plants with code DIP002#M from the Creeper category was entered:

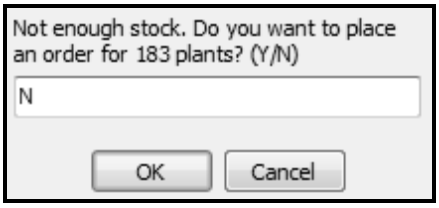


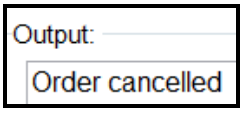
Example of output if a request for 225 plants with code DIP002#M from the Creeper category was entered with only 183 plants with this code being available:





Example of output if a request for 225 plants with code DIP002#M from the Creeper category was entered and the user does NOT want to place the order:





(11)

2.2.2 **Button [2.2.2 - Place an order]**

An order must be placed in the **tblOrders** table using the following information:

- The invoice number for this order in the **tblOrders** table must be F2.
- The plant code and number of plants ordered must be obtained from the two global variables, which contain information that was specified in QUESTION 2.2.1.
- The **NumberDelivered** field must be set to 0 since the delivery has not been made yet.

Example of output when placing an order for 125 plants with plant code DIP002#M:

(6)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION B: 40

SECTION C

QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

Star Nursery invites learners from the local schools to a voluntary training programme as part of a community initiative. Learners who are used as trainees must attend training sessions to enable them to assist customers with enquiries and with the sale of plants.

Do the following:

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

Example of graphical user interface (GUI) that will be displayed:

The program will use the number of hours of the training and the value of the sales made to determine whether or not the trainee qualifies for a bonus.

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

3.1 The incomplete object class (**TTrainee**) provided contains code for the following:

- Declarations of four attributes that describe a **Trainee** object
- A constructor **Create** method
- An incomplete **toString** method
- Two accessor methods: **getName** and **getTraineeID**

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

Names of attributes	Description
fTraineeID	Unique number to identify the trainee
fName	Name and surname of the trainee
fHours	Number of hours of training the trainee attended
fSales	Amount of the sales the trainee made

3.1.1 The number of hours of training attended by the trainee and the amount of the sales made can be increased.

(a) Write code for a method called **updateHours** that receives a value as a parameter. Increase the hours attribute using the received parameter value. (4)

(b) Write code for a method called **updateSales** that receives a value as a parameter. Increase the sales attribute using the received parameter value. (2)

3.1.2 Write code for a method called **qualifiesForBonus**. The method must return a Boolean value TRUE if the trainee qualifies for a bonus or FALSE if the trainee does not qualify for a bonus.

A trainee qualifies for a bonus if the following conditions are met:

- At least 15 hours of training was attended.
- The amount of the sales is at least R 1 200.00. (5)

3.1.3 Write code to complete the provided **toString** method to return a string in the following format:

<**name and surname of the trainee**> (<**the trainee ID**>) attended <**number of hours of training attended**> hours of training and sold plants to the value of <**value of the plants sold formatted to currency and two decimals**>.

Example of output when the **toString** method is called:

Kody Shaw (10) attended 65.66 hours of training and sold plants to the value of R 1 405.00. (4)

3.2 The incomplete unit **Question3_U** has been provided which contains code for the object class to be accessible. An object variable called **objTrainee** has been declared.

A text file called **Logbook.txt** contains the log entries of all the training and sales activities of the trainees.

Each log entry contains information of a trainee in the following format:

```
<trainee ID>;<character T or S indicating training or a sale that was made>#<numerical value indicating the number of hours of training or the value of the sale that was made>
```

Example of the first four entries in the text file:

```
12;T#0.98
15;S#182.00
13;S#118.00
10;T#1.96
```

The first two lines of text can be interpreted as follows:

- **12;T#0.98** – Trainee with ID number 12 attended 0.98 hours of training
- **15;S#182.00** – Trainee with ID number 15 made a sale to the value of R 182.00

Follow the instructions below to code the solution:

3.2.1 Button [3.2.1 - Click to continue]

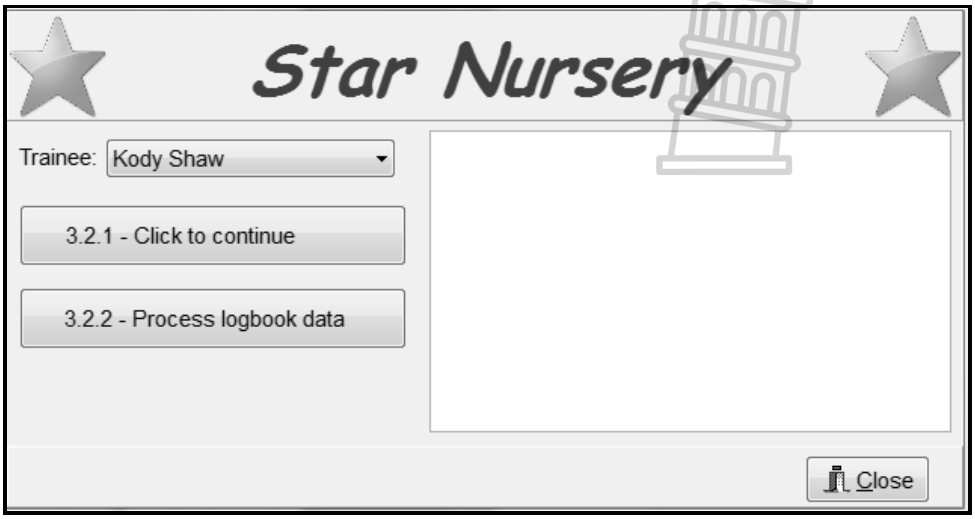
The user must select the name of a trainee from the **cmbTrainee** combo box. Code is provided to do the following:

- Extract the name from the combo box.
- Assign an ID to the selected trainee using the variables **sName** and **iTraineeID**.

Write code to do the following:

- Use the provided variables (**sName** and **iTraineeID**) and instantiate the **objTrainee** object.
- Set the **btnQ3_2_2** button to be visible. (4)

Example of the GUI if Kody Shaw is selected as the trainee and the 'Click to continue' button is clicked.



