basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2025

MARKS: 150

TIME: 3 hours

This question paper consists of 31 pages, 2 data pages, 2 pages for planning and a separate information sheet.

INSTRUCTIONS AND INFORMATION

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

2. Two blank pages have been provided at the end of the question paper, which may be used for planning purposes.

3. An information sheet has been provided for you to complete at the end of the examination session. Ensure that ALL the information that you provided is correct and submit the information sheet before you leave the examination room.

4. 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.

5. This question paper is set with programming terms that are specific to the Delphi programming language. The Delphi programming language must be used to answer the questions.

6. 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.

7. 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.

8. 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.

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

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

11. 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.

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

13. If required, print the programming code of all the programs/classes that you completed. Your examination number must appear on all printouts. You will be given half an hour printing time after the examination session.

14. 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.

15. 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: **DataNov2025.exe**
    - Click on the 'Extract' button.
    - Enter the following password: **SuSTF@rm2025**

    Once extracted, the following list of files will be available in the folder **DataNov2025**:

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

Question 2:
ConnectDB_U.pas
FarmManagementDB - Copy.mdb
FarmManagementDB.mdb
MixedFarms.txt
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question 3:
Beehive_U.pas
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas

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

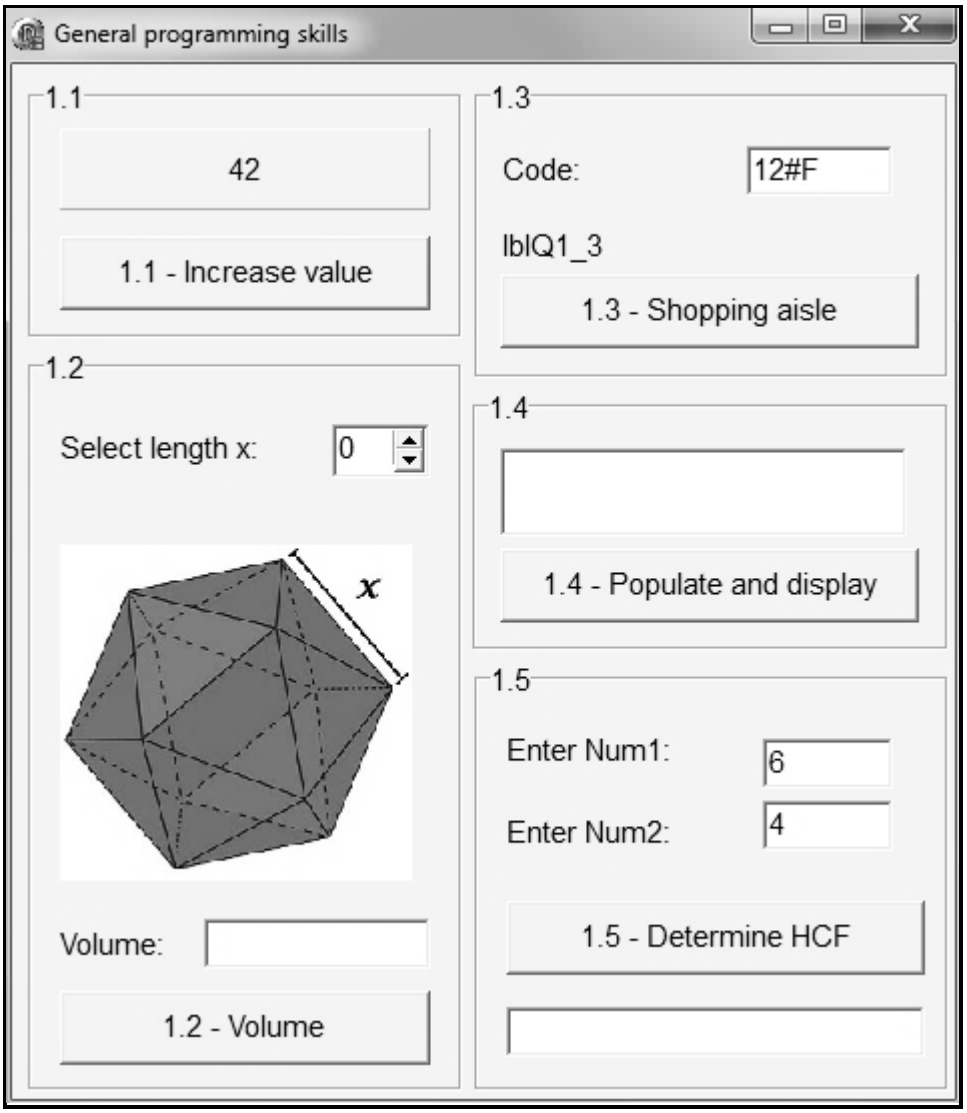
SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your examination number as a comment in the first line of the **Question1_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

Example of the graphical user interface (GUI):



- Complete the code for each section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.5.

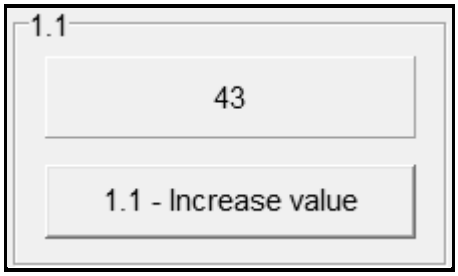
1.1 **Button [1.1 - Increase value]**

The panel **pnlQ1_1** currently displays an integer value that must be increased by the value of 1 when the button is clicked.

Write code to do the following:

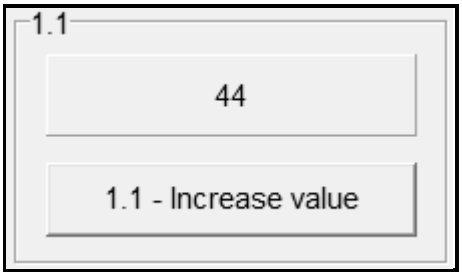
- Extract the integer value displayed on the panel **pnlQ1_1**.
- Increase the extracted value by the value of 1.
- Display the increased value on the panel **pnlQ1_1**.

Example of output after the button was clicked and the extracted value was increased by the value of 1:



NOTE: Each time the user clicks on the 'Increase value' button, the current value displayed on the panel should be increased by the value of 1.

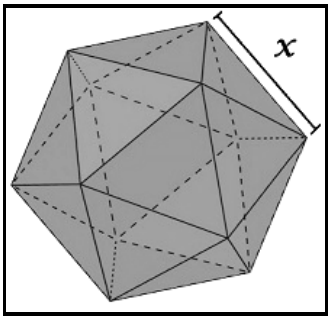
Example of output after the second click of the button:



(4)

1.2 **Button [1.2 - Volume]**

The image below shows a figure of a three-dimensional shape which has sides of equal length. The length of each side is represented by the letter x.



The formula to calculate the volume of the figure is as follows:

$$Volume = \frac{5(3+\sqrt{5})}{12}\ x^3$$

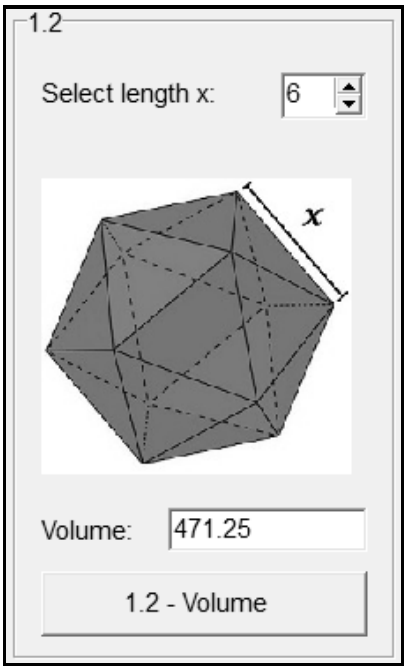
where x represents the length of the sides of the figure.

The user must select the length of x from the spin edit **spnQ1_2**.

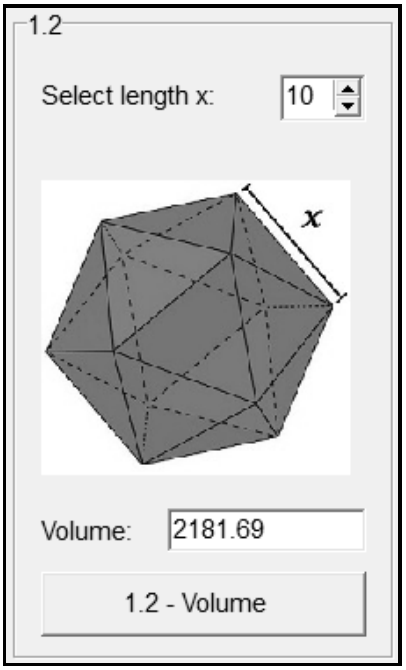
Write code to do the following:

- Extract the length of the side (x), from the spin edit **spnQ1_2**.
- Calculate the volume of the figure using the provided formula and at least TWO DIFFERENT pre-defined mathematical functions.
- Display the volume in the edit box **edtQ1_2**, formatted to TWO decimal places.

Example of output if the value of 6 was selected as the length of x:



Example of output if the value of 10 was selected as the length of x:



(6)

1.3 **Button [1.3 - Shopping aisle]**

A local grocery store uses a coding system to help customers find aisles with specific categories of food. Each category of food is represented by a single alphabet letter. The alphabet letters used and the categories of food that the letters represent are given in the table below.

ALPHABET LETTER	CATEGORY DESCRIPTION
F	Fruit and vegetables
D	Dairy
B	Butchery
S	Sauces
P	Pasta and rice

The user must enter a code in the edit box **edtQ1_3** in the following format:

```
<Aisle number>#<Alphabet letter>
```

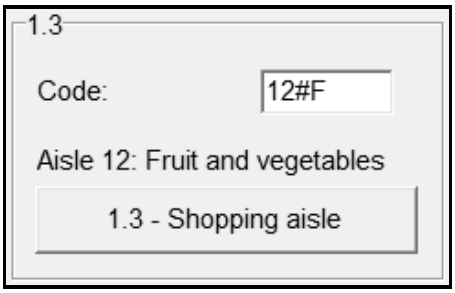
Example: 12#F refers to aisle 12 which stocks fruit and vegetables.

Write code to do the following:

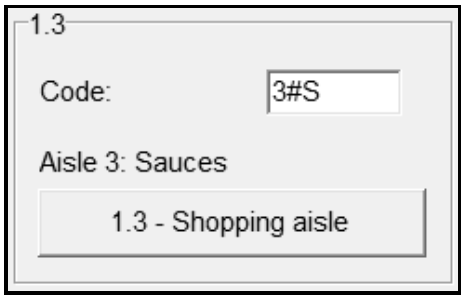
- Extract the code that was entered from edit box **edtQ1_3**.
- Separate the aisle number and the alphabet letter that represent the food category from the code that was entered.
- Use a **case statement** and the alphabet letter from the code, and save the correct category description in a variable.
- Display 'Aisle', the aisle number, a colon character (:), followed by a space and the correct category description on label **lblQ1_3** in the format:

```
Aisle <Aisle number>: <Category description>
```

Example of output if the code 12#F was entered:



Example of output if the code 3#S was entered:



NOTE: You may assume that the code entered is valid. (11)

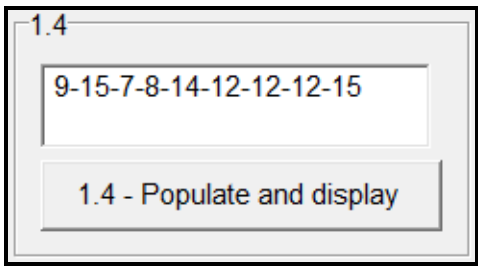
1.4 **Button [1.4 - Populate and display]**

The declaration of an array called **arrNumbers**, which consists of NINE elements of type integer, has been provided.

