basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2023

MARKS: 150

TIME: 3 hours

This question paper consists of 25 pages, 2 data pages and 2 pages for planning.

INSTRUCTIONS AND INFORMATION

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

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

5. 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.

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

7. 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.

8. 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.

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

10. 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.

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

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

13. 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.

14. 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: **DataNov2023.exe**
- Click on the 'Extract' button.
- Enter the following password: **%Learn4Life@**

Once extracted, the following list of files will be available in the folder **DataNov2023**:

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

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

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

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

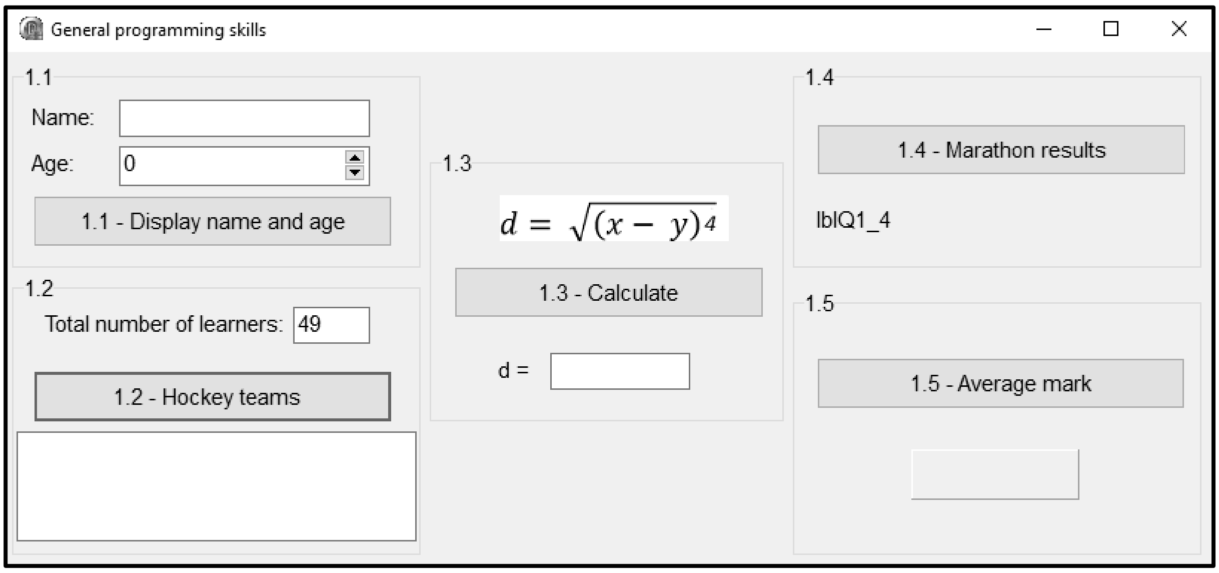
SECTION A

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 that follow.

1.1 **Button [1.1 - Display name and age]**

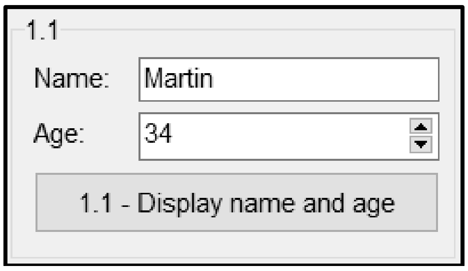
The user must do the following:

- Enter a name in the edit box **edtQ1_1**.
- Enter/Select the person's age in the spin edit box **spnQ1_1**.

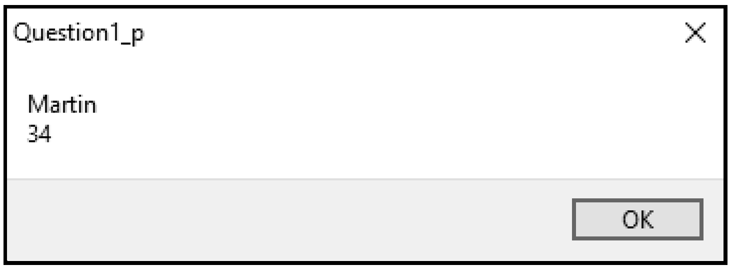
Write code to do the following:

- Extract the name entered from the edit box **edtQ1_1** and store the name in the provided **sName** variable.
- Extract the age entered/selected from the spin edit **spnQ1_1** and store the age in the provided **iAge** variable.
- Display the name and age, one below the other, using an output dialogue box.

Example of input:



Example of output:



```
Martin
34
```

(5)

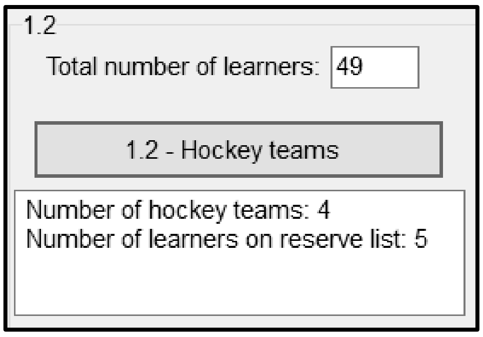
1.2 **Button [1.2 - Hockey teams]**

Learners who are interested in playing hockey are divided into teams of 11 players. A total of 11 players represents a full team in hockey. The remainder of the learners who could not make up a full team will be placed onto a reserve list.

A constant variable which is declared as PLAYERS = 11 has been provided.

Write code to do the following:

- Extract the total number of learners from the edit box **edtQ1_2**.
- Use the total number of learners and the constant PLAYERS to calculate the following:
    - Number of hockey teams made up of 11 players
    - Number of learners on the reserve list



```
Number of hockey teams: 4
Number of learners on reserve list: 5
```

(9)

1.3 **Button [1.3 - Calculate]**

The value of $\boldsymbol{d}$ must be calculated using the formula below:

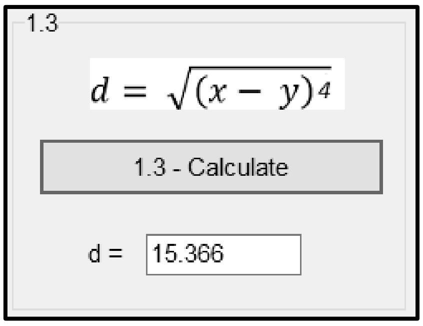
$$d = \sqrt{(x - y)^4}$$

Code has been provided to assign values to the variables **rX** and **rY** which represent the **x** and **y** values in the formula.

Write code to do the following:

- Use the provided **rX** and **rY** variables and appropriate mathematical functions to calculate the value of $\boldsymbol{d}$.
- Display the value of $\boldsymbol{d}$ in the edit box **edtQ1_3**, rounded to THREE decimal places.

Example of output for provided values rX = 12.46 and rY = 8.54:



15.366

(5)

1.4 **Button [1.4 - Marathon results]**

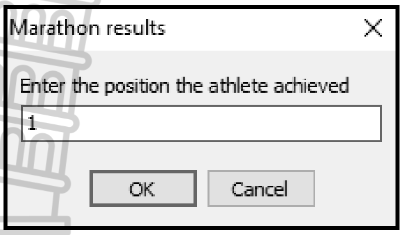
Participants who complete a marathon and finish in one of the top 20 positions will receive a gold, silver or bronze medal, based on their finishing position. All other participants who finish after the 20th position will receive a participation certificate.

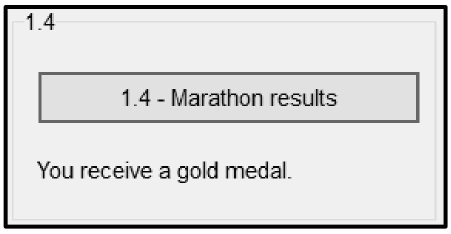
Code has been provided using an input dialogue box for the user to enter the finishing position of a participant. The user input is assigned to the variable **iPosition**.