3.2.2 **Button [3.2.2 – Process logbook data]**

Write code to do the following:

- Check whether or not the **Logbook.txt** text file exists. If the file does NOT exist, display a suitable message and close the program.

- If the text file exists, search in each line of text for the selected trainee's ID.

  If the trainee's ID is found, do the following:

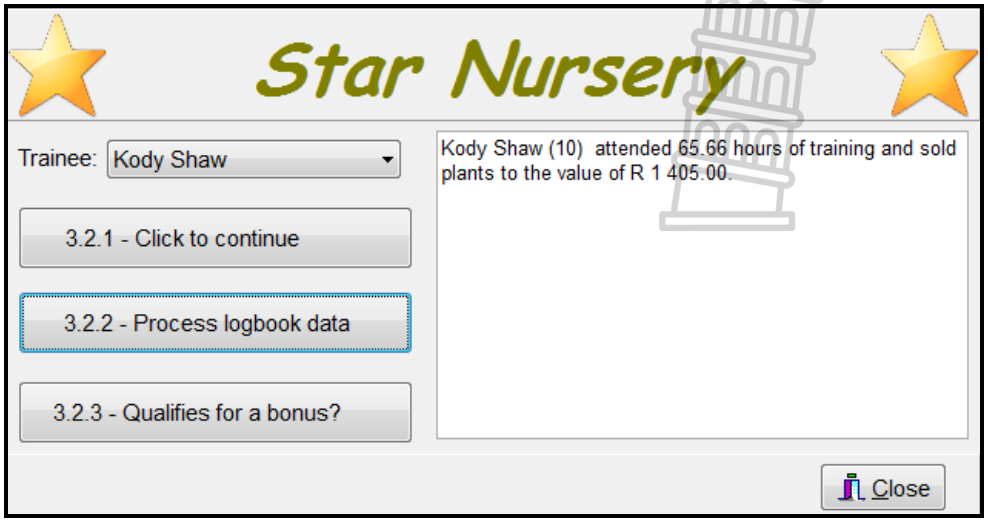
  - Extract the entry type (T or S) and the value indicating the number of hours of training or the amount of the sale from the line of text.
  - Use the **updateHours** or **updateSales** method to update the trainee's hours or sales amount based on the type of entry (T or S).

  Once all the data from the text file has been processed, do the following:

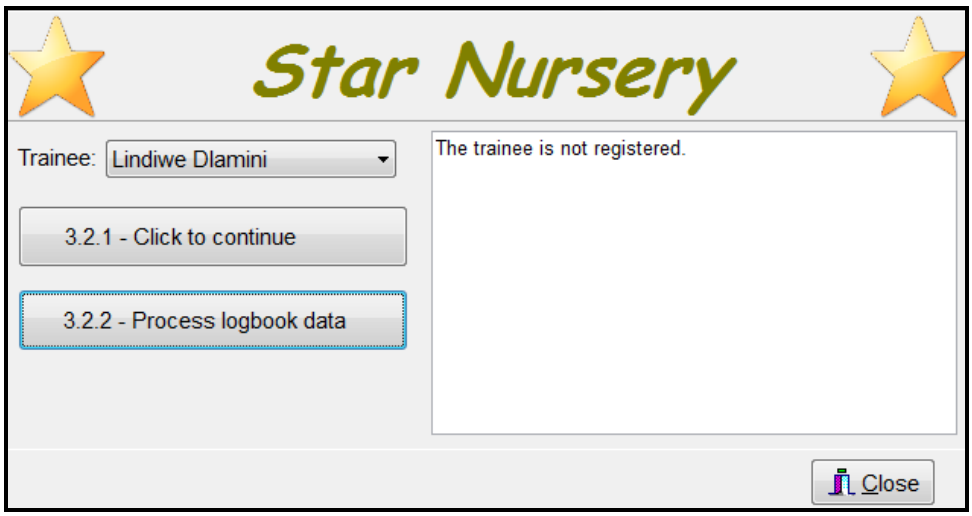
  - Clear the output area.
  - Display the information of the trainee in the output area using the **toString** method.
  - Set the **btnQ3_2_3** button to be visible.

  If the trainee's ID number could NOT be found in the text file, display the message 'The trainee is not registered.' in the output area.

Example of output if trainee Kody Shaw is selected and the logbook data is processed:



Example of output if trainee Lindiwe Dlamini is selected and the logbook data is processed:



(18)

3.2.3 **Button [3.2.3 - Qualifies for a bonus?]**

Write code to do the following:

- Determine if the trainee qualifies for a bonus using the **qualifiesForBonus** method.
- Display a suitable message in the output area indicating whether or not the trainee qualifies for a bonus.

Example of output for trainee Kody Shaw:

```
Kody Shaw (10) attended 65.66 hours of training and
sold plants to the value of R 1 405.00.
The trainee qualifies for a bonus.
```

Example of output for trainee Tyrone Kemsley:

```
Tyrone Kemsley (12) attended 65.66 hours of training
and sold plants to the value of R 1 078.00.
The trainee does NOT qualify for a bonus.
```

(3)

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

TOTAL SECTION C: 40

SECTION D

QUESTION 4: PROBLEM-SOLVING PROGRAMMING

SCENARIO

The trees at the nursery are categorised using the types Citrus, Deciduous, Nut and Tropical.

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.

The GUI shown below is displayed.

The program contains code of the following for the declaration:

- An array named **arrTypes** that contains the four types of trees at the nursery:

```
arrTypes: array [1..4] of String = (
                                   'Citrus',
                                   'Deciduous',
                                   'Nuts',
                                   'Tropical');
```

- A one-dimensional array named **arrList** that contains the list of 24 trees that are available at the nursery.

The format of the code in the array that represents each tree is as follows:

<tree name>#<first letter of tree type>

```
arrList: array[1..24] of String = (
         'Orange#C','Hazelnut#N','Apple#D',
         'Banana#T','Pecan#N','Pear#D',
         'Lemon#C','Papaya#T','Kiwi#T',
         'Apricot#D','Grapefruit#C','Walnut#N',
         'Lime#C','Mango#T','Peach#D',
         'Cashew#N','Almond#N','Tangerine#C',
         'Avocado#T','Cherry#D','Plum#D',
         'Macadamia#N','Kumquat#C','Guava#T');
```

The first two entries in **arrList** array can be explained as follows:

Orange#C means the tree name is Orange and the tree type is Citrus.
Hazelnut#N means the tree name is Hazelnut and the tree type is Nuts.