Write code to do the following:

- Assign a random number in the range from 1 to 15 (inclusive) to each element of array **arrNumbers**.
- Use the memo component **memQ1_4** to display the random numbers from array **arrNumbers** separated by dash (-) characters.

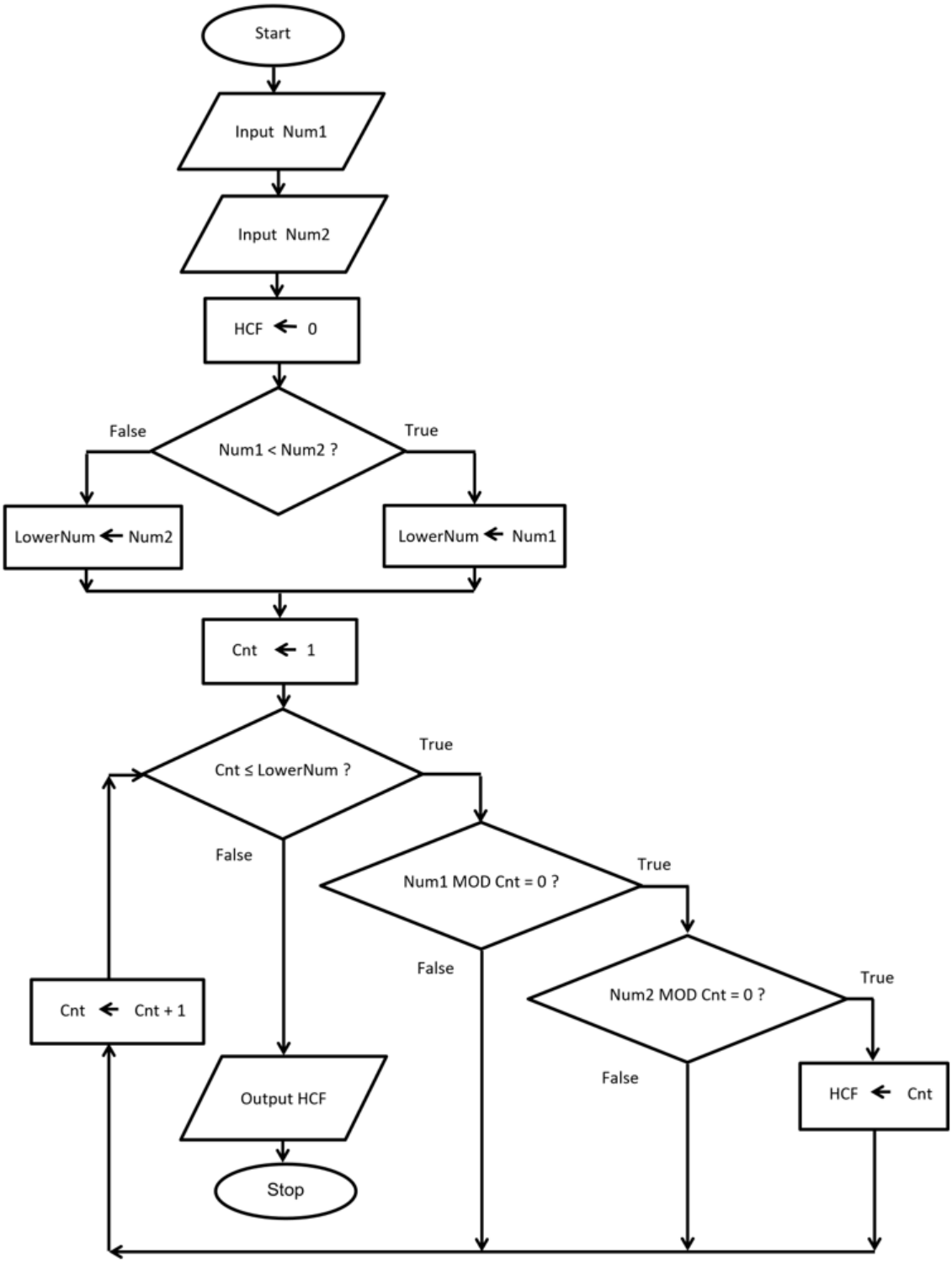
Example of output:



NOTE: The output displayed by your program may differ from the example of output as the numbers have been generated randomly. (8)

1.5 **Button [1.5 - Determine HCF]**

The flowchart below illustrates how to determine the highest common factor (HCF) of any TWO positive integers.



The user must enter integer values for Num1 and Num2.

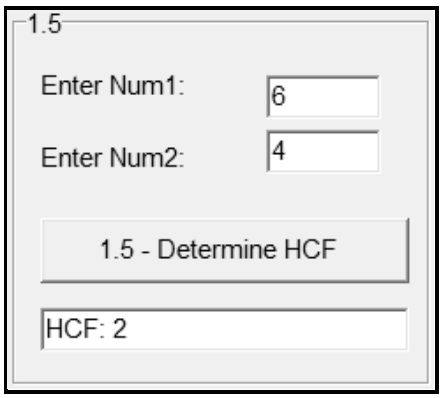
Code has been provided to:

- Extract and save the two integer values Num1 and Num2 in **iNum1** and **iNum2** respectively, and
- Initialise **iHCF** to zero (0)

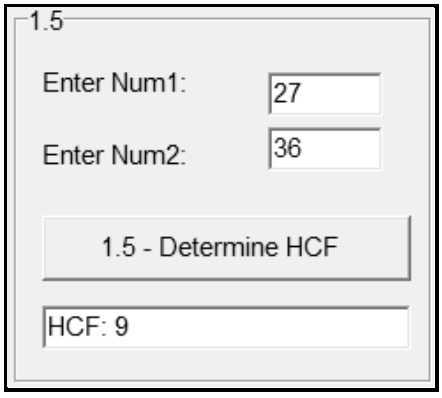
Write code to do the following:

- Code the steps provided in the flowchart to determine the HCF of the two numbers entered.
- Display the HCF in edit box **edtQ1_5**.

Example of output if the numbers 6 and 4 were entered:



Example of output if the numbers 27 and 36 were entered:



(11)

- 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: DATABASE PROGRAMMING

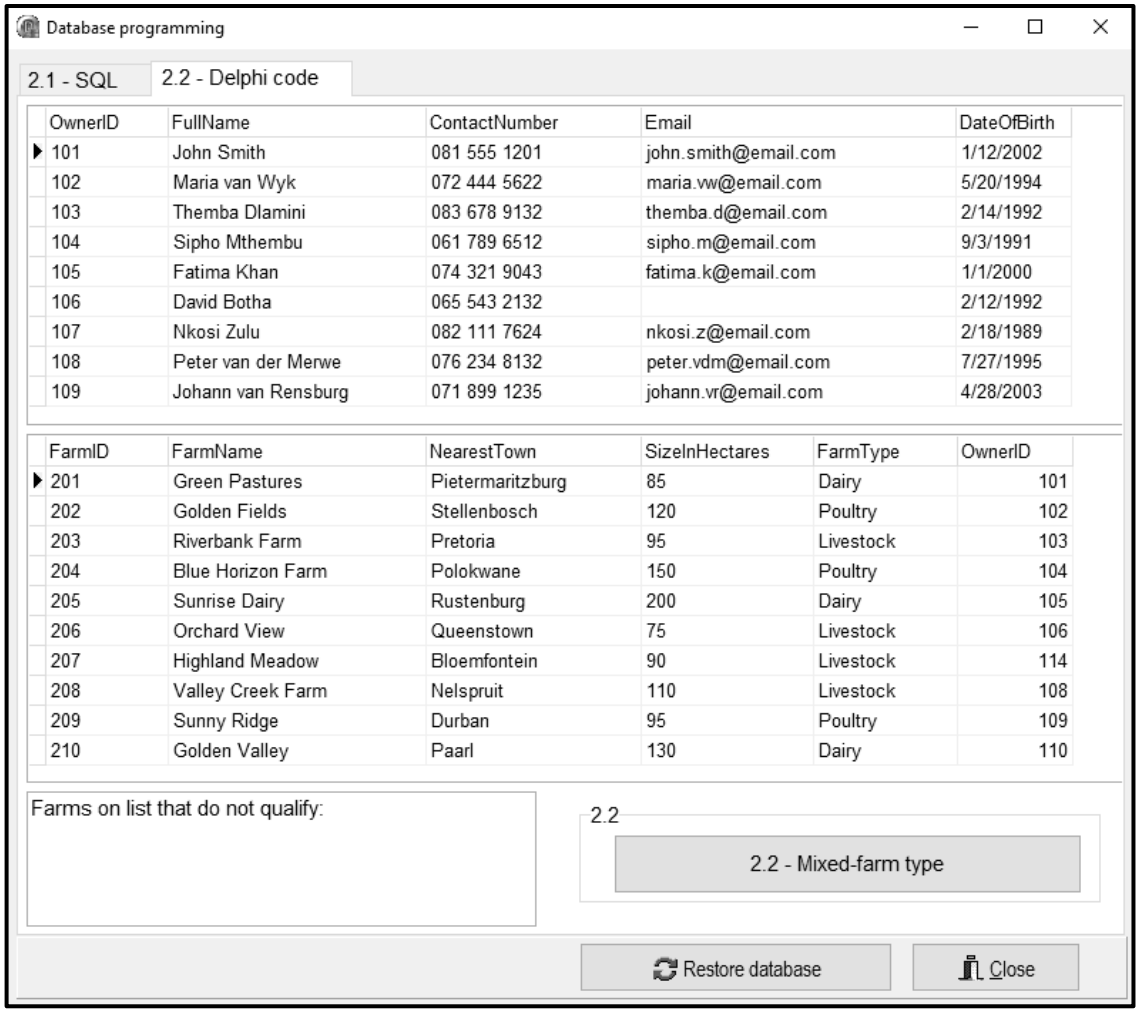
A service provider needs to manage a collection of small farms of various farm types, such as Poultry, Livestock and Dairy. Each farm is owned by only one (a single) farm owner, while one farm owner can own many farms.

A database called **FarmManagementDB.mdb** has been developed, which contains two tables called **tblFarmOwners** and **tblFarms**.

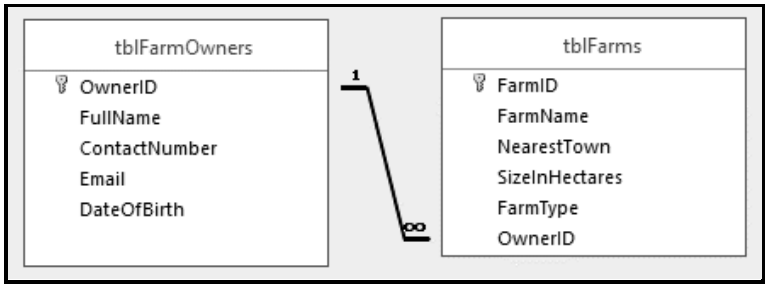
The data pages attached at the end of the question paper provide information on the design of the database and its contents.

Do the following:

- Open the incomplete project file called **Question2_P.dpr** in the **Question 2** folder.
- Add your examination number as a comment in the first line of the **Question2_U.pas** file.
- Compile and execute the program. The program has limited functionality currently. The contents of the tables are displayed, as shown below on the selection of tab sheet **2.2 - Delphi code**.



The relationship between the two tables **tblFarmOwners** and **tblFarms** is shown below.