Write code that uses a case statement to display an appropriate message in the label **lblQ1_4** based on the information in the table below.

Finishing position	**Award**
1	Gold medal
2 and 3	Silver medal
4 to 20	Bronze medal
After 20	Participation certificate

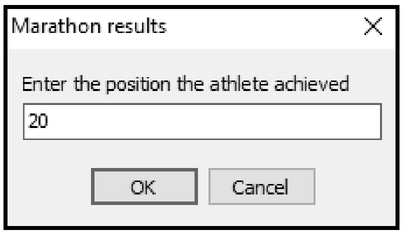
Example of input and output if a learner achieved the first place:

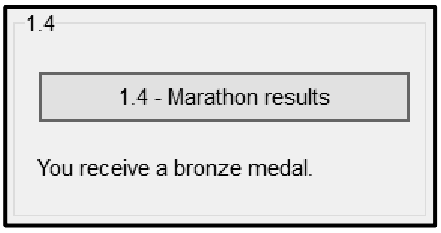




You receive a gold medal.

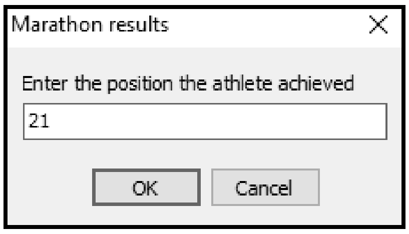
Example of input and output if a learner achieved the 20th place:

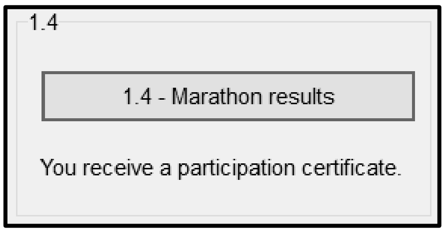




You receive a bronze medal.

Example of input and output if a learner achieved the 21st place:





You receive a participation certificate.

(6)

1.5 **Button [1.5 - Average mark]**

A text file called **'Details.txt'** contains the names and marks of learners in the following format:

```
<Name>#<Mark>
```

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

```
Erinn Stansell#87
Michael Dinjes#90
Gabrielle Wadhams#23
Mirelda Berendsen#47
Elmore Skalls#32
```

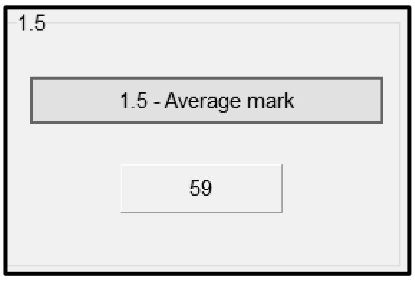
Write code to do the following:

- Open the text file **Details.txt**, read through the lines of text in the text file and determine the average mark for all the learners.
- Display the average mark on the panel **pnlQ1_5**, rounded to the nearest integer.

NOTE: Your code must work correctly for any number of lines of text in the file.

HINT: Use the position of the hash character (#) to extract the mark for each learner.

Example of output:



(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 A: 40

SECTION B

QUESTION 2: DATABASE PROGRAMMING

Universities rely on effective administration and communication to facilitate the smooth running of their processes.

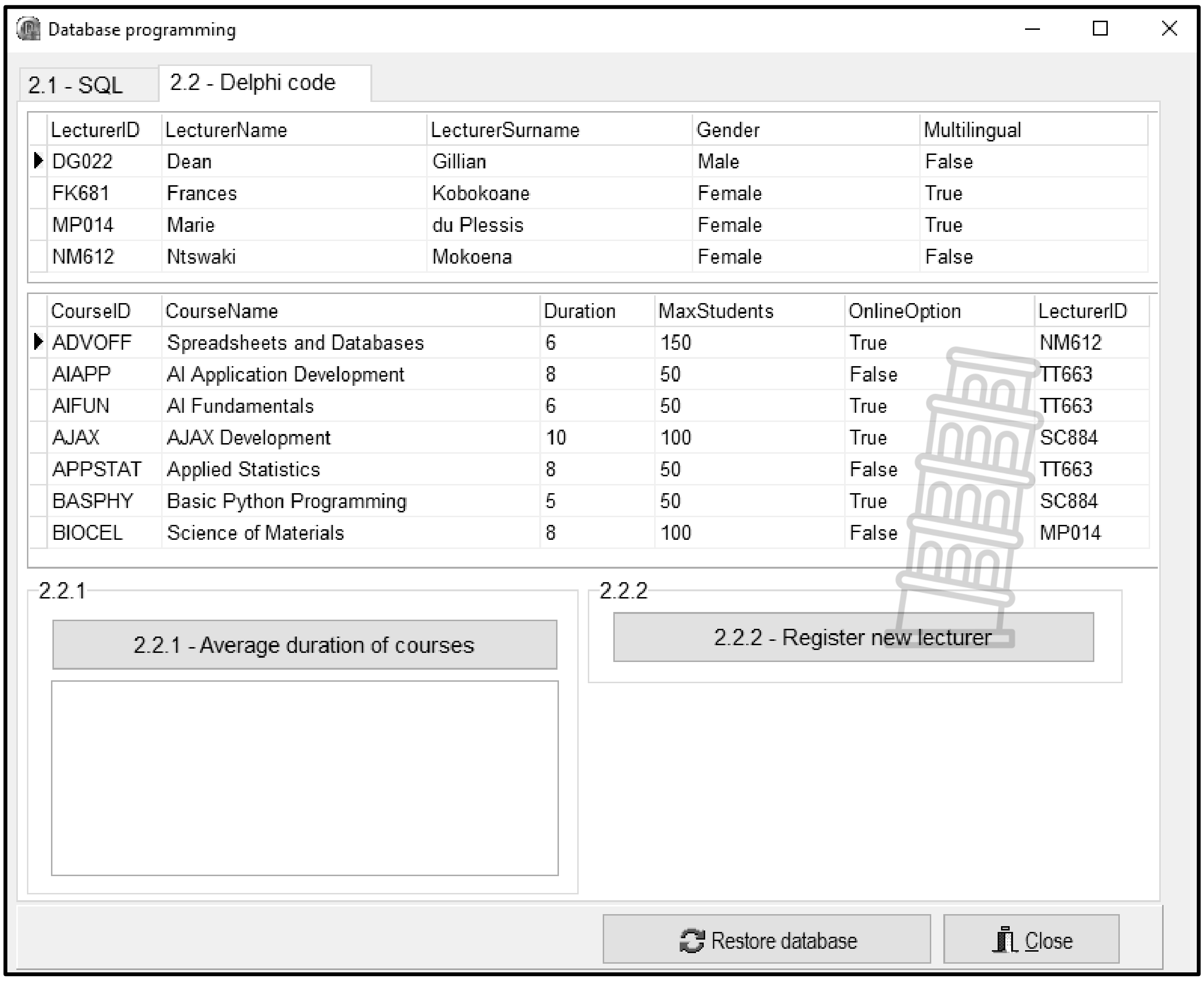
A database called **UniversityDB.mdb**, which contains information on the different lecturers and the courses they teach, has been developed.

The database contains two tables, **tblLecturers** and **tblCourses**.

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

Do the following:

- Open the incomplete project file called **Question2_P.dpr** in the **Question 2** folder.
- Enter your examination number as a comment in the first line of the **Question2_U.pas** unit file.
- Compile and execute the program. The program has no functionality currently. The contents of the tables are displayed, as shown below on the selection of tab sheet **2.2 - Delphi code**.