- A two-dimensional array called **arrTrees** and two integer variables that contains the number of rows and columns of the **arrTrees** array:

```
arrTrees: array [1..4, 1..6] of String;
iTypes: integer = 4;
iNum: integer = 6;
```

NOTE:

- Do NOT change the code provided.
- The use of good programming techniques and modular design must be applied in your solution.

Complete the code for EACH section of QUESTION 4, as described in QUESTION 4.1 to QUESTION 4.3 below.

4.1 **Button [4.1 - Separate by type]**

The data in the **arrList** array must be stored in the two-dimensional array **arrTrees** by row according to the types of trees, e.g. the data of the citrus trees must be stored in row 1, the deciduous trees in row 2, the nut trees in row 3 and the tropical trees in row 4.

Use the **arrTypes** array to determine the row value where the data of the tree must be stored in the **arrTrees** array, e.g. if the data string is 'Orange#C', then the tree is of type 'Citrus' and the data string 'Orange#C' must be stored in row 1. If the data string is 'Hazelnut#N', then the tree is of type 'Nuts' and the data string must be stored in row 3, and so on.

Write code to do the following:

- Store the data of each tree in the **arrList** array in the correct position in the two-dimensional array **arrTrees**.
- Enable buttons **btnQ4_2** and **btnQ4_3**. (11)

4.2 **Button [4.2 - Display]**

Write code to display the types of trees contained in the **arrTypes** array and the trees corresponding with the types of trees from the **arrTrees** array.

Example of output:

```
Citrus:       Orange#C    Lemon#C     Grapefruit#C Lime#C      Tangerine#C  Kumquat#C
Deciduous:    Apple#D     Pear#D      Apricot#D    Peach#D     Cherry#D     Plum#D
Nuts:         Hazelnut#N  Pecan#N     Walnut#N     Cashew#N    Almond#N     Macadamia#N
Tropical:     Banana#T    Papaya#T    Kiwi#T       Mango#T     Avocado#T    Guava#T
```

(7)

4.3 **Button [4.3 - Sort alphabetically]**

The trees in the **arrTrees** array must be sorted alphabetically as per tree type, e.g. citrus trees will appear as follows in row 1 of the **arrTrees** array:

```
Grapefruit  Kumquat  Lemon  Lime  Orange  Tangerine
```

Write code to do the following:

- Delete the last two characters of each element in the **arrTrees** array.
- Sort the **arrTrees** array alphabetical per tree type.
- Execute the code in the button **btnQ4_2** to display the list of trees after it has been sorted.

Example of output:

```
Citrus:       Grapefruit  Kumquat   Lemon     Lime      Orange    Tangerine
Deciduous:    Apple       Apricot   Cherry    Peach     Pear      Plum
Nuts:         Almond      Cashew    Hazelnut  Macadamia Pecan     Walnut
Tropical:     Avocado     Banana    Guava     Kiwi      Mango     Papaya
```

(12)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION D: 30
GRAND TOTAL: 150

INFORMATION TECHNOLOGY

DATABASE INFORMATION FOR QUESTION 2:

The design of the database tables is as follows:

Table: **tblPlants**

This table contains a price list of all the stock in PJB Wholesale Garden Centre.

Field name	Data type	Description
PlantCode	Text (13)	A unique code assigned to each plant item
Description	Text (35)	The description of each plant item
Category	Text (12)	Category of the plant item
SizeOfPot	Text (4)	The pot size of the plant indicated with S, M, L, XL, XXL and XXXL
Colour	Text (20)	Colour of the flowers
Price	Currency	Selling price of the plant
InStock	Integer	Number of plants in stock

Example of data of the first ten records:

PlantCode	Description	Category	SizeOfPot	Colour	Price	InStock
ALS027#L	Alstroemeria Intic Magic White	Shrub	L	White	R 61.90	14
ALS028#L	Alstroemeria Intic Moon Light	Shrub	L	Deep plum	R 61.90	43
AMA004#L	Amaryllis Groot Flower	Flower	L	Red	R 35.50	196
AMA004#M	Amaryllis Groot Flower	Flower	M	Red	R 30.00	31
AMA004#XL	Amaryllis Groot Flower	Flower	XL	Red	R 30.00	28
AMA004#XXL	Amaryllis Groot Flower	Flower	XXL	Red	R 49.00	95
AMA005#L	Amaryllis Belladonna Lily	Flower	L	Pink	R 41.85	12
AMA005#M	Amaryllis Belladonna Lily	Flower	M	Pink	R 30.00	34
AMA005#S	Amaryllis Belladonna Lily	Flower	S	Pink	R 25.00	93
AMA005#XL	Amaryllis Belladonna Lily	Flower	XL	Pink	R 30.00	158

Table: **tblOrders**

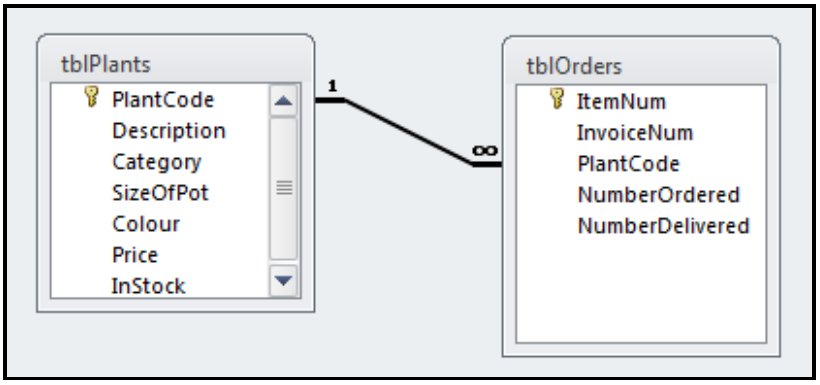
This table contains the records of orders that were placed.