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

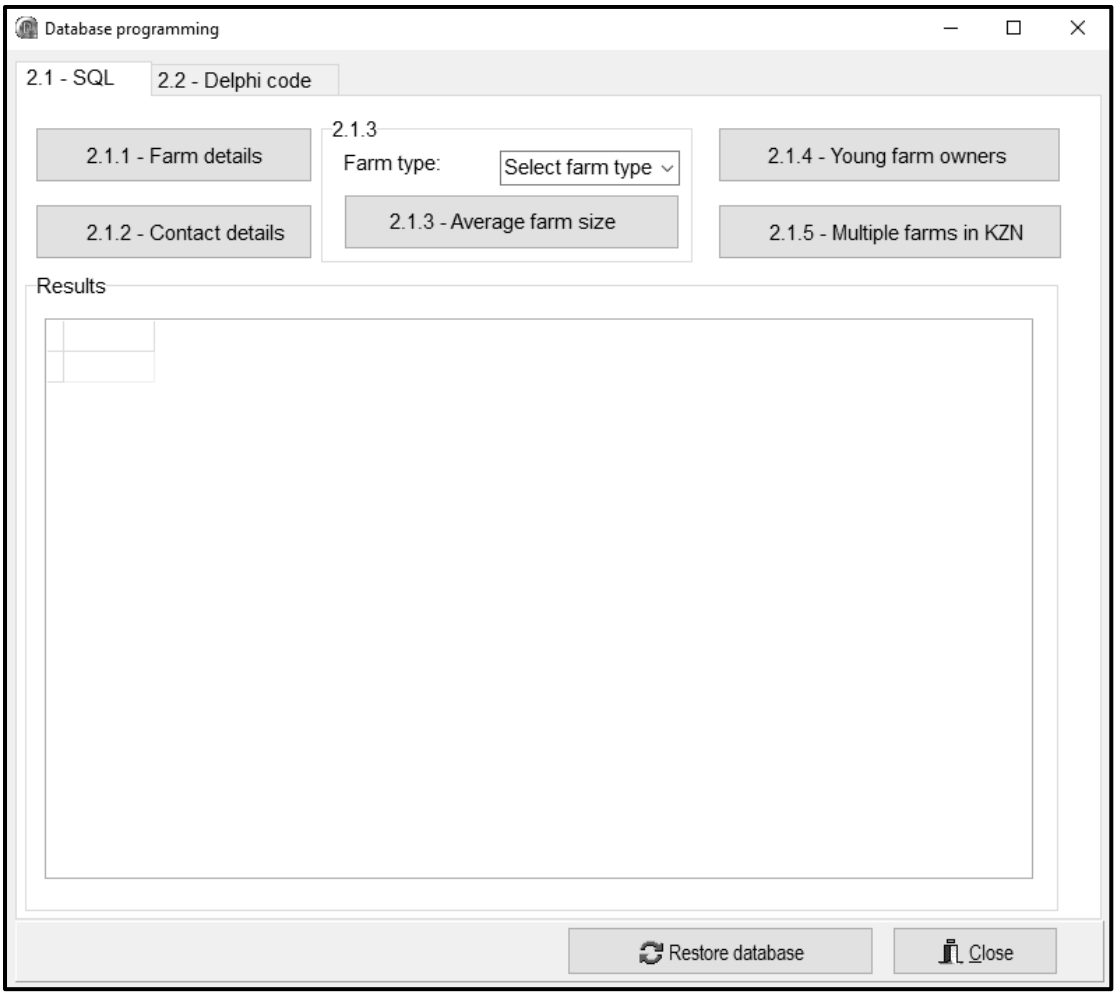
NOTE:

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

Variable	Data type	Description
`tblFarmOwners`	`TADOTable`	Refers to the table **tblFarmOwners**
`tblFarms`	`TADOTable`	Refers to the table **tblFarms**

2.1 Tab sheet [2.1 - SQL]

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



NOTE:

- Use ONLY SQL statements to answer QUESTION 2.1.1 to QUESTION 2.1.5.
- Code to execute the SQL statements and display the results of the queries has been 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 - Farm details]

Display the **FarmName**, **NearestTown** and **SizeInHectares** of all farms, sorted in descending order according to **SizeInHectares**.

Example of output of the first three records:

FarmName	NearestTown	SizeInHectares
Mountain Dew	Rustenburg	312
Blue Crane	Pietermaritzburg	212
Green Pastures	Potchefstroom	211

(3)

2.1.2 **Button [2.1.2 - Contact details]**

Display the **FullName** and **ContactNumber** of all farm owners whose e-mail addresses have not been saved in the table **tblFarmOwners**.

Example of output:

FullName	ContactNumber
David Botha	065 543 2132
Thabiso Tau	076 122 3458
Busi Nkosi	079 951 7536

(2)

2.1.3 **Button [2.1.3 - Average farm size]**

The average size of all the farms of the selected farm type must be calculated and saved in a new field called **AverageSize**.

Code has been provided to extract and save the selected farm type from the combo box **cmbQ2_1_3** in a variable **sFarmType**.

Display the **FarmType** and the **AverageSize**, formatted to TWO decimal places.

Example of output if Livestock was selected as the farm type:

FarmType	AverageSize
Livestock	137.80

(6)

2.1.4 **Button [2.1.4 - Young farm owners]**

Display the **FullName**, **DateOfBirth** and **Age** (calculated field) of all farm owners who are 25 years or younger. The current system date must be used to calculate the age as an integer value.

Use the formula below to calculate the age of the farm owners:

$$Age = \frac{Current\ system\ date - DateOfBirth}{365}$$

Example of output:

FullName	DateOfBirth	Age
John Smith	2002/01/12	23
Fatima Khan	2000/01/01	25
Johann van Rensburg	2003/04/28	22
Fred de Lange	2004/08/09	21

NOTE: The format of the date of birth may differ from this example due to the settings on your computer. (7)

2.1.5 **Button [2.1.5 - Multiple farms in KZN]**

Display the **FullName** and the total number of farms of each farm owner who owns more than one farm in the Durban or Pietermaritzburg area. The total number of farms must be saved in a new field called **TotalFarms**.

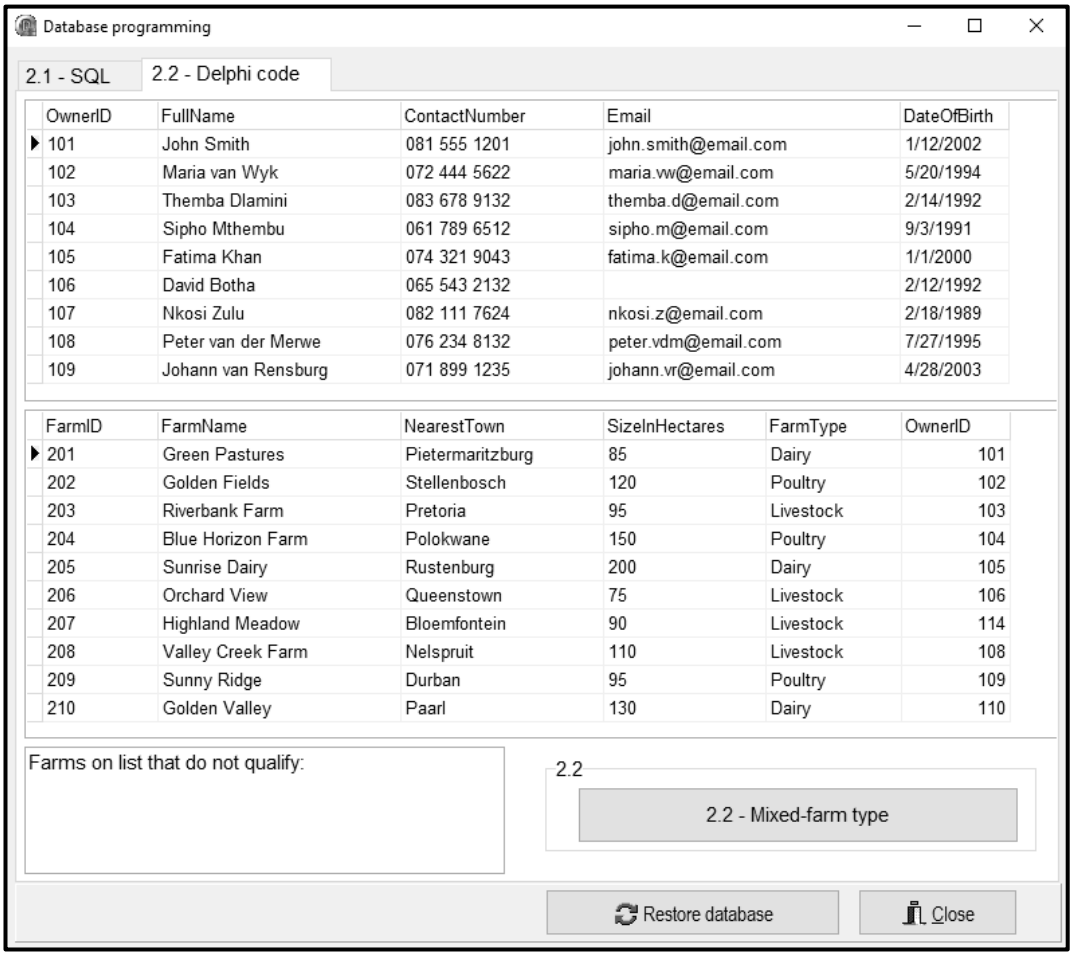
Example of output:

FullName	TotalFarms
Busi Nkosi	2
Johann van Rensburg	2
John Smith	3

(7)

2.2 Tab sheet [2.2 - Delphi code]

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



OwnerID	FullName	ContactNumber	Email	DateOfBirth
101	John Smith	081 555 1201	john.smith@email.com	1/12/2002
102	Maria van Wyk	072 444 5622	maria.vw@email.com	5/20/1994
103	Themba Dlamini	083 678 9132	themba.d@email.com	2/14/1992
104	Sipho Mthembu	061 789 6512	sipho.m@email.com	9/3/1991
105	Fatima Khan	074 321 9043	fatima.k@email.com	1/1/2000
106	David Botha	065 543 2132		2/12/1992
107	Nkosi Zulu	082 111 7624	nkosi.z@email.com	2/18/1989
108	Peter van der Merwe	076 234 8132	peter.vdm@email.com	7/27/1995
109	Johann van Rensburg	071 899 1235	johann.vr@email.com	4/28/2003

FarmID	FarmName	NearestTown	SizeInHectares	FarmType	OwnerID
201	Green Pastures	Pietermaritzburg	85	Dairy	101
202	Golden Fields	Stellenbosch	120	Poultry	102
203	Riverbank Farm	Pretoria	95	Livestock	103
204	Blue Horizon Farm	Polokwane	150	Poultry	104
205	Sunrise Dairy	Rustenburg	200	Dairy	105
206	Orchard View	Queenstown	75	Livestock	106
207	Highland Meadow	Bloemfontein	90	Livestock	114
208	Valley Creek Farm	Nelspruit	110	Livestock	108
209	Sunny Ridge	Durban	95	Poultry	109
210	Golden Valley	Paarl	130	Dairy	110

NOTE:

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

Button [2.2 – Mixed-farm type]

A mixed-farm type refers to a farm with a combination of different types of farming activities, such as poultry and dairy. Farmers will need to request to be changed to a mixed-farm type, if they qualify for it. Only farms larger than 100 hectares qualify to be a mixed-farm type.

A text file called **MixedFarms.txt**, which contains a list of **FarmID**s of the farms that could possibly qualify to be a mixed-farm type, has been provided.

The first THREE lines of text in the **MixedFarms.txt** text file are shown below.

```
201
204
206
```

The program must update the farm type to 'Mixed' if the farm qualifies to be a mixed farm. If the farm does NOT qualify, display the **FarmID**, **FarmName** and **SizeInHectares** of the farm in the rich edit **redQ2_2** component.

Code has been provided to clear the rich edit **redQ2_2** component and display a suitable heading.

Write code to do the following:

- Test if the text file **MixedFarms.txt** exists and display a suitable message using a ShowMessage dialog box if the text file does not exist.
- If the text file exists, read each **FarmID** from the text file.
  - Identify the farm in the **tblFarms** table.
  - Test if the farm is more than 100 hectares in size:
    - If true, update the **FarmType** field of the farm to 'Mixed'.
    - Otherwise, display the **FarmID**, **FarmName** and **SizeInHectares** of the farm in the rich edit **redQ2_2** component, in the format shown in the example on the next page.