LecturerID	LecturerName	LecturerSurname	Gender	Multilingual
DG022	Dean	Gillian	Male	False
FK681	Frances	Kobokoane	Female	True
MP014	Marie	du Plessis	Female	True
NM612	Ntswaki	Mokoena	Female	False

CourseID	CourseName	Duration	MaxStudents	OnlineOption	LecturerID
ADVOFF	Spreadsheets and Databases	6	150	True	NM612
AIAPP	AI Application Development	8	50	False	TT663
AIFUN	AI Fundamentals	6	50	True	TT663
AJAX	AJAX Development	10	100	True	SC884
APPSTAT	Applied Statistics	8	50	False	TT663
BASPHY	Basic Python Programming	5	50	True	SC884
BIOCEL	Science of Materials	8	100	False	MP014

2.2.1
2.2.1 - Average duration of courses
2.2.2
2.2.2 - Register new lecturer
Restore database
Close

- 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.

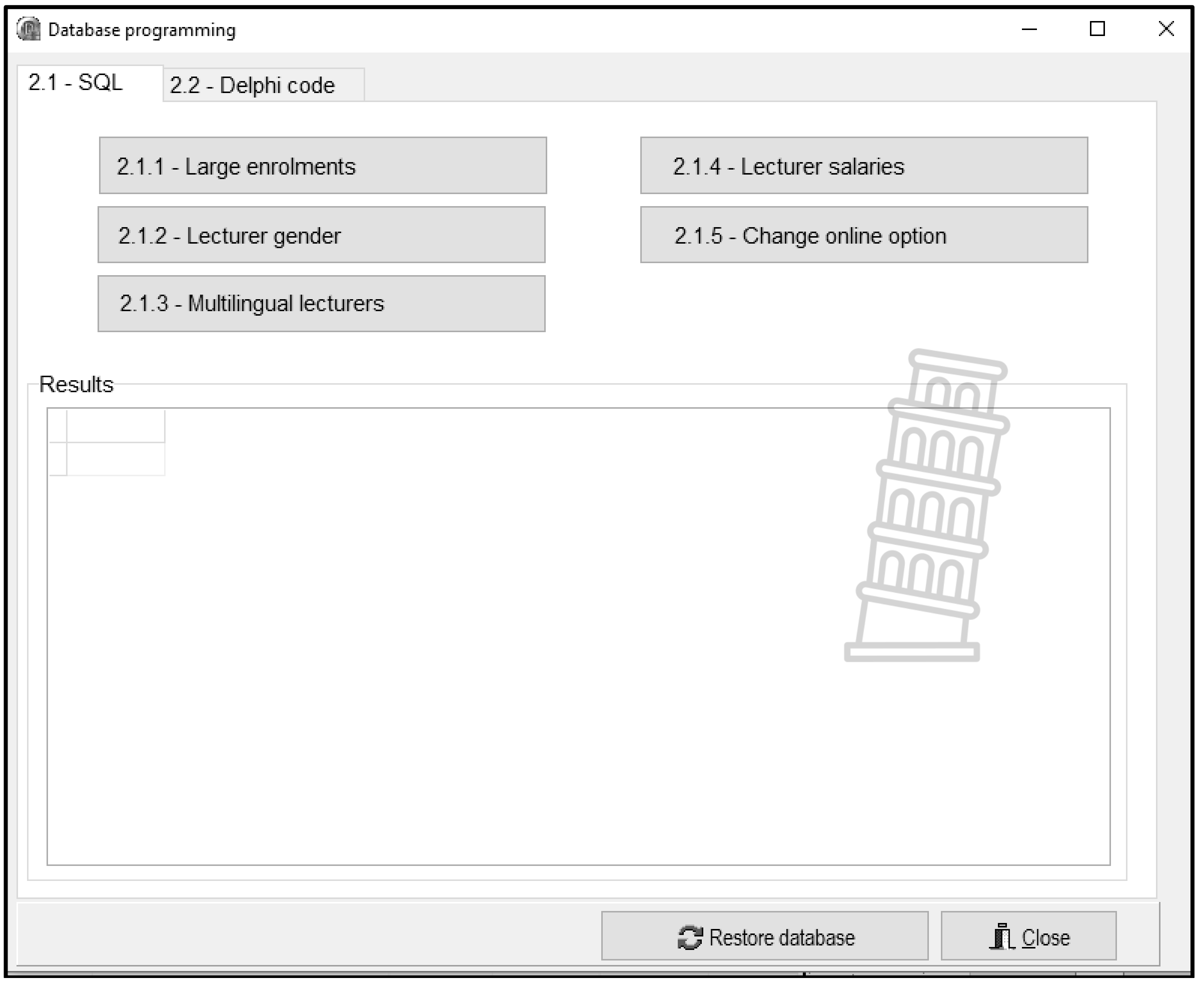
NOTE:

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

Variable	**Data type**	**Description**
`tblLecturers`	`TADOTable`	Refers to the table **tblLecturers**
`tblCourses`	`TADOTable`	Refers to the table **tblCourses**

2.1 **Tab sheet [2.1 - SQL]**

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



NOTE:

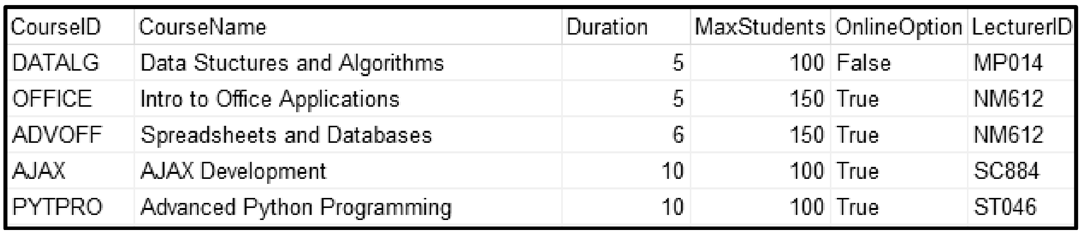
- Use ONLY SQL statements to answer QUESTION 2.1.1 to QUESTION 2.1.5.
- Code to execute the SQL statements and display the results of the queries is provided. The SQL statements assigned to the variables **sSQL1**, **sSQL2**, **sSQL3**, **sSQL4** and **sSQL5** are incomplete.

Complete the SQL statements to perform the tasks described in QUESTION 2.1.1 to QUESTION 2.1.5 below.

2.1.1 **Button [2.1.1 - Large enrolments]**

Display the details of all the courses in the **tblCourses** table, that can accommodate 100 or more students.

Example of output of the first five records:



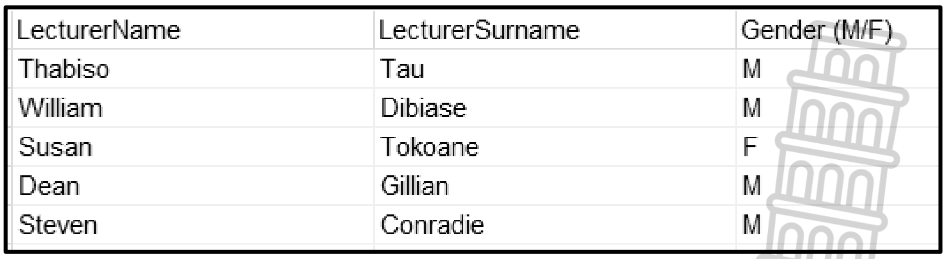
CourseID	CourseName	Duration	MaxStudents	OnlineOption	LecturerID
DATALG	Data Stuctures and Algorithms	5	100	False	MP014
OFFICE	Intro to Office Applications	5	150	True	NM612
ADVOFF	Spreadsheets and Databases	6	150	True	NM612
AJAX	AJAX Development	10	100	True	SC884
PYTPRO	Advanced Python Programming	10	100	True	ST046

(3)

2.1.2 **Button [2.1.2 - Lecturer gender]**

Display the **LecturerName**, **LecturerSurname** and the first letter of the gender of all the lecturers. Display the gender using a column heading called **Gender (M/F)**.