Field name	Data type	Description
ItemNum	AutoNumber	A unique number assigned to an item in an order
InvoiceNum	Text (3)	Every order receives an invoice number. An order may contain multiple items.
PlantCode	Text (13)	The code of the plant ordered – foreign key
NumberOrdered	Integer	Number of plants of a specific item ordered
NumberDelivered	Integer	Number of plants of a specific item delivered

Example of data of the first ten records:

ItemNum	InvoiceNum	PlantCode	NumberOrdered	NumberDelivered
1	F1	ROS003#XXL	25	25
2	F1	ROS004#XXL	15	15
3	F1	ROS005#XXL	10	0
4	F2	ARM003#S	46	46
5	F2	ARM004#S	26	26
6	F2	CYR001#S	85	45
7	F2	CYR001#XL	44	44
8	D2	ARG006#S	70	70
9	D2	CUP001#XL	36	36
10	D2	CUP001#M	25	25

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

2019

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 23 pages.

GENERAL INFORMATION:

- These marking guidelines must 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 question paper were not followed or the requirements of the question were not met.

- **ANNEXURES A, B, C and D** (pages 3–9) include the marking grid for each question and a table for a summary of the learner's marks.

- **ANNEXURES E, F, G and H** (pages 10–23) contain examples of a programming solution for QUESTION 1 to QUESTION 4 in programming code.

- Copies of **ANNEXURES A, B, C, D** and the **summary of learner's marks** (pages 3–9) should be made for each learner and completed during the marking session.

ANNEXURE A

SECTION A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
A learner must be penalised only once if the same error is repeated.			
1.1	**Button [1.1 - Random number]** Random value generated✓ In correct range: Lower limit✓upper limit✓ Display the random number converted to a string✓	**4**	
1.2	**Button [1.2 - Calculate minutes]** Correct use of variables ✓ Extract the number of participants ✓ Calculate the number of minutes: Test if number of participants <=20✓ Set participant time = 2.5 ✓ Else ✓ if number of participants <=50 ✓ *or: Test if number of participants > 20 AND <= 50* Set participant time = 2.3 ✓ Else *or: Test if number of participants > 50* Set participant time = 2 ✓ Number of minutes= number of participants * participant time ✓ Display number of minutes ✓ to two decimal places ✓ Display number of minutes ✓ rounded to the next minute✓	**13**	
1.3	**Button [1.3 - Calculate factorial]** Extract the number from the spin edit ✓ Set factorial to 1✓ Loop ✓from 1 *(or 2)* to the number✓ Multiply factorial by the value of the loop variable✓ and assign the answer to factorial ✓ Display the factorial in the edit box✓	**7**	
1.4	**Button [1.4 - Reverse words]** Extract the sentence ✓ Add a space to the end✓ Initialise variables for word✓ and new sentence✓ Loop ✓to the length of the sentence ✓ Extract character at index in sentence✓ Test if character✓ = space✓ Add word ✓ and space✓ to new sentence Set word to empty string✓ Else✓ Add character ✓ to front of word✓ Display the new sentence with reversed words✓	**16**	
	TOTAL SECTION A	**40**	

ANNEXURE B

SECTION B

QUESTION 2: MARKING GRID - DATABASE PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1.1	**Button [2.1.1 – List of roses]** **SQL**: `SELECT * FROM tblPlants WHERE Category = "Rose"` **Concepts**: SELECT all fields ✓ FROM Correct table ✓ WHERE Condition: Category = "Rose" ✓	**3**	

2.1.2	**Button [2.1.2 - Pink roses and flowers]**
	SQL:
	`SELECT PlantCode, Category, Colour, SizeOfPot`
	`FROM tblPlants`
	`WHERE`
	`(Category = "ROSE" OR Category = "FLOWER")`
	`AND Colour LIKE "%Pink%"`
	Concepts:
	SELECT all the correct fields ✓
	FROM correct table ✓
	WHERE
	Conditions: ROSE or FLOWER ✓
	brackets around both OR conditions ✓ AND ✓
	Colour LIKE %Pink% ✓
	
	Alternative:
	SELECT all the correct fields ✓
	FROM correct table ✓
	WHERE
	Conditions: Category IN ✓ ("ROSE", "FLOWER") ✓
	AND ✓
	Colour LIKE %Pink% ✓
		6	

2.1.3	**Button [2.1.3 – Average price per category]**
	`SELECT Category, Format(Avg(Price),"Currency")`
	`AS [AveragePrice]`
	`FROM tblPlants GROUP BY Category`
	Concepts:
	SELECT Category ✓
	Average of price field ✓ as AveragePrice✓– Format as Currency✓
	FROM tblPlants
	GROUP BY category ✓
		5	

QUESTION 2: MARKING GRID – CONTINUE

2.1.4	**Button [2.1.4 – Display information for invoice number F2]** `SELECT InvoiceNum, Description, NumberOrdered FROM tblPlants, tblOrders WHERE tblPlants.PlantCode = tblOrders.PlantCode AND InvoiceNum = "F2";` **Concepts**: SELECT InvoiceNum, Description, NumberOrdered fields ✓ FROM both tables ✓ WHERE Conditions: Plantcode = plantcode✓ both table names ✓ AND InvoiceNum = "F2" ✓	**5**	
2.1.5	**Button [2.1.5 – Update item delivery]** `UPDATE tblOrders SET NumberDelivered = NumberOrdered WHERE ItemNum = QuotedStr(sItemNum);` **Concepts**: UPDATE correct table ✓SET✓ NumberDelivered = NumberOrdered ✓ WHERE ItemNum = variable (correct format) ✓	**4**	
	Subtotal: SQL	**[23]**	

2.2.1	**Button [2.2.1 – Check stock]**
	Move to first record✓
	Loop while not end of table ✓
	if (sPlantCode = tblPlants['PlantCode']) ✓
	if (iNumOrdered > tblPlants['InStock']) ✓
	cContinue = Must order be placed ('Y' or 'N') ✓
	if cContinue = 'Y'
	iNumOrdered = tblPlants['InStock'] ✓ else display 'order cancelled'
	if (iNumOrdered <= tblPlants['InStock']) ✓ OR
	order must be placed for available stock ✓
	Display output using correct fields from tblPlants in redDisplay✓
	btnQ2_2_2.Enabled ✓
	Move to next record ✓
		11	