Example of records in the **tblFarms** table before the **FarmType** was changed:

FarmID	FarmName	NearestTown	SizeInHectares	FarmType	OwnerID
201	Green Pastures	Pietermaritzburg	85	Dairy	101
202	Golden Fields	Stellenbosch	120	Poultry	102
203	Riverbank Farm	Pretoria	95	Livestock	103
204	Blue Horizon Farm	Polokwane	150	Poultry	104
205	Sunrise Dairy	Rustenburg	200	Dairy	105
206	Orchard View	Queenstown	75	Livestock	106
207	Highland Meadow	Bloemfontein	90	Livestock	114
208	Valley Creek Farm	Nelspruit	110	Livestock	108
209	Sunny Ridge	Durban	95	Poultry	109
210	Golden Valley	Paarl	130	Dairy	110

Example of records in the **tblFarms** table after the **FarmType** was changed:

FarmID	FarmName	NearestTown	SizeInHectares	FarmType	OwnerID
201	Green Pastures	Pietermaritzburg	85	Dairy	101
202	Golden Fields	Stellenbosch	120	Poultry	102
203	Riverbank Farm	Pretoria	95	Livestock	103
204	Blue Horizon Farm	Polokwane	150	Mixed	104
205	Sunrise Dairy	Rustenburg	200	Dairy	105
206	Orchard View	Queenstown	75	Livestock	106
207	Highland Meadow	Bloemfontein	90	Livestock	114
208	Valley Creek Farm	Nelspruit	110	Livestock	108
209	Sunny Ridge	Durban	95	Poultry	109
210	Golden Valley	Paarl	130	Dairy	110

Example of output of farms listed in the text file that do not qualify to be a mixed-farm type:

```
Farms on list that do not qualify:
201 - Green Pastures - 85 hectares
206 - Orchard View - 75 hectares
207 - Highland Meadow - 90 hectares
217 - Olifansberg - 99 hectares
```

(15)

- 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

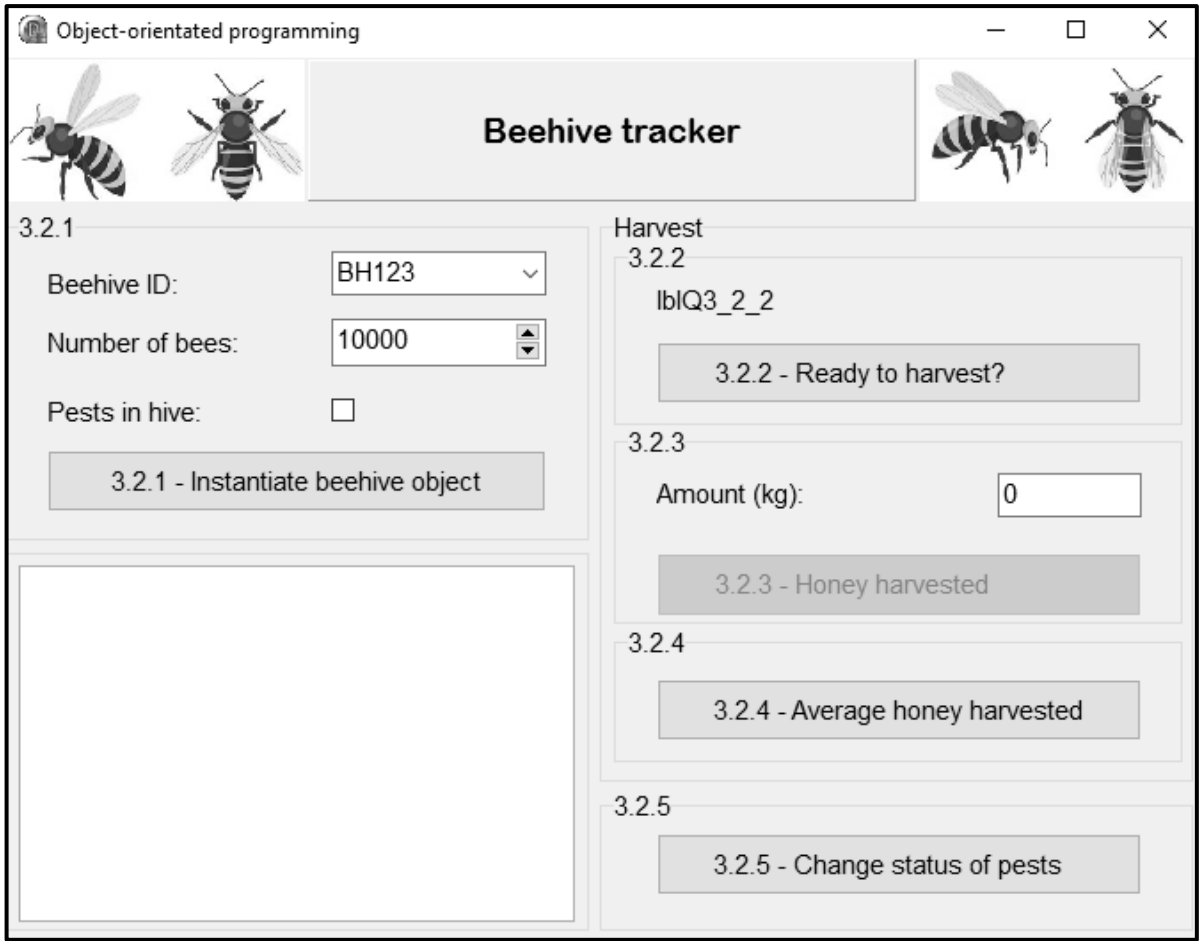
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

During warm seasons honey is regularly harvested from a beehive. A beehive harvesting tracker is used to monitor and check the health status of a beehive colony.

Do the following:

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

Example of the graphical user interface (GUI):



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

3.1 The provided incomplete object class (**TBeehive)** contains the declaration of six attributes, which describe a **Beehive** object.

The attributes of a **Beehive** object have been declared as follows:

Attribute	Type	Description
fBeehiveID	String	An ID used to identify the beehive
fBeeCount	Integer	The approximate number of bees in the beehive
fPests	Boolean	Indicates whether pests were found in the beehive during the last harvest
fNoOfHarvests	Integer	Number of harvests completed in the beehive
fTotalHoneyHarvested	Real	The total amount of honey, in kilograms, harvested from the beehive
fHarvestDates	String	A concatenated string indicating all the dates on which the hive has been harvested so far, each date on a new line

The following methods have been provided:

- A completed method **setPests** that will change the current status of the **fPests** attribute
- A completed **toString** method

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

3.1.1 Write code for a constructor method to receive THREE parameters for the **fBeehiveID**, **fBeeCount** and **fPests** attributes.

Assign the parameters to the respective attributes and set the remaining attributes to the following values:

fNoOfHarvests = 2
fTotalHoneyHarvested = 80
fHarvestDates = '2025/01/05' + #13 + '2025/06/23' (4)

3.1.2 Write code for an accessor method called **getNoOfHarvests** to return the **fNoOfHarvests** attribute. (2)

3.1.3 Write code for a method called **calcAverage** to return the average amount of honey harvested in the hive, using the formula below.

$$Average = \frac{Total\ honey\ harvested}{Number\ of\ harvests}$$ (4)

3.1.4 Write code for a method called **updateBeehiveDetails** that receives a parameter value for the amount of honey harvested as a real value.

Write code to do the following:

- Add the parameter value to the **fTotalHoneyHarvested** attribute.
- Increment the **fNoOfHarvests** attribute by one.
- Add (join) the code for a new line (#13) and the system date to the content of the **fHarvestDates** attribute. (6)

3.1.5 Write code for a method called **checkHealthStatus** to return a Boolean value TRUE which indicates that the beehive is healthy, or FALSE if the beehive is not healthy.

The following criteria must be met for the status of the beehive to be healthy:

- The number of bees must be more than 7 000 OR the number of harvests must be less than or equal to 3.
- There must be no pests in the beehive. (6)

3.2 An incomplete program 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 **objBeehive**.

Code has been provided in the **FormCreate** method to disable button **btnQ3_2_3** and format the rich edit **redQ3_2** component.

Write code to perform the tasks described in QUESTION 3.2.1 to QUESTION 3.2.5.

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

Before harvesting honey from a beehive, the user must enter the details of the beehive.

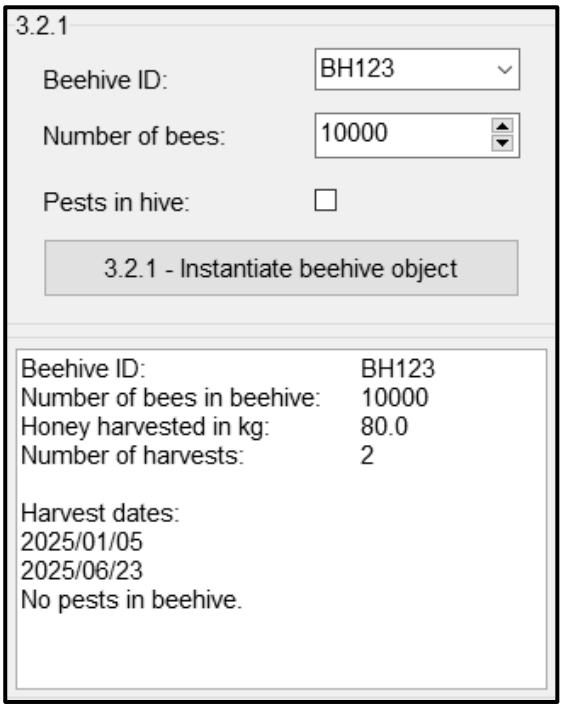
Code has been provided to do the following:

- Clear the rich edit **redQ3_2** component.
- Extract the beehive ID from combo box **cmbQ3**.
- Extract the number of bees from spin edit **spnQ3**.
- Extract the pest status from checkbox **chbQ3**.

Write code to do the following:

- Use the information extracted to instantiate the **objBeehive** object.
- Display the details of the object in the rich edit **redQ3_2** using the given **toString** method.

Example of input and output:



(5)

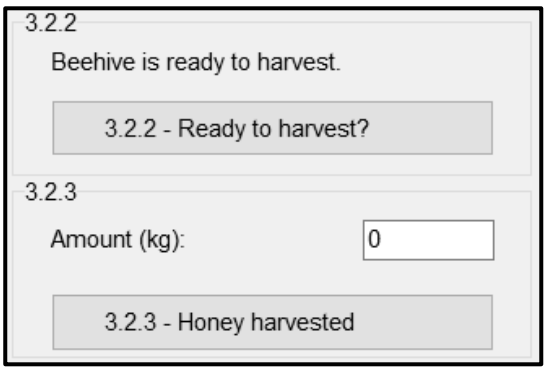
3.2.2 **Button [3.2.2 - Ready to harvest?]**

The health status of the beehive will need to be checked before a new harvest takes place.

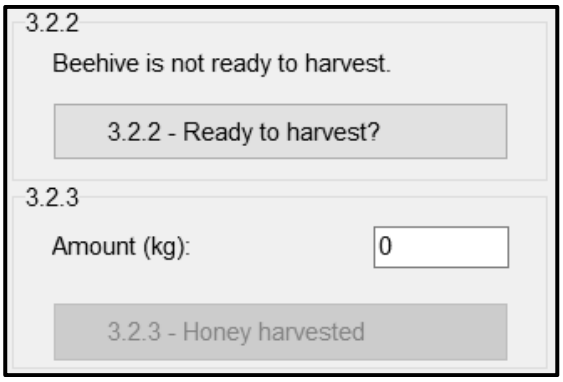
Write code to do the following:

- Call the **checkHealthStatus** method and display a suitable message in the label **lblQ3_2_2** to indicate whether the beehive is healthy (ready to harvest) or not healthy (not ready to harvest).
- If the beehive is ready to harvest, enable button **btnQ3_2_3**.
- If the beehive is NOT ready to harvest, disable button **btnQ3_2_3**.