Example of output of the first five records:



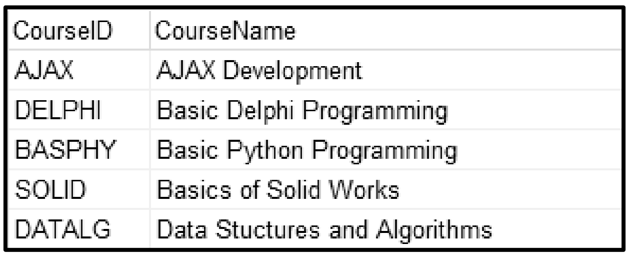
LecturerName	LecturerSurname	Gender (M/F)
Thabiso	Tau	M
William	Dibiase	M
Susan	Tokoane	F
Dean	Gillian	M
Steven	Conradie	M

(4)

2.1.3 **Button [2.1.3 - Multilingual lecturers]**

Display the **CourseID** and **CourseName** of all courses that have a multilingual lecturer. The results must be sorted in alphabetical order according to the course name.

Example of output of the first five records:



CourseID	CourseName
AJAX	AJAX Development
DELPHI	Basic Delphi Programming
BASPHY	Basic Python Programming
SOLID	Basics of Solid Works
DATALG	Data Stuctures and Algorithms

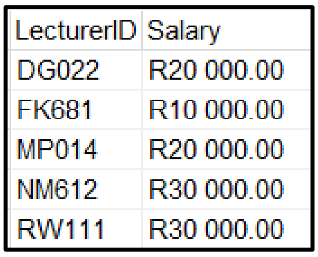
(6)

2.1.4 **Button [2.1.4 - Lecturer salaries]**

A lecturer's salary is determined by the number of courses facilitated by that lecturer. An amount of R10 000 is paid for each course facilitated by the lecturer. For example, if a lecturer facilitates three courses, the lecturer will get a total salary of R30 000.

Display the **LecturerID** and the total salary in a new column with the heading **Salary**, formatted to currency.

Example of output of the first five records:



LecturerID	Salary
DG022	R20 000.00
FK681	R10 000.00
MP014	R20 000.00
NM612	R30 000.00
RW111	R30 000.00

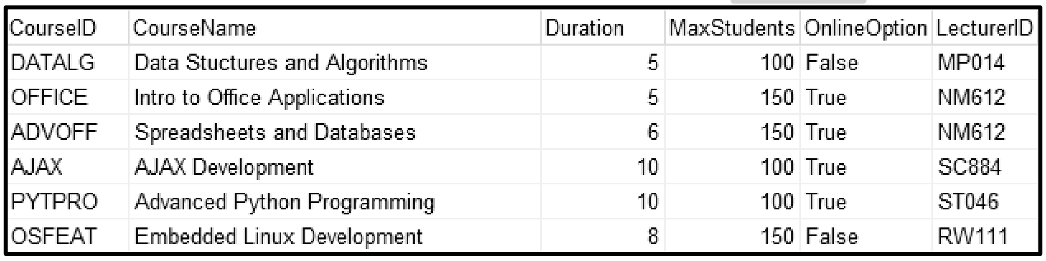
(5)

2.1.5 **Button [2.1.5 - Change online option]**

Write code to change the **OnlineOption** field to false if the **CourseName** field contains the word 'Programming'.

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

Example of the first six records of the **tblCourses** table before the online option was changed:



CourseID	CourseName	Duration	MaxStudents	OnlineOption	LecturerID
DATALG	Data Structures and Algorithms	5	100	False	MP014
OFFICE	Intro to Office Applications	5	150	True	NM612
ADVOFF	Spreadsheets and Databases	6	150	True	NM612
AJAX	AJAX Development	10	100	True	SC884
PYTPRO	Advanced Python Programming	10	100	True	ST046
OSFEAT	Embedded Linux Development	8	150	False	RW111

Example of the first six records of the **tblCourses** table after the online option was changed to false for the programming courses:

CourseID	CourseName	Duration	MaxStudents	OnlineOption	LecturerID
DATALG	Data Stuctures and Algorithms	5	100	False	MP014
OFFICE	Intro to Office Applications	5	150	True	NM612
ADVOFF	Spreadsheets and Databases	6	150	True	NM612
AJAX	AJAX Development	10	100	True	SC884
PYTPRO	Advanced Python Programming	10	100	False	ST046
OSFEAT	Embedded Linux Development	8	150	False	RW111

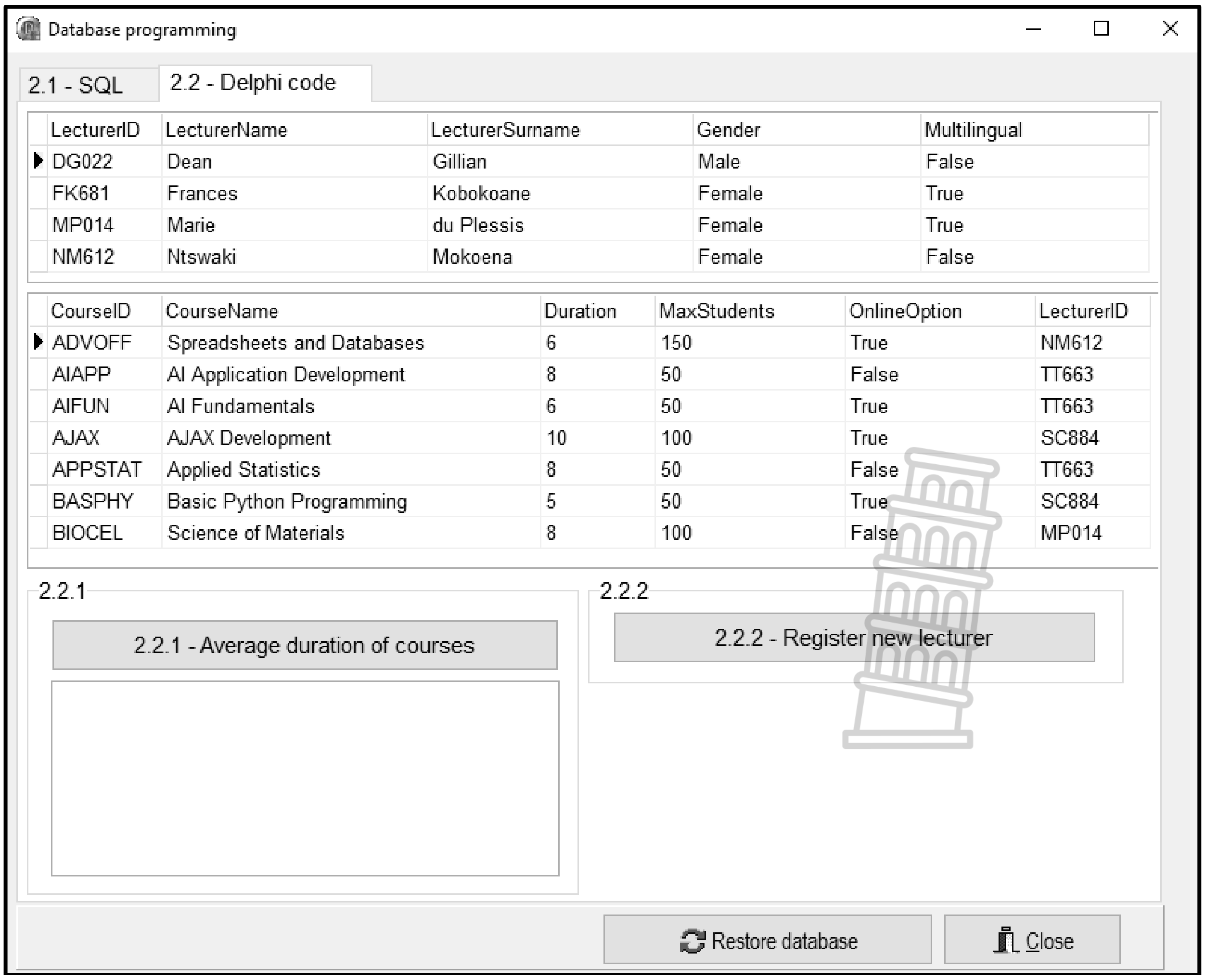
(4)

2.2 Tab sheet [2.2 - Delphi code]

NOTE:

- Use ONLY Delphi programming code to answer QUESTION 2.2.1 and QUESTION 2.2.2.
- NO marks will be awarded for SQL statements in QUESTION 2.2.