2.2.2	**Button [2.2.2 – Place an order]** Change tblOrders to INSERT mode ✓ Set ['InvoiceNum] to F2 ✓ Set ['PlantCode'] to plant code ✓ Set ['NumberOrdered'] to number of plants ordered✓ Set ['NumberDelivered'] to 0 ✓ POST✓	**6**	
	Subtotal: Delphi code	**[17]**	
	TOTAL SECTION B	**40**	

ANNEXURE C

SECTION C

QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.1(a)	**updateHours method** Declare method ✓ with parameter of correct data type ✓ Increment the fHours attribute ✓ with parameter value ✓	**4**	
3.1.1(b)	**updateSales method** Declare a method with parameter of correct data type ✓ Increment the fSales attribute with parameter value ✓	**2**	
3.1.2	**qualifiesForBonus method** Declare a method that returns a Boolean value ✓ Test if (hours >= 15)✓ AND ✓ (sales >= 1200)✓ Assign TRUE to the result of the method Else Assign FALSE to the result of the method ✓ *Alternative:* Initialise a Boolean return variable to FALSE If test is positive assign TRUE to return variable Assign return variable to result of the method	**5**	
3.1.3	**toString method** Adding the relevant attributes (TraineeID, Hours and Sales) ✓ to the concatenated string ✓ Convert the numerical values to string ✓ Formatting the sales value to currency to two decimal places ✓	**4**	
	Subtotal: Object class	**[15]**	

QUESTION 3: MARKING GRID – CONTINUE

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 – Click to continue]** *Instantiate the objTrainee:* objTrainee✓:= TTrainee.Create✓(parameters in correct order and data type ✓) *{object variable^ := Class using create method^ (parameters^)}* Show the btnQ3_2_2 button using visible property or show method ✓	**4**	
3.2.2	**Button [3.2.2 – Process logbook data]** Test if text file exists ✓ If file does not exists then show message and close program ✓ If the file exists: Initialise a Boolean flag (found) to false ✓ AssignFile and Reset to read from file ✓ Use a loop to read up to the end of the file ✓ Read a line of text ✓ Determine if the entry ✓is related to selected trainee using the getTraineeID method ✓ Split the line of text into the entry type (T/S) and the numerical value (copy/pos/delete/ sLine[1]) ✓✓✓ Test (using IF/CASE-statement) for Tor S ✓call the relevant method with argument ✓✓ Change Boolean flag to true (found)✓ Once all the entries were process – end of file reached: If at least one trainee ID was found use toString method ✓ to display object data set btnQ2_2_3 button to visible ✓ If trainee ID was not found display a suitable message✓	**18**	
3.2.3	**Button [3.2.3 – Qualify for a bonus?]** Test by using the qualifiesForBonus method ✓ to determine if trainee qualifies for bonus ✓ Display suitable messages ✓	**3**	
	Subtotal: Form class	**[25]**	
	TOTAL SECTION C	**40**	

ANNEXURE D

SECTION D

QUESTION 4: MARKING GRID – PROBLEM SOLVING

Question	DESCRIPTION	MAX MARKS	LEARNER'S MARKS
4.1	**Button [4.1 – Separate by type]** Loop row from 1 to iTypes ✓ Set column to 0 ✓ Loop index from 1 to 24 ✓ Test if last character ✓of arrList[index] ✓ = first character ✓ of arrTypes[row] ✓ Increment column value✓ arrTrees[row,column] ✓ = arrList[index] ✓ Enable buttons btnQ4_2 and btnQ4_3 ✓	**11**	
4.2	**Button [4.2 – Display]** Loop row from 1 to iTypes ✓ Set string variable sLine to arrTypes[row] +':' ✓ Loop col from 1 to iNum ✓ Join arrTrees[row,col] ✓to the string variable sLine + space ✓ Display the string variable in the output area ✓ Loops correctly nested ✓	**7**	
4.3	**Button [4.3 – Sort alphabetically]** Loop row from 1 to iTypes ✓ Loop col from 1 to iNum ✓ Delete ✓ the last two characters✓ from arrTrees[row,col] ✓ Loop col from 1 to iNum - 1 ✓ Loop c from col + 1 to iNum ✓ Test if arrTrees[row,col] > arrTrees[row,c] ✓ Set temp = arrTrees[row,col] ✓ Set arrTrees[row,col] = arrTrees[row,c] ✓ Set arrTrees[row,c] = temp ✓ Execute code in button btnQ4_2 ✓	**12**	
	TOTAL SECTION D	**30**	

SUMMARY OF LEARNER'S MARKS:

CENTRE NUMBER:		**EXAMINATION NUMBER:**			
	SECTION A	**SECTION B**	**SECTION C**	**SECTION D**	
	QUESTION 1	**QUESTION 2**	**QUESTION 3**	**QUESTION 4**	**GRAND TOTAL**
MAX. MARKS	**40**	**40**	**40**	**30**	**150**
LEARNER'S MARKS					