Example of output if the beehive is ready to harvest:



Example of output if the beehive is NOT ready to harvest:



(4)

3.2.3 **Button [3.2.3 - Honey harvested]**

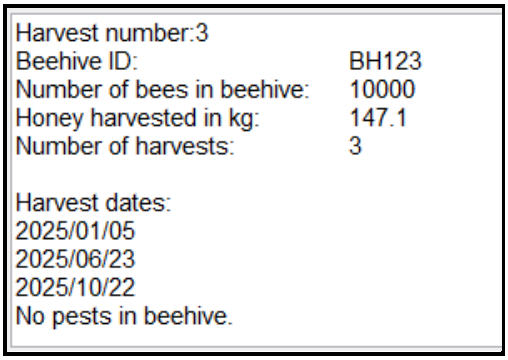
The amount of honey harvested during a new harvest must be added to the total amount of honey harvested from the beehive.

Code has been provided to clear the rich edit **redQ3_2** component.

Write code to do the following:

- Extract the amount of honey harvested from the edit box **edtQ3_2_3**.
- Call the **updateBeehiveDetails** method using the amount of honey as an argument.
- Display the following in the **redQ3_2** component:
    - The number of harvests in the format:
      `'Harvest number: ' <Number of harvests>`
    - The updated details of the beehive using the **toString** method

Example of output if the honey harvested during the new harvest was 67,1 kg and the number of completed harvests is three:

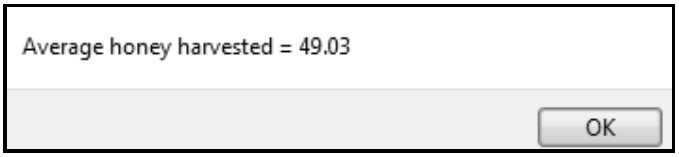


(4)

3.2.4 **Button [3.2.4 - Average honey harvested]**

Write code to call the relevant method to display the average amount of honey harvested from the beehive in a ShowMessage dialog box, formatted to TWO decimal places.

Example of output if the total amount of honey after three harvests was 147,1 kg:



(3)

3.2.5 **Button [3.2.5 - Change status of pests]**

The status of pests in the beehive must be changed, depending on whether pests have been detected or not.

A method called **setPests** has been provided in the object class that will change the current status of pests in the beehive.

Write code to do the following:

- Call the **setPests** method.
- Disable the button **btnQ3_2_3**.

Example of the disabled 'Honey harvested' button when the user clicked on the 'Change status of pests' button once:

(2)

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

TOTAL SECTION C: 40

SECTION D

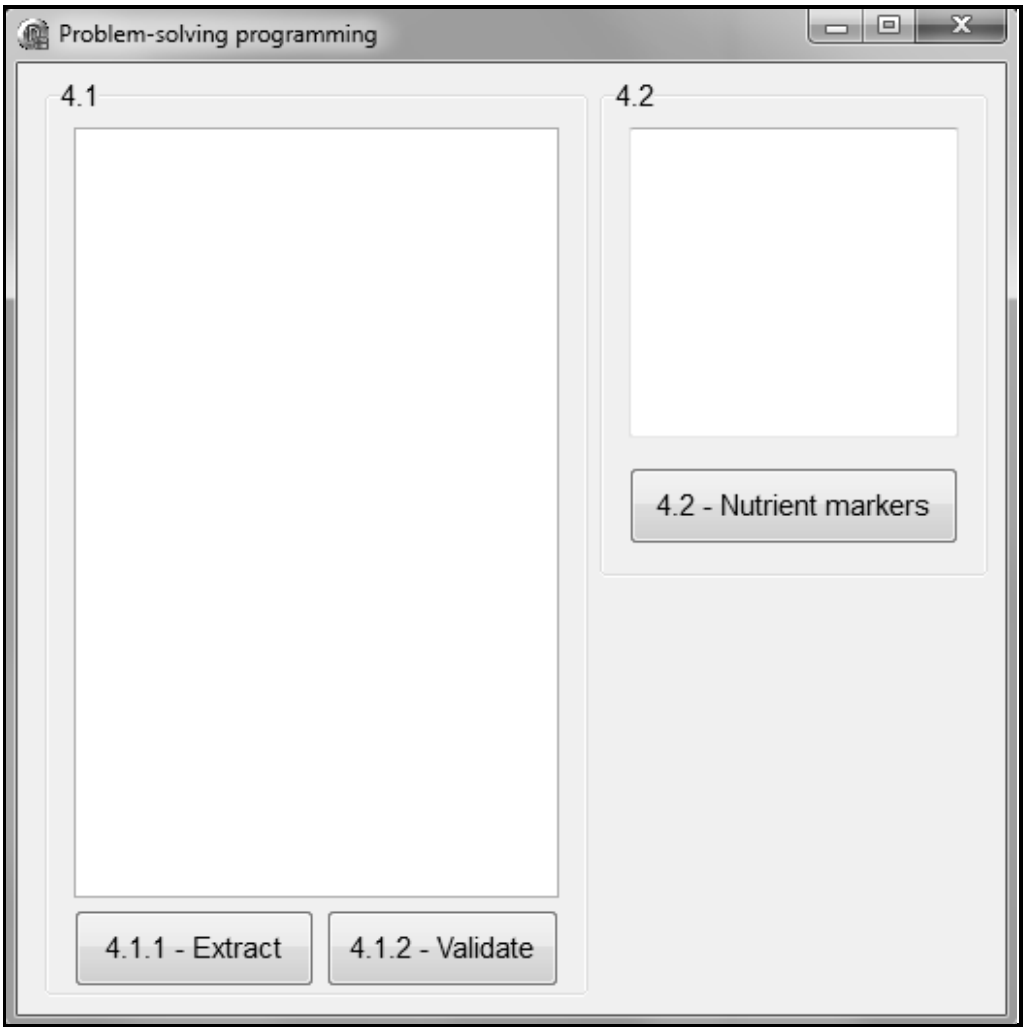
QUESTION 4: PROBLEM-SOLVING PROGRAMMING

SustainaFarm Collective is a company that does research on sustainable farming practices in areas where different types of crops have been planted.

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 limited functionality currently.

Example of the graphical user interface (GUI):



- Complete the code for each section of QUESTION 4, as described in QUESTION 4.1 and QUESTION 4.2.

4.1 The SustainaFarm Collective monitors the soil composition across ten hectares of farmland, recording the percentage values for clay, sand and silt in the soil per hectare.

You have been provided with the following:

- A completed **Display** procedure
- Declarations for the following arrays:
    - A populated one-dimensional array of type string where each element of the array represents the amount of clay, sand and silt in the soil:

```
arrSoil: array[1..10] of String =
('20:50:10', '40:30:30', '30:28:30', '60:20:20',
'25:30:45', '50:40:10', '30:55:15', '45:35:20',
'35:0:55', '55:25:20');
```

    - A two-dimensional array of type real:

```
arr2DSoil: array[1..10, 1..3] of Real;
```

4.1.1 **Button [4.1.1 - Extract]**

Write code to extract the Clay, Sand and Silt values from the **arrSoil** array for each hectare and store them in the provided two-dimensional array **arr2DSoil**.

Code has been provided to do the following:

- Clear the rich edit component **redQ4_1**.
- Call the **Display** method to output the data in the **arr2DSoil** two-dimensional array in the rich edit component **redQ4_1**.

Example of output:

Hectare	Clay	Sand	Silt
1	20.0	50.0	10.0
2	40.0	30.0	30.0
3	30.0	28.0	30.0
4	60.0	20.0	20.0
5	25.0	30.0	45.0
6	50.0	40.0	10.0
7	30.0	55.0	15.0
8	45.0	35.0	20.0
9	35.0	0.0	55.0
10	55.0	25.0	20.0

(8)

4.1.2 **Button [4.1.2 - Validate]**

It is essential for the SustainaFarm Collective that the sum of the Clay, Sand and Silt values for each hectare is equal to the value of 100.

Code has been provided to clear the **redQ4_1** component and to display the heading 'Hectares adjusted'.

Write code to do the following:

- Calculate the sum of the Clay, Sand and Silt values for each hectare.
- If the sum for any hectare does not add up to the value of 100, do the following:
    - Adjust the Clay, Sand and Silt values in the same ratio for that hectare to ensure that the sum of the three values add up to 100.
    - Display the hectare number in the **redQ4_1** component in the format:

```
Hectare: <hectare number>
```

NOTE: Assume that the total for each hectare will not exceed 100.

Code has been provided to do the following:

- Display the heading 'Updated data:'.
- Call the **Display** method to output the data of the **arr2DSoil** two-dimensional array in the rich edit component **redQ4_1**.

Example of output:

```
Hectares adjusted:
Hectare 1
Hectare 3
Hectare 9

Updated data:
========================
Hectare     Clay   Sand   Silt
1           25.0   62.5   12.5
2           40.0   30.0   30.0
3           34.1   31.8   34.1
4           60.0   20.0   20.0
5           25.0   30.0   45.0
6           50.0   40.0   10.0
7           30.0   55.0   15.0
8           45.0   35.0   20.0
9           38.9   0.0    61.1
10          55.0   25.0   20.0
```

(9)

4.2 **Button [4.2 - Nutrient markers]**

The SustainaFarm Collective is researching crop growth potential using unique nutrient markers. Nutrient markers are special numbers where the sum of the factorials of their digits equals the number itself.

Example 1:

The number 145 is a nutrient marker since the sum of the factorials of the digits equals the original number 145.

```
1! = 1 x 1 = 1
4! = 4 × 3 × 2 × 1 = 24
5! = 5 × 4 × 3 × 2 × 1 = 120

1 + 24 + 120 = 145
```

Example 2:

The number 32 is NOT a nutrient marker.

```
3! = 3 x 2 x 1 = 6
2! = 2 × 1 = 2

6 + 2 = 8 ≠ 32
```

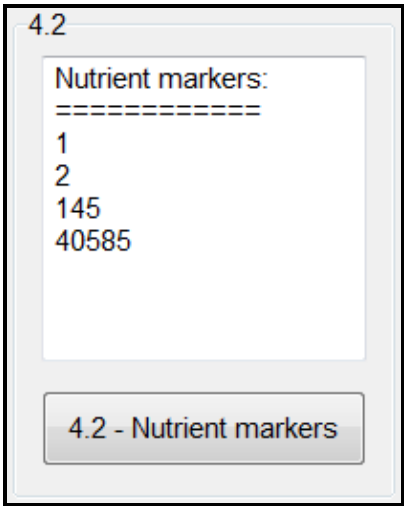
Code has been provided to do the following:

- Clear the memo component **memQ4_2**.
- Display the underlined heading 'Nutrient markers:'.

Write code to do the following:

- Determine all the nutrient markers in the range from 1 to 50 000 (inclusive).
- Display each nutrient marker in the memo component **memQ4_2**.

Example of output:



(13)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- Make a printout of 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: **tblFarmOwners**

This table contains the details of farm owners.