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



LecturerID	LecturerName	LecturerSurname	Gender	Multilingual
DG022	Dean	Gillian	Male	False
FK681	Frances	Kobokoane	Female	True
MP014	Marie	du Plessis	Female	True
NM612	Ntswaki	Mokoena	Female	False

CourseID	CourseName	Duration	MaxStudents	OnlineOption	LecturerID
ADVOFF	Spreadsheets and Databases	6	150	True	NM612
AIAPP	AI Application Development	8	50	False	TT663
AIFUN	AI Fundamentals	6	50	True	TT663
AJAX	AJAX Development	10	100	True	SC884
APPSTAT	Applied Statistics	8	50	False	TT663
BASPHY	Basic Python Programming	5	50	True	SC884
BIOCEL	Science of Materials	8	100	False	MP014

2.2.1 **Button [2.2.1 - Average duration of courses]**

The workload of all lecturers must be determined in order to issue them with a fair number of courses to facilitate.

Use the **redQ2_2_1** component to display the output.

Code has been provided to clear the **redQ2_2_1** component.

Write code to do the following:

- Display the **LecturerID**, **LecturerName** and **LecturerSurname** as a heading in the following format:
  `<LecturerID>: <LecturerName> <LecturerSurname>`
- Display a numbered list of course names facilitated by each lecturer.
- Calculate and display the average duration of the courses facilitated by each lecturer, formatted to TWO decimal places.

Example of output for the first two lecturers:

```
DG022: Dean Gillian
1. Intro to Computer Literacy
2. Fundamentals of Physics
Average duration of courses:   7.50

FK681: Frances Kobokoane
1. Basics of Solid Works
Average duration of courses:   6.00
```

(14)

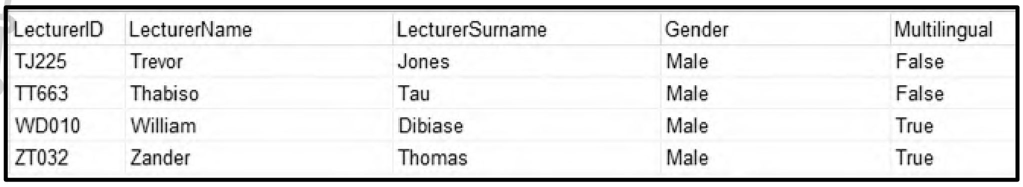
2.2.2 **Button [2.2.2 - Register new lecturer]**

When a new lecturer is appointed, their details must be added to the database.

Write code to add a new record to the **tblLecturers** table using the following details:

- `LecturerID – ZT032`
- `LecturerName – Zander`
- `LecturerSurname – Thomas`
- `Gender – Male`
- `Multilingual – True`

Example of records in the **tblLecturers** table, which indicates that the record has been added to the table successfully:



LecturerID	LecturerName	LecturerSurname	Gender	Multilingual
TJ225	Trevor	Jones	Male	False
TT663	Thabiso	Tau	Male	False
WD010	William	Dibiase	Male	True
ZT032	Zander	Thomas	Male	True

(4)

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

TOTAL SECTION B: 40

SECTION C

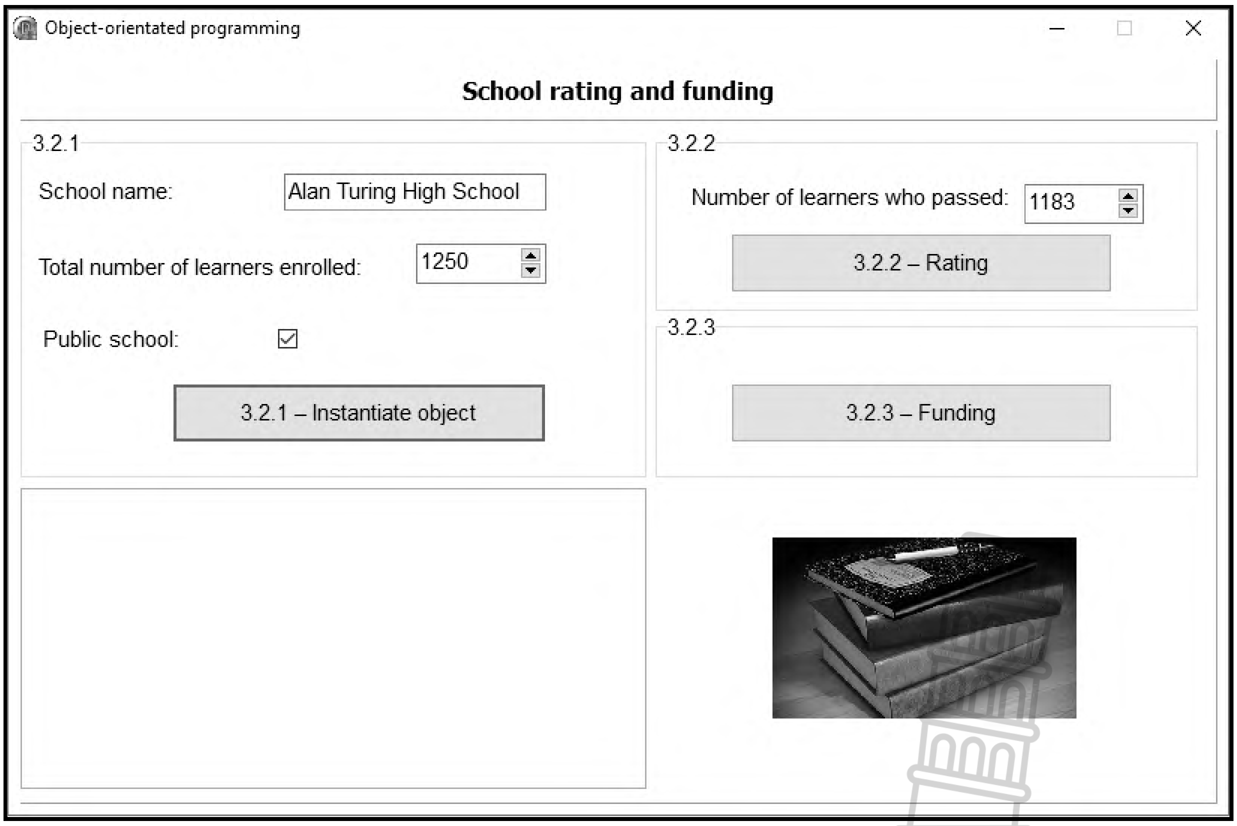
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

The local schools' district office requires a record of schools in the district and a report on the results of the schools to determine the amount of funding that each school will receive.

Do the following:

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

Example of the graphical user interface (GUI):


Object-orientated programming
School rating and funding
3.2.1
School name:
Alan Turing High School
Total number of learners enrolled:
1250
Public school:
☑
3.2.1 – Instantiate object
3.2.2
Number of learners who passed:
1183
3.2.2 – Rating
3.2.3
3.2.3 – Funding

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

3.1 The provided incomplete object class (**TSchool**) contains the declaration of four attributes which describe a **School** object.