ANNEXURE E: SOLUTION FOR QUESTION 1

```
unit Question1_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, ExtCtrls, Math, Grids, DBGrids;
type
  TfrmQuestion1 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    btnQ1_2: TButton;
    edtMinsRounded: TEdit;
    edtSentence: TEdit;
    btnQ1_4: TButton;
    edtMinutes: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    GroupBox4: TGroupBox;
    Label4: TLabel;
    edtReverse: TEdit;
    Label5: TLabel;
    btnQ1_3: TButton;
    edtFactorial: TEdit;
    btnQ1_1: TButton;
    edtRandomNumber: TEdit;
    spnNumber: TSpinEdit;
    edtParticipants: TEdit;
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion1: TfrmQuestion1;

implementation

{$R *.dfm}
// ============================================================================
// Question 1.1            4 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  edtRandomNumber.Text := IntToStr(Random(21) + 100);
 //edtRandomNumber.Text := IntToStr(RandomRange(100,121));
end;
```

```
// ============================================================================
// Question 1.2            13 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  iNum: integer;
  rMinutes, rHours, rFrac, rParticipantTime: real;
begin
  iNum := StrToInt(edtParticipants.Text);
  if iNum <= 20 then
    rParticipantTime := 2.5;

  if (iNum > 20) AND (iNum <= 50) then
     rParticipantTime := 2.3;

  if iNum > 50 then
     rParticipantTime := 2.0;

  rMinutes := iNum * rParticipantTime;
  edtMinutes.Text := FloatToStrf(rMinutes,ffFixed,6,2);
  edtMinsRounded.Text := IntToStr(Ceil(rMinutes));
end;
```

```
// Alternative:
  if iNum <= 20 then
    rParticipantTime := 2.5
  else
   if iNum <= 50 then
     rParticipantTime := 2.3
   else
     rParticipantTime := 2.0;
```

```
// ============================================================================
// Question 1.3            7 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  iNum, I, iFactorial: integer;
begin
  iNum := spnNumber.Value;
  iFactorial := 1;
  for I := 2 to iNum do
    iFactorial := iFactorial * I;
  edtFactorial.Text := IntToStr(iFactorial);
end;
// ============================================================================
// Question 1.4            16 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sString, sWord, sReverseString: String;
  iPos, I: integer;
begin
  sString := edtSentence.Text + ' ';
  sWord := '';
  sReverseString := '';
  for I := 1 to length(sString) do
  begin
    if sString[I] = ' ' then
    begin
      sReverseString := sReverseString + sWord + ' ';
      sWord := '';
    end
    else
    begin
      sWord := sString[I] + sWord;
    end;
  end;
  edtReverse.Text := sReverseString;
end;
end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;
// ---  Delphi and Database programming  ---
//
// Possible solution for Question 2.
// ---------------------------------------
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, ConnectDB_U, DB, ADODB, Grids,
  DBGrids, ComCtrls, DateUtils, DBCtrls;
type
  TfrmDBQuestion2 = class(TForm)
    pnlBtns: TPanel;
    bmbClose: TBitBtn;
    bmbRestoreDB: TBitBtn;
    grpTblOrders: TGroupBox;
    grpTblPlants: TGroupBox;
    dbgPlants: TDBGrid;
    dbgOrders: TDBGrid;
    tabsQ2_2ADO: TTabSheet;
    tabsQ2_1SQL: TTabSheet;
    redDisplay: TRichEdit;
    grpResults: TGroupBox;
    dbgrdSQL: TDBGrid;
    grpOutput: TGroupBox;
    pgcTabs: TPageControl;
    pnlTables: TPanel;
    btnQ2_1_1: TButton;
    btnQ2_1_3: TButton;
    btnQ2_1_2: TButton;
    btnQ2_1_4: TButton;
    btnQ2_2_1: TButton;
    btnQ2_2_2: TButton;
    GroupBox1: TGroupBox;
    cmbPlantCode: TComboBox;
    Label1: TLabel;
    Category: TLabel;
    cmbCategory: TComboBox;
    Label2: TLabel;
    edtNumPlants: TEdit;
    btnQ2_1_5: TButton;
    procedure bmbRestoreDBClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure cmbCategoryChange(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
  private
  public
  end;

var
  frmDBQuestion2: TfrmDBQuestion2;
  dbCONN: TConnection;
```

```
  // --- Provided global variables to be used ---
  tblPlants, tblOrders: TADOTable;

  // To be used in Question 2.2.1 and 2.2.2
  iNumOrdered: integer;
  sPlantCode: String;

implementation

{$R *.dfm}
{$R+}
// Question 2.1 - SQL SECTION

// ======================================================
// Question 2.1.1              3 marks
// ======================================================
procedure TfrmDBQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
  // Question 2.1.1
  sSQL1 := 'SELECT * FROM tblPlants WHERE Category = "Rose"';

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

// ======================================================
// Question 2.1.2              6 marks
// ======================================================
procedure TfrmDBQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
begin
  // Question 2.1.2
  sSQL2 :=
    'SELECT PlantCode, Category, Colour, SizeOfPot FROM tblPlants WHERE '
    + ' (Category = "Rose" OR Category = "Flower") AND (Colour LIKE "%Pink%")';

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

// ======================================================
// Question 2.1.3              5 marks
// ======================================================
procedure TfrmDBQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3: String;
begin
  // Question 2.1.3
  sSQL3 := 'SELECT Category, Format(Avg(Price),"Currency") AS AveragePrice' +
    ' FROM tblPlants GROUP BY Category';

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

```
// ====================================================
// Question 2.1.4              5 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
begin
  // Question 2.1.4
  sSQL4 :=
    'SELECT InvoiceNum, Description,  NumberOrdered FROM tblPlants, tblOrders '
    + ' WHERE tblPlants.PlantCode = tblOrders.PlantCode AND InvoiceNum = "F2"';

  // Provided code - do not change
  dbCONN.runSQL(sSQL4);
end;
// ====================================================
// Question 2.1.5              4 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5, sItemNum : String;
begin
     sItemNum := InputBox('Input','Item number: ','');
  // Question 2.1.5
    sSQL5 :=
    'UPDATE tblOrders SET NumberDelivered = NumberOrdered WHERE ItemNum = ' +
sItemNum;
  // Provided code - do not change
  dbCONN.executeSQL(sSQL5,dbgOrders);
end;
```

// Question 2.2 – Section with Delphi code

```
// ====================================================
// Question 2.2.1              11 marks
// ====================================================
procedure TfrmDBQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  cContinue : char;
begin
  // Provided code
  redDisplay.Clear;
  iNumOrdered := StrToInt(edtNumPlants.Text);
  sPlantCode := cmbPlantCode.Text;
  // ====================================================
  // Question 2.2.1
  cContinue := 'Y';
  tblPlants.First;
  while (NOT tblPlants.Eof) do
  begin
    if (sPlantCode = tblPlants['PlantCode']) then
    begin
     if (iNumOrdered > tblPlants['InStock'])  then
      begin
         cContinue := InputBox('Place order?','Not enough stock. Do you want to
place an order for ' + IntToStr (tblPlants['InStock']) + ' plants?
(Y/N)','Y')[1];
         if cContinue = 'Y' then
             iNumOrdered := tblPlants['InStock']
         else
           begin
```