Field name	Data type	Description
OwnerID (PK)	Number	Unique identifier for the farm owner
FullName	Text (20)	Full name of the farm owner
ContactNumber	Text (12)	Contact number of the farm owner
Email	Text (25)	E-mail address of the farm owner
DateOfBirth	Date/Time	Date of birth of the farm owner

Example of the first nine records of the **tblFarmOwners** table:

OwnerID	FullName	ContactNumber	Email	DateOfBirth
101	John Smith	081 555 1201	john.smith@email.com	2002/01/12
102	Maria van Wyk	072 444 5622	maria.vw@email.com	1994/05/20
103	Themba Dlamini	083 678 9132	themba.d@email.com	1992/02/14
104	Sipho Mthembu	061 789 6512	sipho.m@email.com	1991/09/03
105	Fatima Khan	074 321 9043	fatima.k@email.com	2000/01/01
106	David Botha	065 543 2132		1992/02/12
107	Nkosi Zulu	082 111 7624	nkosi.z@email.com	1989/02/18
108	Peter van der Merwe	076 234 8132	peter.vdm@email.com	1995/07/27
109	Johann van Rensburg	071 899 1235	johann.vr@email.com	2003/04/28

Table: **tblFarms**

This table contains the details of each farm.

Field name	Data type	Description
FarmID (PK)	Number	Unique identifier for each farm
FarmName	Text (20)	Name of the farm
NearestTown	Text (20)	The town nearest to the farm
SizeInHectares	Number	Size of the farm in hectares
FarmType	Text (10)	Type of farm (e.g. Dairy, Poultry, Mixed)
OwnerID (FK)	Number	The owner ID of the farm owner

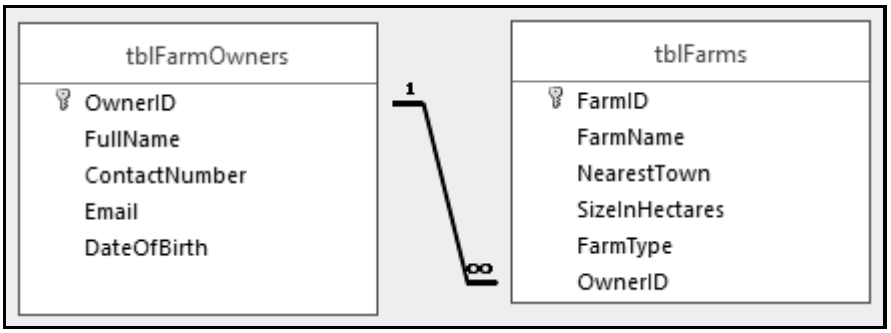
Example of the first eleven records of the **tblFarms** table:

FarmID	FarmName	NearestTown	SizeInHectares	FarmType	OwnerID
201	Green Pastures	Pietermaritzburg	85	Dairy	101
202	Golden Fields	Stellenbosch	120	Poultry	102
203	Riverbank Farm	Pretoria	95	Livestock	103
204	Blue Horizon Farm	Polokwane	150	Mixed	104
205	Sunrise Dairy	Rustenburg	200	Dairy	105
206	Orchard View	Queenstown	75	Livestock	106
207	Highland Meadow	Bloemfontein	90	Livestock	114
208	Valley Creek Farm	Nelspruit	110	Livestock	108
209	Sunny Ridge	Durban	95	Poultry	109
210	Golden Valley	Paarl	130	Dairy	110
211	Riverside Ranch	Pretoria	100	Livestock	108

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:



PLANNING PAGE 1

PLANNING PAGE 2

Examination sticker	150

INFORMATION TECHNOLOGY P1 – NOVEMBER 2025
INFORMATION SHEET *(to be completed by the candidate AFTER the 3-hour session)*

CENTRE NUMBER: _____

EXAMINATION NUMBER: _____

WORK STATION NUMBER: _____

Version of Delphi used during the INFORMATION TECHNOLOGY NSC NOV. 2025 Examination:

Mark appropriate box with a cross (**X**)	Delphi 2010	Delphi XE	Delphi 10.3	Delphi Community	Delphi 11	Delphi 12	Other: (*Specify*) _____

FOLDER NAME: _____

Candidate must tick if the file name(s) used for each answer has been saved and/or attempted.

Question number	File name	Saved (✓)	Attempted (✓)	Maximum Mark	Mark Achieved	Marker Code
1	Question1_P.dproj			**40**		
2	Question2_P.dproj			**40**		
3	Beehive_U.pas			**22**		
	Question3_P.dproj			**18**		
4	Question4_P.dproj			**30**		
TOTAL				**150**		

Comment: *(for office/marker use only)*

_____

_____

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2025

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 27 pages.

GENERAL INFORMATION:

- These marking guidelines are to be used as the basis for the marking session. They were prepared for use by markers. All markers are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' work.

- Note that learners who provide an alternate correct solution to that given as example of a solution in the marking guidelines will be given full credit for the relevant solution, unless the specific instructions in the paper was not followed or the requirements of the question was not met

- **Annexures A, B, C** and **D** (pages 3 to 12) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 13 to 27) contain examples of solutions for Questions 1 to 4 in programming code.

- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 12) should be made for each learner and completed during the marking session.

ANNEXURE A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
1.1	**Button [1.1 - Increase value]** Extract the number from pnlQ1_1 ✓ Convert to integer ✓ Increase the number by 1 ✓ Display the number in pnlQ1_1 converted to string ✓	**4**	

1.2	**Button [1.2 - Volume]**
	
	Extract the length of the one side from spnQ1_2 ✓
	Calculate volume:
	`rVolume :=`
	`  5 * ✓ (3 + SQRT(5)) ✓/ 12 ✓ * Power(iSide,3) ✓`
	
	Display volume in edtQ1_2 converted to a string and formatted to 2 decimal places ✓
	
	NOTE: Any TWO mathematical functions can be used.
		6	

1.3	**Button [1.3 - Shopping aisle]** Extract the code from edtQ1_3 ✓ Find the position of hash (#) (or other mechanism to separate the aisle number from the category) ✓ Extract aisle number ✓ Extract/use the category as character ✓ Use a Case correctly ✓ For all possible characters ✓ ('F', 'D', 'B', 'S', 'P') ✓ to determine category description ✓ Compile a string with "Aisle ", aisle number ✓, ": " and category description ✓ Display the string in lblQ1_3 ✓	**11**	

1.4	**Button [1.4 - Populate and display]** Loop from 1 to length of array ✓ Generate random numbers ✓ in correct range (1 – 15) ✓ Assign the random number to each element in the array ✓ Concatenate the array values to build the string ✓ separating the values with a dash (-) ✓ Mechanism to deal with extra dash ✓ Display the concatenated string in memQ1_4 ✓	**8**	
1.5	**Button [1.5 - Determine HCF]** Test IF Num1 < Num2 ✓ Assign Num1 to LowerNum ✓ Else Assign Num2 to LowerNum ✓ *//Use of the While loop* Cnt = 1; ✓ While Cnt <= LowerNum ✓ Test IF ✓ Num1 MOD Cnt = 0 ✓ Test IF Num2 MOD Cnt = 0 ✓ HCF = Cnt ✓ Inc(Cnt) ✓ Display HCF in edtQ1_5 ✓ *//Alternative using the FOR… loop* Loop Cnt (1) from 1 to (1) LowerNum (1) Test IF (1) Num1 MOD Cnt = 0 (1) Test IF Num2 MOD Cnt = 0 (1) HCF = Cnt (1) Display HCF in edtQ1_5 (1)	**11**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – SQL AND DATABASE PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**		
2.1.1	**Button [2.1.1 - Farm details]** `SELECT FarmName, NearestTown, SizeInHectares` ✓ `FROM tblFarms` ✓ `ORDER BY SizeInHectares DESC` ✓	**3**	

2.1.2	**Button [2.1.2 - Contact details]**
	`SELECT FullName, ContactNumber`
	`FROM tblFarmOwners` ✓
	`WHERE Email IS NULL` ✓
	
	Alternative:
	`ISNULL(Email)`
		2	

2.1.3	**Button [2.1.3 - Average farm size]**
	`SELECT FarmType,`
	`FORMAT(AVG(SizeInHectares)` ✓`,"0.00"` ✓`)`
	`AS AverageSize` ✓
	`FROM tblFarms`
	`WHERE FarmType =` ✓ `"' + sFarmType + '"` ✓
	`GROUP BY FarmType` ✓
	
	Alternative:
	`FarmType = '+ QuotedStr(sFarmType) +'`
	`FarmType LIKE "' + sFarmType + '"`
		6	

2.1.4	**Button [2.1.4 - Young farm owners]**
	`SELECT Fullname, DateOfBirth,` ✓
	`INT` ✓ `((NOW` ✓ `- DateOfBirth) /365` ✓`) AS Age`
	`FROM tblFarmOwners` ✓
	`WHERE INT((NOW - DateOfBirth)/365)` ✓ `<= 25` ✓
	
	Alternative:
	`(Date() - DateOfBirth)`
		7	

2.1.5	**Button [2.1.5 - Multiple farms in KZN]**
	`SELECT FullName, COUNT(*) ✓ AS TotalFarms`
	`FROM tblFarmOwners, tblFarms ✓`
	`WHERE tblFarmOwners.OwnerID = tblFarms.OwnerID ✓`
	`AND ✓ (NearestTown = "Durban" OR NearestTown = "Pietermaritzburg") ✓`
	`GROUP BY FullName ✓`
	`HAVING COUNT(*) > 1 ✓`
	
	`Alternative:`
	`NearestTown IN ("Durban","Pietermaritzburg")`
		7	

	Subtotal:	**25**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**Database Manipulation**		
	Button [2.2 - Mixed-farm type] Test IF file exists = False/try..except ✓ Display a suitable message ✓ Exit or else statement ✓ AssignFile(tFile,'MixedFarms.txt') ✓ Reset(tFile) ✓ Loop through Text file ✓ Readln(tFile, FarmID variable) ✓ tblFarms.First ✓ Loop through tblFarms ✓ Test IF FarmID variable = tblFarms['FarmID'] ✓ Test IF tblFarms['SizeInHectares'] > 100 ✓ tblFarms.Edit ✓ tblFarms['FarmType'] = 'Mixed' ✓ tblFarms.Post Else Display tblFarms['FarmID'] as string, tblFarms['FarmName'] and tblFarms['SizeInHectares'] as string in redQ2_2 ✓ tblFarms.Next ✓ End loop (tblFarms) End loop (Text file)	**15**	
	Subtotal:	**15**	
	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**Constructor method** Constructor heading with String, Integer and Boolean parameters ✓ Assign the three parameters to correct attributes ✓ Assign the following default values to the other attributes: fNoOfHarvests = 2 fTotalHoneyHarvested = 80 ✓ fHarvestDates = '2025/01/05' + #13 + '2025/06/23' ✓	**4**	
3.1.2	**Function getNoOfHarvests** Function heading getNoOfHarvests with integer return type ✓ Return fNoOfHarvests ✓	**2**	
3.1.3	**Function calcAverage** Function heading calcAverage ✓ with real return type ✓ Return ✓ fTotalHoneyHarvested/ fNoOfHarvests ✓ **ALSO ACCEPT:** fTotalHoneyHarvested/ getNoOfHarvests	**4**	
3.1.4	**Procedure updateBeehiveDetails** Procedure heading with a real parameter ✓ fTotalHoneyHarvested = fTotalHoneyHarvested ✓ + parameter ✓ fNoOfHarvests = fNoOfHarvests + 1 ✓ fHarvestDates = fHarvestDates ✓ + #13 + system date ✓	**6**	

3.1.5	**Function checkHealthStatus** Function heading with Boolean return type ✓ Test IF ((fBeeCount > 7000) ✓ OR (fNoOfHarvests <=3)) ✓ AND (fPests = False) ✓ Result = True ✓ Else Result = False ✓ *Alternative 1:* Function heading with Boolean return type (1) Result = False (1) Test IF (fBeeCount > 7000) (1) OR (fNoOfHarvests <=3) (1) Test IF (fPests = False) (1) Result = True (1) *Alternative 2:* Function heading with Boolean return type (1) Result = (2) ((fBeeCount > 7000) (1) OR (fNoOfHarvests <=3)) (1) AND (fPests = False) (1)	**6**	
	Subtotal: Object class	**22**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 - Instantiate beehive object]** objBeehive := ✓ TBeehive.Create() ✓ Extracted parameters in sequence ✓ Display object in redQ3_2 ✓ using toString method ✓	**5**	