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

Attribute	**Data type**	**Description**
fSchoolName	String	The name of the school
fTotalLearners	Integer	The total number of learners enrolled at the school
fPublicSchool	Boolean	True if it is a public school, otherwise false
fRating	Char	A rating (**A**, **B**, **C** or **Z**) assigned to each school based on the percentage pass rate of the school: A – 80% or higher B – From 60% up to 79% C – Lower than 60% Z – Rating not assigned

An incomplete **constructor** method has been provided.

NOTE: You are NOT allowed to add any additional attributes or user-defined methods, unless you are instructed to do so explicitly in one of the questions.

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

3.1.1 **Constructor Create** has been provided with three parameters.

Write code to complete the constructor method as follows:

- Assign the values in the parameter list to the corresponding attributes, **fSchoolName**, **fTotalLearners** and **fPublicSchool**.
- Assign a default rating of 'Z' to the **fRating** attribute. (3)

3.1.2 Write an accessor method called **getPublicSchool** for the **fPublicSchool** attribute. (2)

3.1.3 Every year the rating of the school will be updated based on the percentage pass rate of the school. The percentage pass rate is calculated using the total number of learners who passed and the total number of learners at the school.

The information in the table below is used to determine the school's rating:

Rating	**Percentage pass rate**
A	80% or higher
B	From 60% up to 79%
C	Lower than 60%

Write code for a method called **updateRating** to do the following:

- Receive the total number of learners who passed as a parameter value.
- Use the parameter value and the **fTotalLearners** attribute to calculate the percentage pass rate of the school.
- Use the table provided to determine and set the value of the **fRating** attribute. (8)

3.1.4 An amount of R145,50 is allocated per learner enrolled at the school.

Write code for a method called **calcFunding** to calculate and return the total amount of funding that will be received by the school.

The funding is calculated using the following formula:

$$funding = learners\ enrolled \times 145.50$$ (4)

3.1.5 Write code for a **toString** method to return a string which describes the object. The Boolean attribute **fPublicSchool** must be used to determine whether the phrase 'Public school' or 'Private school' must be added to the string.

The format of the string to be returned is shown below.

```
<fSchoolName>
------------------------
Total number of learners: < fTotalLearners >
Rating: <fRating>
Public school/Private school
```

Example:

```
Alan Turing High School
------------------------
Total number of learners: 1250
Rating: Z
Public school
```

(7)

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 **objSchool**.

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

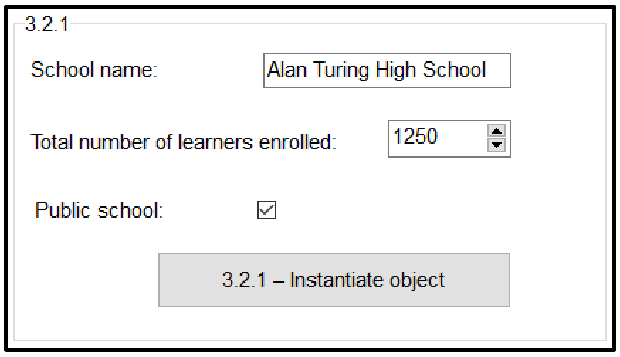
3.2.1 **Button [3.2.1 – Instantiate object]**

The user must enter the school name in the edit box **edtQ3_2_1**, select the total number of learners in the spin edit **spnQ3_2_1** and tick the check box **chbQ3_2_1** if the school is a public school.

Write code to do the following:

- Extract the school name from edit box **edtQ3_2_1**, the total number of learners in the spin edit **spnQ3_2_1** and determine whether the check box **chbQ3_2_1** has been checked (ticked) or not.
- Use the information extracted to instantiate a new **School** object.
- Use the **toString** method to display the information of the **School** object in the rich edit **redQ3**.

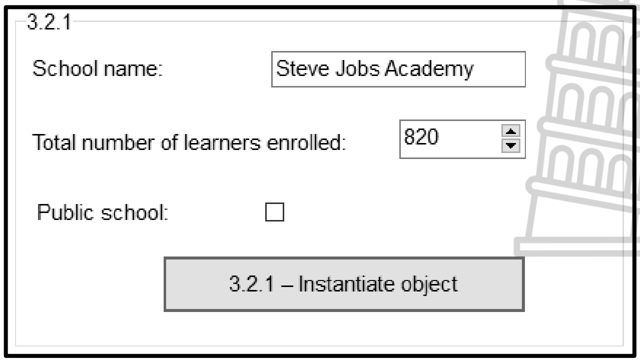
Example of input and output for a public school:


3.2.1
School name:
Total number of learners enrolled:
Public school:
3.2.1 – Instantiate object

School name: Alan Turing High School
Total number of learners enrolled: 1250
Public school: ✓

```
Alan Turing High School
-----------------------------
Total number of learners: 1250
Rating: Z
Public school
```

Example of input and output for a private school:


3.2.1
School name:
Total number of learners enrolled:
Public school:
3.2.1 – Instantiate object

School name: Steve Jobs Academy
Total number of learners enrolled: 820
Public school: ☐

```
Steve Jobs Academy
-----------------------------
Total number of learners: 820
Rating: Z
Private school
```

(7)

3.2.2 **Button [3.2.2 – Rating]**

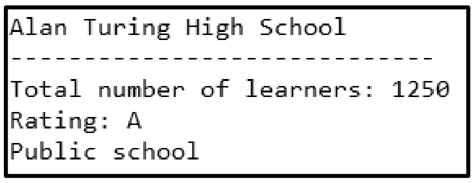
The rating of a school is determined by the percentage pass rate of the school.

The spin edit **spnQ3_2_2** must be used to enter the total number of learners who passed.

Write code to do the following:

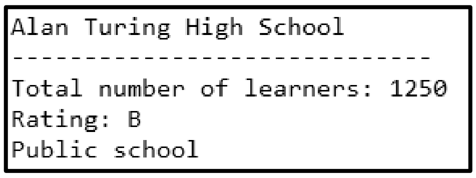
- Call the **updateRating** method using the value extracted from the spin edit **spnQ3_2_2** as an argument.
- Call the **toString** method to display the updated information of the **School** object in the rich edit **redQ3**.

Example of output if the total number of learners is 1 250 and the total number of learners who passed is 1 183:



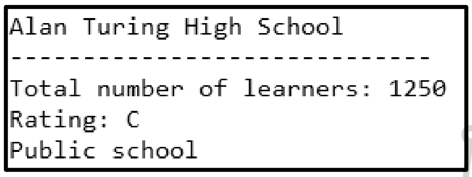
```
Alan Turing High School
-----------------------------
Total number of learners: 1250
Rating: A
Public school
```

Example of output if the total number of learners is 1 250 and the total number of learners who passed is 999:



```
Alan Turing High School
-----------------------------
Total number of learners: 1250
Rating: B
Public school
```

Example of output if the total number of learners is 1 250 and the total number of learners who passed is 500:



```
Alan Turing High School
-----------------------------
Total number of learners: 1250
Rating: C
Public school
```

(4)

3.2.3 **Button [3.2.3 – Funding]**

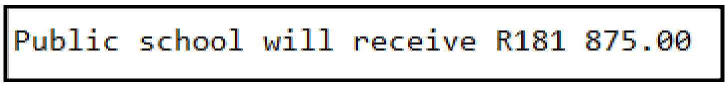
Funding will be available to public schools only. Use the rich edit **redQ3** to display the output.

Write code to do the following:

Use the relevant method to test if the school is a public school.

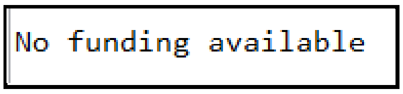
- If the school is a public school, call the **calcFunding** method to display the amount of funding the school will receive, formatted as currency with a message as indicated on the next page.
- If the school is not a public school, display a suitable message indicating that no funding is available.