```
            btnQ2_2_2.Enabled := false;
            redDisplay.Lines.Add('Order cancelled');
          end;
      end;
      if (iNumOrdered <= tblPlants['InStock']) OR (cContinue = 'Y') then
      begin
        redDisplay.Lines.Add('Plant code:    ' + tblPlants['PlantCode'] + #13 +
'Colour:    ' + tblPlants['Colour'] + #13 + 'Number ordered:    ' +
IntToStr(iNumOrdered) + #13 + 'Price per item:    ' +
FloatToStrF(tblPlants['Price'],ffCurrency,8,2));
        btnQ2_2_2.Enabled := true;
      end
    end;
    tblPlants.Next;
  end;
end;
// ======================================================
// Question 2.2.2              6 marks
// ======================================================
procedure TfrmDBQuestion2.btnQ2_2_2Click(Sender: TObject);
begin
  // Question 2.2.2
  tblOrders.Insert;
  tblOrders['InvoiceNum'] := 'F9';
  tblOrders['PlantCode'] := sPlantCode;
  tblOrders['NumberOrdered'] := iNumOrdered;
  tblOrders['NumberDelivered'] := 0;
  tblOrders.Post;
  showMessage('Order placed');
end;

// ============================================================
// Setup of database connections - DO NOT CHANGE!
// ============================================================
procedure TfrmDBQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  dbCONN.RestoreDatabase(dbgPlants, dbgOrders, dbgrdSQL);
  dbCONN.formatTables;
  redDisplay.Clear;
  tblPlants := dbCONN.tblOne;
  tblOrders := dbCONN.tblMany;
end;

procedure TfrmDBQuestion2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  dbCONN.dbDisconnect;
end;

procedure TfrmDBQuestion2.FormCreate(Sender: TObject);
begin
  CurrencyString := 'R';
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblPlants := dbCONN.tblOne;
  tblOrders := dbCONN.tblMany;
  dbCONN.setupGrids(dbgPlants, dbgOrders, dbgrdSQL);
  dbCONN.formatTables;
  pgcTabs.ActivePageIndex := 0;
  // Formatting display
  redDisplay.Clear;
  btnQ2_2_2.Enabled := false;
end;
```

```
procedure TfrmDBQuestion2.cmbCategoryChange(Sender: TObject);
begin
  cmbPlantCode.Clear;
  dbCONN.fillCombo(cmbCategory.Text);
  while NOT tblPlants.Eof do
  begin
    cmbPlantCode.Items.Add(tblPlants['PlantCode']);
    tblPlants.Next;
  end;
  tblPlants.Filtered := false;
  tblPlants.First;
end;


end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

OBJECT CLASS:

```
unit Trainee_U;
interface
// ============================================================================
// Provided code
// ============================================================================
type
   TTrainee = class(TObject)
    private
        fTraineeID: integer;
        fName: String;
        fHours,
        fSales: real;
     public
      constructor Create(iTraineeID: integer; sName: String);
      function toString: String;
      function getTraineeID: integer;
      function getName: String;
      procedure updateHours(rValue: real);
      procedure updateSales(rValue: real);
      function qualifiesForBonus: boolean;
   end;

implementation

uses SysUtils, DateUtils;
// ============================================================================
// Provided code
// ============================================================================
constructor TTrainee.Create(iTraineeID: integer; sName: String);
begin
   fTraineeID := iTraineeID;
   fName := sName;
   fHours := 0;
   fSales := 0;
end;

function TTrainee.getName: String;
begin
  Result := fName;
end;

function TTrainee.getTraineeID: integer;
begin
  Result := fTraineeID;
end;
// ============================================================================
// Question 3.1.1(a)        4 marks
// ============================================================================
procedure TTrainee.updateHours(rValue: real);
begin
  fHours := fHours + rValue;
end;
// ============================================================================
// Question 3.1.1(b)        2 marks
// ============================================================================
procedure TTrainee.updateSales(rValue: real);
begin
  fSales := fSales + rValue;
end;
```

```
// ============================================================================
// Question 3.1.2           5 marks
// ============================================================================
function TTrainee.qualifiesForBonus: boolean;
begin
  Result := (fHours >= 15) AND (fSales >= 1200);
end;
// ============================================================================
// Question 3.1.3           4 marks
// ============================================================================
function TTrainee.toString: String;
begin
  Result := fName + ' (' + IntToStr(fTraineeID) + ') ' +
            ' attended ' + FloatToStrF(fHours, ffFixed, 6, 2) +
            ' hours of training and ' +
            'sold plants to the value of ' +
            FloatToStrF(fSales, ffCurrency, 8, 2) + '.' ;
end;

end.
```