3.2.2	**Button [3.2.2 - Ready to harvest?]**
	
	Test IF objBeehive.checkHealthStatus = TRUE ✓
	Display 'Ready to harvest' on lblQ3_2_2
	btnQ3_2_3.Enabled = True
	Else ✓
	Display 'Not ready to harvest' on lblQ3_2_2 ✓ //both
	btnQ3_2_3.Enabled = False ✓ //both
	
	Concepts:
	Test IF condition (1) with Else statement (1)
	Display message 'Ready to harvest'/ 'Not ready to harvest' in lblQ3_2_2 (1)
	btnQ3_2_3.Enabled = True/False(1)
		4	

3.2.3	**Button [3.2.3 - Honey harvested]** Extract Value from edtQ3_2_3 as real data type ✓ Call objBeehive.updateBeehiveDetails (Value) ✓ Display in redQ3_2: Label + objBeehive.getNoOfHarvests as string ✓ objBeehive.toString ✓	**4**	
3.2.4	**Button [3.2.4 - Average honey harvested]** Call objBeehive.calcAverage ✓ to display average using ShowMessage ✓ formatted to 2 decimal places ✓	**3**	
3.2.5	**Button [3.2.5 - Change status of pests]** Call objBeehive.setPests ✓ Disable btnQ3_2_3 ✓	**2**	
	Subtotal Form class:	**18**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1.1	**Button [4.1.1 - Extract]** Loop Row from 1 to 10 ✓ Loop Col from 1 to 2 ✓ Find position of placeholder ✓ in arrSoil[Row] ✓ Extract value as number ✓ and store in arr2DSoil[Row,Col] ✓ Delete from index 1 up to placeholder ✓ Mechanism to deal with last value ✓	**8**	
4.1.2	**Button [4.1.2 - Validate]** Loop Row from 1 to 10 ✓ Initialise Total to 0 ✓ Loop Col from 1 to 3 ✓ Increment Total with value of arr2DSoil[Row,Col] ✓ Test IF Total < 100 ✓ Loop A from 1 to 3 ✓ Calculate ratio arr2DSoil[Row, A] ✓ / Total *100 ✓ Display hectare with Row number as string in redQ4_1 ✓	**9**	

4.2	**Button [4.2 - Nutrient markers]**
	
	Loop Loop1 from 1 to 50000 ✓
	Initialise Sum to 0 ✓
	Loop ✓ Loop2 from 1 to Length(Loop1) ✓ //number of digits
	Initialise Factorial to 1 ✓
	Loop ✓ Loop3 from 1 to
	typecast integer (typecast string (Loop1)✓ [Loop2]) ✓ //value of digit
	Factorial = Factorial * Loop3✓
	Sum = Sum + Factorial ✓
	Test ✓ IF Sum = Loop1 ✓
	Display Loop1/ Sum as string in memQ4_2 ✓
	
	Concepts:
	Outer Loop1 (1)
	Initialise Sum variable (1)
	Inner Loop2 - Nested loop2 (1) using length of the string of outer loop variable (1)
	Initialise Factorial variable (1)
	Digit extraction - Nested Loop3 (1), Digit extraction (1), Typecast to integer (1)
	Factorial calculation (1)
	Sum logic (1)
	Comparison logic - IF statement (1) Sum = outer loop1 (1)
	Output (1)
		13	

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

SUMMARY OF LEARNER'S MARKS:

CENTRE NUMBER:		LEARNER'S 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, math, StdCtrls, ExtCtrls, Spin, jpeg, ComCtrls;

type
    gpbQ1_1: TGroupBox;
    pnlQ1_1: TPanel;
    btnQ1_1: TButton;
    gpbQ1_2: TGroupBox;
    lblLength: TLabel;
    spnQ1_2: TSpinEdit;
    imgIcosahedron: TImage;
    edtQ1_2: TEdit;
    btnQ1_2: TButton;
    gpbQ1_3: TGroupBox;
    lblDate: TLabel;
    edtQ1_3: TEdit;
    lblQ1_3: TLabel;
    gpbQ1_4: TGroupBox;
    memQ1_4: TMemo;
    btnQ1_4: TButton;
    lblVolume: TLabel;
    btnQ1_3: TButton;
    GroupBox1: TGroupBox;
    edtQ1_5_Num1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    edtQ1_5_Num2: TEdit;
    btnQ1_5: TButton;
    edtQ1_5: TEdit;
    procedure btnQ1_1Click(Sender: TObject);
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure btnQ1_5Click(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);
var
  iNum: Integer;
begin

// Question 1.1
  iNum := StrToInt(pnlQ1_1.Caption);
  Inc(iNum);
  pnlQ1_1.Caption := IntToStr(iNum);
end;

// ============================================================================
// Question 1.2          6 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  iSide: Integer;
  rVolume: Real;
begin
// Question 1.2
  iSide := spnQ1_2.Value;
  rVolume := 5 * (3 + Sqrt(5)) / 12 * Power(iSide,3);
  edtQ1_2.Text := FloatToStrF(rVolume, ffFixed, 8, 2);
end;

// ============================================================================
// Question 1.3          11 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  sCode, sOutput: string;
  sAisle: string;
  cCategory : Char;
  iPos : Integer;
begin
// Question 1.3
  sCode := edtQ1_3.Text;
  iPos := Pos('#', sCode);
  cCategory := sCode[iPos+1];
  sAisle := Copy(sCode, 1, iPos-1);
  case cCategory of
    'F':  sOutput := 'Fruit and vegetables';
    'D':  sOutput := 'Dairy';
    'B':  sOutput := 'Butchery';
    'S':  sOutput := 'Sauces';
    'P':  sOutput := 'Pasta and Rice';
  end;
  sOutput := 'Aisle ' + sAisle + ': ' + sOutput;
  lblQ1_3.Caption := sOutput;
end;
```

```
// ========================================================================
// Question 1.4   8 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
//Provided code
var
  arrNumbers: array [1 .. 9] of Integer;
//End of provided code

  iCnt: Integer;
  sOut: String;
begin
// Question 1.4

  sOut := '';
  for iCnt := 1 to length(arrNumbers) do
  begin
    arrNumbers[iCnt] := RandomRange(1, 16);
    sOut := sOut + '-' + IntToStr(arrNumbers[iCnt]);
  end;
  Delete(sOut, length(sOut), 1);
  memQ1_4.Text := sOut;
end;

// ========================================================================
// Question 1.5       11 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  iCnt: Integer;
  iHCF, iLowerNum, iNum1, iNum2: Integer;
begin
//Provided code
  iNum1 := StrToInt(edtQ1_5_Num1.Text);
  iNum2 := StrToInt(edtQ1_5_Num2.Text);
  iHCF := 0;

// Question 1.5
// Code using a while loop
if iNum1 < iNum2 then
    iLowerNum := iNum1
  else
    iLowerNum := iNum2;

// Code using a while loop
  iCnt := 1;
  While iCnt <= iLowerNum do
  begin
    if (iNum1 MOD iCnt = 0) then
      if (iNum2 MOD iCnt = 0) then
        iHCF := iCnt;
    Inc(iCnt);
  end;
```

```
Code using a Repeat Until loop
 { iCnt : = 1;
  Repeat
    if (iNum1 MOD iCnt = 0) then
      if (iNum2 MOD iCnt = 0) then
        iHCF := iCnt;
    Inc(iCnt);
 Until iCnt > iLowerNum;
  end;}

//Code using a For loop
  {for iCnt := 1 to iLowerNum do
  begin
    if (iNum1 MOD iCnt = 0) then
      if (iNum2 MOD iCnt = 0) then
      begin
        iHCF := iCnt;
      end;
  end;}

  edtQ1_5.Text := 'HCF: ' + IntToStr(iHCF);

end;
end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
//=============================================================================
// Question 2.1 – Section: SQL statements
//=============================================================================

//=============================================================================
//  Question 2.1.1          3 marks
//=============================================================================
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: string;
begin
  // Question 2.1.1

  sSQL1 := 'SELECT FarmName, NearestTown, SizeInHectares FROM tblFarms ' +
    'ORDER BY SizeInHectares DESC';

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

//=============================================================================
//  Question 2.1.2          2 marks
//=============================================================================
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: string;
begin
  // Question 2.1.2

  sSQL2 := 'SELECT FullName, ContactNumber  ' +
    'FROM tblFarmOwners WHERE Email IS NULL';

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

//=============================================================================
//  Question 2.1.3          6 marks
//=============================================================================
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3, sFarmType: string;
begin
  // Provided code
  sFarmType := cmbQ2_1_3.Text;

  // Question 2.1.3
  sSQL3 :=
    'SELECT FarmType, FORMAT(AVG(SizeInHectares),"0.00") as AverageSize ' +
    'FROM tblFarms ' + 'WHERE FarmType = "' + sFarmType + '" GROUP BY
     FarmType';

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

```
//=============================================================================
//  Question 2.1.4          7 marks
//=============================================================================
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: string;
begin
  //Question 2.1.4

  sSQL4 :=  'SELECT FullName, DateOfBirth, INT((NOW - DateOfBirth)/365)
                AS Age ' +
            'FROM tblFarmOwners ' +
            'WHERE INT((NOW - DateOfBirth)/365) <= 25  ';
  // Alternative
   sSQL4:= 'SELECT FullName, INT((NOW - DateOfBirth)/365) AS Age
    ' +
    ' FROM tblFarmOwners' +
    ' WHERE INT((NOW - DateOfBirth)/365) <= 25  ';

  // Provided code - do not change
  dbCONN.runSQL(sSQL4)

end;

//=============================================================================
//  Question 2.1.5          7 marks
//=============================================================================
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5: string;
begin
  // Question 2.1.5
  sSQL5 :=  'SELECT  FullName,  COUNT(*) AS TotalFarms ' +
            'FROM tblFarmOwners, tblFarms ' +
            'WHERE tblFarmOwners.OwnerID = tblFarms.OwnerID AND
            (NearestTown = "Durban" OR NearestTown = "Pietermaritzburg")' +
            'GROUP BY FullName ' +
            'HAVING COUNT(*) > 1 ';

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

```
//=============================================================================
// Question 2.2 – Section Delphi code
//=============================================================================

//=============================================================================
// Question 2.2              15 marks
// ============================================================================

procedure TfrmQuestion2.btnQ2_2Click(Sender: TObject);
var
  tFile: TextFile;
  iFarmID: Integer;
  sFarmType: string;
begin
  // Provided code - do not change
  redQ2_2.Clear;
  redQ2_2.Lines.Add('Farms on list that do not qualify: ');

  // Question 2.2
  Assignfile(tFile, 'MixedFarms.txt');
  try
    Reset(tFile);
  except
    ShowMessage('File does not exist');
    Exit;
  end;

  while not Eof(tFile) do
  begin
    Readln(tFile, iFarmID);
    tblFarms.First;
    while not tblFarms.Eof do
    begin
      if (tblFarms['FarmID'] = iFarmID) then
      begin
        if tblFarms['SizeInHectares'] > 100 then
        begin
          tblFarms.Edit;
          tblFarms['FarmType'] := 'Mixed';
          tblFarms.Post;
        end
        else
          redQ2_2.Lines.Add(IntToStr(tblFarms['FarmID']) + ' - ' + tblFarms
              ['FarmName'] + ' - ' + IntToStr(tblFarms['SizeInHectares'])+
                                      ' hectares');
      end;
      tblFarms.Next;
    end;
  end;

  // Provided code
  tblFarms.Sort := 'FarmID ASC';
end;
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
unit Beehive_U;

interface

Uses SysUtils, DateUtils, Math;