Example of output if the school is a public school and has a total number of 1 250 learners:



```
Public school will receive R181 875.00
```

Example of output if the school is NOT a public school:



```
No funding available
```

(5)

- 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

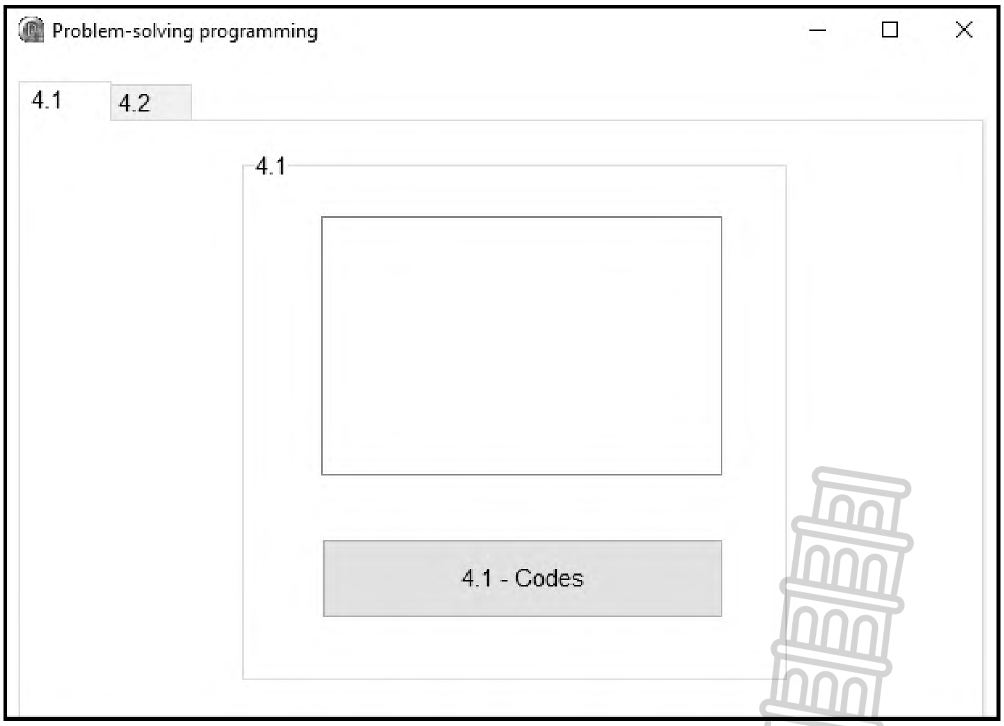
Do the following:

- 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.
- Two tab sheets named **tshQ4_1** and **tshQ4_2** are provided on the graphical user interface (GUI).

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

4.1 **Tab sheet [4.1]**

Example of the graphical user interface (GUI) for tab sheet 4.1:



You have been provided with an array called **arrCodes** that contains five codes. Each code contains letters, digits and special characters.

```
arrCodes: array [1 .. 5] of String =

    ('An7J*Q#D&N','pL78K#$.%BV','89@FGh0&Y56#$Q','Bn4m321&*#T',
    'P2QwER%$#a');
```

A special character refers to a character that is not a letter and not a digit.

Button [4.1 - Codes]

Write code to do the following:

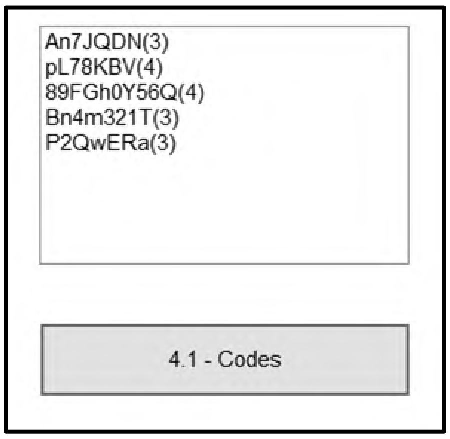
Remove all the special characters from each code in the array to create a new code. Count how many special characters were removed from each code.

Add the new code and the number of special characters deleted from it to the list box **lstQ4_1** in the following format:

```
<New code>(number of special characters deleted)
```

NOTE: Your code must work for any set of data in the array.

Example of output for the codes currently in the array:



```
An7JQDN(3)
pL78KBV(4)
89FGh0Y56Q(4)
Bn4m321T(3)
P2QwERa(3)
```

(12)

4.2 **Tab sheet [4.2]**

A school timetable has five days and seven periods per day.

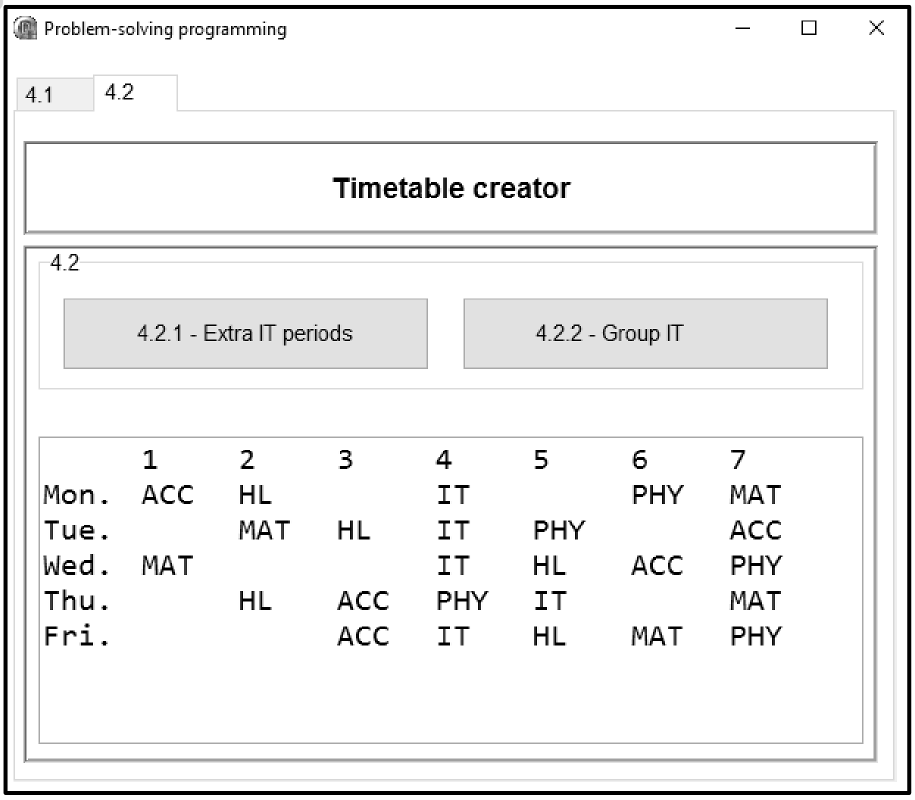
You have been provided with the following declarations:

```
arrDays: array [1 .. 5] of String =
     ('Mon.', 'Tue.', 'Wed.', 'Thu.', 'Fri.');

arrTimeTable: array [1 .. 5, 1 .. 7] of String;

arrSubjectCodes: array [1 .. 5] of String =
     ('IT', 'HL', 'ACC', 'PHY', 'MAT');
```

NOTE: Code has been provided to do the following:

- Populate the **arrTimeTable** array by placing the five subject codes IT, HL, ACC, PHY and MAT randomly in the array, once per day for the five days. There will be two free periods (empty spaces) each day.
- Display the seven periods per day for each of the five days on the timetable (Monday to Friday).

Example of the graphical user interface (GUI) for tab sheet 4.2:



```
      1     2     3     4     5     6     7
Mon.  ACC   HL          IT          PHY   MAT
Tue.        MAT   HL    IT    PHY         ACC
Wed.  MAT               IT    HL    ACC   PHY
Thu.        HL    ACC   PHY   IT          MAT
Fri.              ACC   IT    HL    MAT   PHY
```

4.2.1 **Button [4.2.1 - Extra IT periods]**

The school principal decided to add four extra periods, once per day (Monday to Thursday), for the subject code 'IT' to the timetable.