FORM UNIT

```
unit Question3_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls,  ComCtrls, Trainee_U, pngimage;
type
  TfrmQuestion3 = class(TForm)
    pnl1: TPanel;
    bmbClose: TBitBtn;
    pnlH: TPanel;
    imgLogo: TImage;
    imgL2: TImage;
    btnQ3_1_1: TBitBtn;
    redQ3: TRichEdit;
    btnQ3_2_2: TBitBtn;
    cmbTrainee: TComboBox;
    lbl1: TLabel;
    btnQ3_2_3: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure btnQ3_1_1Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
  public
  end;

var
  frmQuestion3: TfrmQuestion3;
  objTrainee : TTrainee;

implementation
{$R *.dfm}
{$R+}
//============================================================
```

```
procedure TfrmQuestion3.btnQ3_1_1Click(Sender: TObject);
var
   sName : String;
   iTraineeID : integer;
begin  //Question 3.2.1
 //Provided code - do not change!
 sName := cmbTrainee.Items[cmbTrainee.ItemIndex];
 iTraineeID := INTEGER(cmbTrainee.Items.Objects[
       cmbTrainee.Items.IndexOf(cmbTrainee.Items[cmbTrainee.ItemIndex])]);
// ============================================================================
// Question 3.2.1           4 marks
// ============================================================================
 objTrainee := TTrainee.Create(iTraineeID,sName);
 btnQ3_2_2.Visible := true; //OR  btnQ3_2_2.Show;
end;
// ============================================================================
// Question 3.2.2           18 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  TFile  : TextFile;
  sLine  : String;
  cType  : char;
  rValue : real;
  bFound : boolean;
  iIDx   : integer;
begin  //Question 3.2.2
  bFound := False;
  if NOT FileExists('Logbook.txt') then
   begin
     MessageDlg('File NOT found', mtInformation, [mbOK], 0);
     Exit;
   end;

  AssignFile(TFile, 'Logbook.txt');
  Reset(TFile);
  While not Eof(TFile) do
   begin
     Readln(TFile, sLine);
     iIDx := StrToInt(Copy(sLine, 1, Pos(';', sLine)-1));
     if (objTrainee.getTraineeID = iIDx) then
      begin
        Delete(sLine, 1, Pos(';', sLine));
        cType := sLine[1];
        Delete(sLine, 1, Pos('#', sLine));
        rValue := StrToFloat(sLine);
        case cType of
        'T' : objTrainee.updateHours(rValue);
        'S' : objTrainee.updateSales(rValue);
        end;
        bFound := True;
      end;
   end;
  CloseFile(TFile);
  redQ3.Clear;
  if bFound
  then
    begin
     redQ3.Lines.Add(objTrainee.toString);
     btnQ3_2_3.Visible := True;
    end
      else
```

```
     begin
       redQ3.Lines.Add('The trainee is not registered.');
       btnQ3_2_3.Visible := False;
     end;
end;

// ============================================================================
// Question 3.2.3           3 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
begin  //Question 3.2.3
   if objTrainee.qualifiesForBonus then
    begin
      redQ3.Lines.Add('The trainee qualifies for a bonus.')
    end
   else
    begin
      redQ3.Lines.Add('The trainee does NOT qualify for a bonus.');
    end;
end;
// ============================================================================
// Provided code
// ============================================================================
//{$REGION 'Provided code - do not change!'}
procedure TfrmQuestion3.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if Assigned(objTrainee) then
   begin
     objTrainee.Free;
   end;
end;

//-------------------------------------------------------------
procedure TfrmQuestion3.FormCreate(Sender: TObject);
begin
 //Add names and uniques numbers of trainees to combobox
 cmbTrainee.AddItem('Kody Shaw', TObject(10));
 cmbTrainee.AddItem('Luvuyo Bertola', TObject(11));
 cmbTrainee.AddItem('Tyrone Kemsley', TObject(12));
 cmbTrainee.AddItem('Craig Biggie', TObject(13));
 cmbTrainee.AddItem('Lindi Mahlati', TObject(14));
 cmbTrainee.AddItem('Sandy Brown', TObject(15));
 cmbTrainee.AddItem('Lindiwe Dlamini', TObject(16));
 //Ensure that the first trainee is selected.
 cmbTrainee.ItemIndex := 0;
end;

//{$ENDREGION}

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
// A possible solution for Question 4
unit Question4_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls;
type
  TfrmQuestion4 = class(TForm)
    redQ4: TRichEdit;
    btnQ4_1: TButton;
    btnQ4_2: TButton;
    btnQ4_3: TButton;
    procedure btnQ4_2Click(Sender: TObject);
    procedure btnQ4_1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btnQ4_3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion4: TfrmQuestion4;
  arrTypes: array [1..4] of String = (
    'Citrus',
    'Deciduous',
    'Nuts',
    'Tropical'
  );

  arrList: array [1..24] of String = (
    'Orange#C',
    'Hazelnut#N',
    'Apple#D',
    'Banana#S',
    'Pecan#N',
    'Pear#D',
    'Lemon#C',
    'Papaya#S',
    'Kiwi#S',
    'Apricot#D',
    'Grapefruit#C',
    'Walnut#N',
    'Lime#C',
    'Mango#S',
    'Peach#D',
    'Cashew#N',
    'Almond#N',
    'Tangerine#C',
    'Avocado#S',
    'Cherry#D',
    'Plum#D',
    'Macadamia#N',
    'Kumquat#C',
    'Guava#S'
  );
  arrTrees: array [1..4, 1..6] of String;
  iTypes: integer = 4;
  iNum: integer = 6;
```

```
implementation
{$R *.dfm}

// ============================================================================
// Question 4.1           11 marks
//=============================================================================
procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  i, r, c: integer;
begin
  redQ4.Clear;
  for r := 1 to iTypes do
  begin
    c := 0;
    for i := 1 to 24 do
      if arrList[i][length(arrList[i])] = arrTypes[r][1] then
      begin
        Inc(c);
        arrTrees[r, c] := arrList[i];
      end; // if
  end; // r
  btnQ4_2.Enabled := True;
  btnQ4_3.Enabled := True;
end;

// ============================================================================
// Question 4.2           7 marks
//=============================================================================
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  r, c: integer;
  sLine: String;
begin
  redQ4.Clear;

  for r := 1 to iTypes do
  begin
    sLine := arrTypes[r] + ':' +  #9 ;
    for c := 1 to iNum do
      sLine := sLine + arrTrees[r, c] + #9 ;
    redQ4.Lines.Add(sLine);
  end;
end;
```

```
// ============================================================================
// Question 4.3           12 marks
//============================================================================
procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  sTemp: String;
  i, j, c: integer;
begin
  for i := 1 to iTypes do
  begin
    for j := 1 to iNum do
      delete(arrTrees[i,j],length(arrTrees[i,j]) - 1,2);
    // end delete loop
    // sorting
    for j := 1 to iNum -1 do
      begin
      for c :=  j + 1 to iNum do
        if arrTrees[i,j] > arrTrees[i,c] then
        begin
           sTemp := arrTrees[i,j];
           arrTrees[i,j] := arrTrees[i,c];
           arrTrees[i,c] := sTemp;
        end;//if
      end; //j  for sorting
  end;//i
  btnQ4_2.Click;
end;

// ============================================================================
// Provided code
//============================================================================
procedure TfrmQuestion4.FormActivate(Sender: TObject);
begin
  redQ4.Paragraph.TabCount := 7;
  redQ4.Paragraph.Tab[0] := 20;
  redQ4.Paragraph.Tab[1] := 110;
  redQ4.Paragraph.Tab[2] := 200;
  redQ4.Paragraph.Tab[3] := 290;
  redQ4.Paragraph.Tab[4] := 380;
  redQ4.Paragraph.Tab[5] := 470;
  redQ4.Paragraph.Tab[6] := 570;
  btnQ4_2.Enabled := False;
  btnQ4_3.Enabled := False;
end;


end.
```