Type
  TBeehive = class(TObject)
  Private
    fBeehiveID: String;
    fBeeCount: Integer;
    fPests: Boolean;
    fNoOfHarvests: Integer;
    fTotalHoneyHarvested: Real;
    fHarvestDates: String;

  Public
    Constructor Create(sBeehiveID: String; iBeeCount: Integer;
                           bPests : boolean);
    Function getNoOfHarvests: Integer;
    Function calcAverage: Real;
    Procedure updateBeehiveDetails(rHoneyKG: Real);
    Function checkHealthStatus: Boolean;


    // Provided code
    Procedure setPests;
    Function toString: String;
  End;
implementation

{ TBeehive }
// ============================================================================
// Question 3.1.1          4 marks
// ============================================================================
Constructor TBeehive.Create(sBeehiveID: String; iBeeCount: Integer; bPests
: boolean);
begin
  fBeehiveID := sBeehiveID;
  fBeeCount := iBeeCount;
  fPests := bPests;
  fNoOfHarvests := 2;
  fTotalHoneyHarvested := 80;
  fHarvestDates :=  '2025/01/05' + #13 +  '2025/06/23';
end;
// ============================================================================
// Question 3.1.2          2 marks
// ============================================================================
function TBeehive.getNoOfHarvests: Integer;
begin
  Result := fNoOfHarvests;
end;
```

```
// ============================================================================
// Question 3.1.3          4 marks
// ============================================================================
function TBeehive.calcAverage: Real;
begin
  Result := fTotalHoneyHarvested / fNoOfHarvests;
end;
// ============================================================================
// Question 3.1.4          6 marks
// ============================================================================
procedure TBeehive.updateBeehiveDetails(rHoneyKG: Real);
begin
  fTotalHoneyHarvested := fTotalHoneyHarvested + rHoneyKG;
  Inc(fNoOfHarvests);
  fHarvestDates := fHarvestDates + #13 + DateToStr(Date);
end;
// ============================================================================
// Question 3.1.5          6 marks
// ============================================================================
   function TBeehive.checkHealthStatus: Boolean;
   var
       bHealthy : Boolean;
   begin
     bHealthy := False;
     if (fPests = False) AND ((fBeeCount > 7000) OR
                                    (fNoOfHarvests <= 3)) then
        bHealthy := True;
     Result := bHealthy;
   //Alternative
     Result:= (fPests = False) AND ((fBeeCount > 7000) OR
                                    (fNoOfHarvests <= 3))
   end;
// ============================================================================
// Provided methods
// ============================================================================
   procedure TBeehive.setPest;
   begin
     if fPests = True then
       fPests := False
     else
       fPests := True;
   end;
   function TBeehive.toString: String;
   var
     sPestStatus: String;
   begin
     if fPestStatus then
       sPestStatus :='Pests in beehive.'
     else
       sPestStatus := 'No pests in beehive.';
     Result := 'Beehive ID: ' + #9 + fBeehiveID + #13 +
               'Number of bees in beehive: ' +#9 + IntToStr(fBeeCount)+#13+
               'Honey harvested in kg: ' + #9 +
                FloatToStrF(fTotalHoneyHarvested, ffFixed, 8, 1)+#13+
               'Number of harvests:'+ #9+ IntToStr(fNoOfHarvests)+#13#13+
               'Harvest dates: ' + #13 + fHarvestDates + #13 + sPestStatus;
   end;
   end.
```

Main Form Unit

```
unit Question3_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, pngimage, ExtCtrls, Spin, ComCtrls, Math,
beehive_u;

type
  TfrmQuestion3_2 = class(TForm)
    pnlQ3_2: TPanel;
    imgBeeLeft: TImage;
    gpbQ3_2_1: TGroupBox;
    cmbQ3: TComboBox;
    lblCode: TLabel;
    btnQ3_2_1: TButton;
    lblNumBees: TLabel;
    gpbQ3_2_3: TGroupBox;
    lblHavest: TLabel;
    imgBeeRight: TImage;
    gpbQ3_2_4: TGroupBox;
    btnQ3_2_4: TButton;
    redQ3_2: TRichEdit;
    btnQ3_2_3: TButton;
    spnQ3: TSpinEdit;
    edtQ3_2_3: TEdit;
    btnQ3_2_2: TButton;
    gpbQ3_2_2: TGroupBox;
    GroupBox1: TGroupBox;
    lblQ3_2_2: TLabel;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    btnQ3_2_5: TButton;
    Edit1: TEdit;
    chbQ3: TCheckBox;
    lblPests: TLabel;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnQ3_2_4Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_5Click(Sender: TObject);
    procedure chbQ3Click(Sender: TObject);
    procedure GroupBox2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion3_2: TfrmQuestion3_2;
  // Provided code
  objBeehive: TBeehive;


implementation
```

```
// ========================================================================
// Question 3.2.1          5 marks
// ========================================================================
procedure TfrmQuestion3_2.btnQ3_2_1Click(Sender: TObject);
var
  // Provided code
  sBeehiveID: string;
  iNumBees: Integer;
  bPests: Boolean;
begin

  // Provided code
  redQ3_2.Clear;
  sBeehiveID := cmbQ3.Text;
  iNumBees := spnQ3.Value;
  bPests := chbQ3.checked;

  // Question 3.2.1
  objBeehive := TBeehive.Create(sBeehiveID, iNumBees, bPests);
  redQ3_2.Lines.Add(objBeehive.toString);
end;;

// ========================================================================
// Question 3.2.2          4 marks
// ========================================================================
procedure TfrmQuestion3_2.btnQ3_2_2Click(Sender: TObject);
begin
  // Question 3.2.2
  if objBeehive.checkHealthStatus then
    lblQ3_2_2.Caption := 'Beehive is ready to harvest.'
  else
    lblQ3_2_2.Caption := 'Beehive is not ready to harvest.';
  btnQ3_2_3.Enabled := objBeehive.checkHealthStatus;
end;

// ========================================================================
// Question 3.2.3          4 marks
// ========================================================================
procedure TfrmQuestion3_2.btnQ3_2_3Click(Sender: TObject);
var
  rAmount: Real;
begin
  // Provided code
  redQ3_2.Clear;

  // Question 3.2.3
  rAmount := StrToFloat(edtQ3_2_3.Text);
  objBeehive.updateBeehiveDetails(rAmount);
  redQ3_2.Lines.Add('Harvest number:' +
           IntToStr(objBeehive.getNoOfHarvests));
  redQ3_2.Lines.Add(objBeehive.toString);
end;
```

```
// ============================================================================
// Question 3.2.4        3 marks
// ============================================================================
procedure TfrmQuestion3_2.btnQ3_2_4Click(Sender: TObject);
begin
  // Question 3.2.4
  ShowMessage('Average honey harvested = ' + FloatToStrF
      (objBeehive.calcAverage, ffFixed, 6, 2));
end;

// ============================================================================
// Question 3.2.5        2 marks
// ============================================================================
procedure TfrmQuestion3_2.btnQ3_2_5Click(Sender: TObject);
begin
  objBeehive.setPests;
  btnQ3_2_3.Enabled := false;
end;

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

{$REGION 'Provided code - DO NOT MODIFY'}

procedure TfrmQuestion3_2.FormCreate(Sender: TObject);
begin
  redQ3_2.clear;
  redQ3_2.Paragraph.tabcount := 1;
  redQ3_2.Paragraph.tab[0] := 150;

  btnQ3_2_3.Enabled := False;
end;
{$ENDREGION}

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;

interface

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

type
  TfrmQuestion4 = class(TForm)
    gbxQ4_1: TGroupBox;
    gbxQ4_2: TGroupBox;
    btnQ4_1_1: TButton;
    btnQ4_1_2: TButton;
    btnQ4_2: TButton;
    memQ4_2: TMemo;
    redQ4_1: TRichEdit;
    procedure btnQ4_1_1Click(Sender: TObject);
    procedure btnQ4_1_2Click(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);

  private
    { Private declarations }
  public
    procedure Display;
  end;

var
  frmQuestion4: TfrmQuestion4;
  // Provided code
  arrSoil: array [1 .. 10] of String = (
    '20:50:10',
    '40:30:30',
    '30:28:30',
    '60:20:20',
    '25:30:45',
    '50:40:10',
    '30:55:15',
    '45:35:20',
    '35:0:55',
    '55:25:20'
  );
  arr2DSoil: array [1 .. 10, 1 .. 3] of Real;

implementation

{$R *.dfm}
```

```
// ========================================================================
// Question 4.1.1          8 marks
// ========================================================================
procedure TfrmQuestion4.btnQ4_1_1Click(Sender: TObject);
var
  iRow, iCol, iPos: Integer;
begin
  // Question 4.1.1

  for iRow := 1 to 10 do
  begin
    for iCol := 1 to 2 do
    begin
      iPos := pos(':', arrSoil[iRow]);
      arr2DSoil[iRow, iCol] := StrToInt(copy(arrSoil[iRow], 1, iPos - 1));
      delete(arrSoil[iRow], 1, iPos);
    end;
    arr2DSoil[iRow, iCol] := StrToInt(arrSoil[iRow]);
  end;

  // Provided code
  redQ4_1.Clear;
  Display;
end;

// ========================================================================
// Question 4.1.2          9 marks
// ========================================================================
procedure TfrmQuestion4.btnQ4_1_2Click(Sender: TObject);
var
  iRow, iCol, A: Integer;
  rClay, rSand, rSilt, rSum: Real;
begin
  // Provided code
  redQ4_1.Clear;
  redQ4_1.lines.add('Hectares adjusted:');
// Question 4.1.2
  for iRow := 1 to 10 do
  begin
    rSum := 0;
    for iCol := 1 to 3 do
      rSum := rSum + arr2DSoil[iRow, iCol];

    if (rSum < 100) then
    begin
      for A := 1 to 3 do
        arr2DSoil[iRow, A] := (arr2DSoil[iRow, A] / rSum * 100);
      redQ4_1.Lines.Add('Hectare ' + IntToStr(iRow));
    end;
  end;
  // Provided code
  redQ4_1.Lines.Add(#13 + 'Updated data: ');
  redQ4_1.Lines.Add('========================');
  Display;
end;


// ========================================================================
```

```
// Question 4.2          13 marks
// ============================================================================
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  rTFactor, rTNum: Real;
  iValue: Integer;
  iLoop1, iLoop2, iLoop3: Integer;
begin
  // Provided code
  memQ4_2.Clear;
  memQ4_2.Lines.Add('Nutrient markers:');
  memQ4_2.Lines.Add('=================');

  // Question 4.2
  For iLoop1 := 1 to 50000 do
  begin
    rTNum := 0;
    For iLoop2 := 1 to Length(IntToStr(iLoop1)) do
    begin
      rTFactor := 1;
      For iLoop3 := 1 to StrToInt(IntToStr(iLoop1)[iLoop2]) do
        rTFactor := rTFactor * iLoop3;
      rTNum := rTNum + rTFactor;
    end;
    if rTNum = iLoop1 then
      memQ4_2.Lines.Add(IntToStr(iLoop1));
  end;
end;


//============================================================================
// Display Method
// ============================================================================
// Provided code
procedure TfrmQuestion4.Display;
var
  iRow, iCol: Integer;
  sTotal: String;
begin
  redQ4_1.Lines.Add('Hectare' + #9 + 'Clay' + #9 + 'Sand' + #9 + 'Silt');
  For iRow := 1 to 10 do
  begin
    sTotal := IntToStr(iRow) + #9#9;
    For iCol := 1 to 2 do
      sTotal := sTotal + FloatToStrF(arr2DSoil[iRow, iCol],
                                     ffFixed, 8, 1) + #9;
    sTotal := sTotal + FloatToStrF(arr2DSoil[iRow, 3], ffFixed, 8, 1);
    redQ4_1.Lines.Add(sTotal);
  end;
end;

end.
```