The extra IT periods must be placed in the **arrTimeTable** array from Monday to Thursday at the **first available period** using the code 'IT'.

Example of output after the extra IT periods have been placed in the first available periods from Monday to Thursday:

```
      1     2     3     4     5     6     7
Mon.  ACC   HL    IT    IT          PHY   MAT
Tue.  IT    MAT   HL    IT    PHY         ACC
Wed.  MAT   IT          IT    HL    ACC   PHY
Thu.  IT    HL    ACC   PHY   IT          MAT
Fri.              ACC   IT    HL    MAT   PHY
```

NOTE: By default, there will be six free periods available on the timetable after all the subject codes have been placed, including the extra IT periods.

Due to the random placement of subject codes, your output may differ from the example above. (6)

4.2.2 **Button [4.2.2 - Group IT]**

The IT teacher requested to have both IT periods placed consecutively (one after the other), from Monday to Thursday.

Write code to do the following:

Rearrange the periods on each day to place the two IT periods one after the other. The first IT period for the day must remain in its place, while the second IT period must be swapped with another period in order to be placed next to the first IT period.

Example of output after the IT periods have been grouped together:

```
      1     2     3     4     5     6     7
Mon.  ACC   HL    IT    IT          PHY   MAT
Tue.  IT    IT    HL    MAT   PHY         ACC
Wed.  MAT   IT    IT          HL    ACC   PHY
Thu.  IT    IT    ACC   PHY   HL          MAT
Fri.              ACC   IT    HL    MAT   PHY
```

(12)

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

TOTAL SECTION D: 30
GRAND TOTAL: 150

INFORMATION TECHNOLOGY P1

DATABASE INFORMATION FOR QUESTION 2:

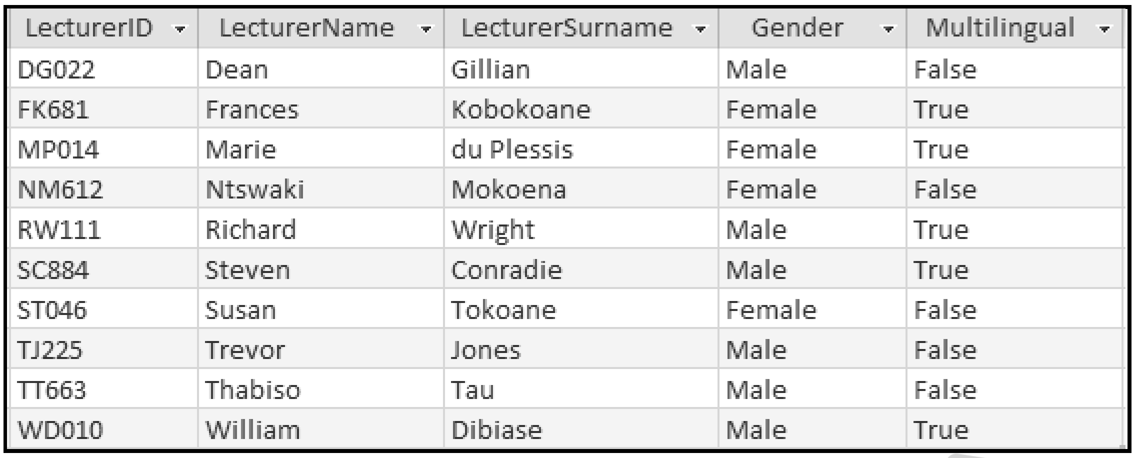
The design of the database tables for the database **UniversityDB** is as follows:

Table: **tblLecturers**

This table contains the details of the lecturers.

Field name	**Data type**	**Description**
LecturerID	Text (5)	A unique ID used to identify a lecturer
LecturerName	Text (25)	The first name of the lecturer
LecturerSurname	Text (25)	The last name of the lecturer
Gender	Text (6)	The gender of the lecturer
Multilingual	Boolean	A field that indicates whether a lecturer can present a course in more than one language

Example of the records van the **tblLecturers** table:



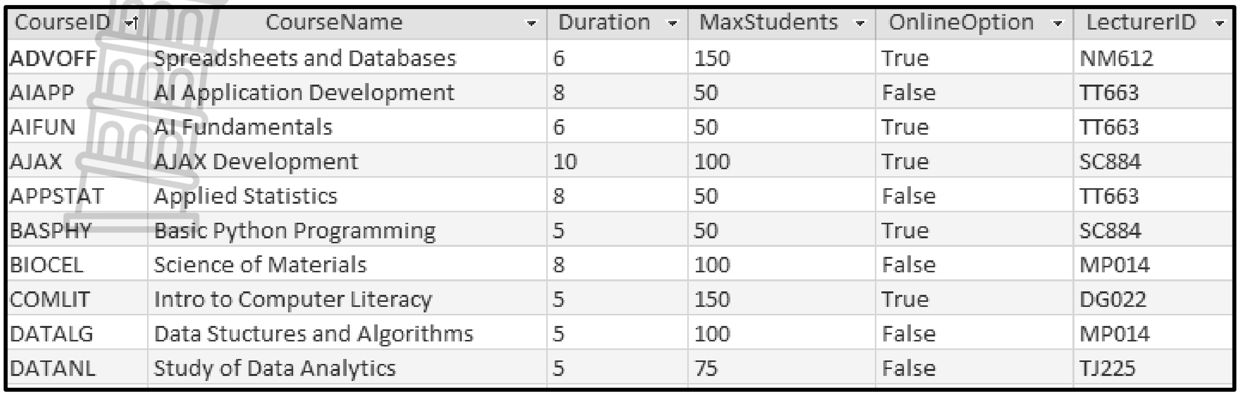
LecturerID	LecturerName	LecturerSurname	Gender	Multilingual
DG022	Dean	Gillian	Male	False
FK681	Frances	Kobokoane	Female	True
MP014	Marie	du Plessis	Female	True
NM612	Ntswaki	Mokoena	Female	False
RW111	Richard	Wright	Male	True
SC884	Steven	Conradie	Male	True
ST046	Susan	Tokoane	Female	False
TJ225	Trevor	Jones	Male	False
TT663	Thabiso	Tau	Male	False
WD010	William	Dibiase	Male	True

Table: **tblCourses**

This table contains information on the courses offered at the university.

Field name	**Data type**	**Description**
CourseID	Text (10)	A unique ID used to identify a course
CourseName	Text (35)	The name of the course
Duration	Number	The duration of the course (in weeks)
MaxStudents	Number	The maximum number of students allowed in the course
OnlineOption	Boolean	A field that indicates whether an online option is available to students
LecturerID	Text (5)	The ID of the lecturer that presents the course

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

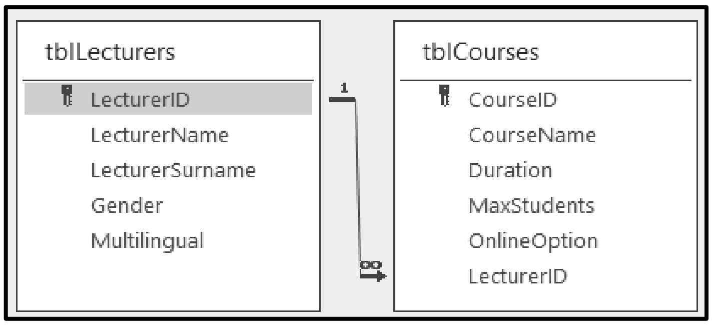


CourseID	CourseName	Duration	MaxStudents	OnlineOption	LecturerID
ADVOFF	Spreadsheets and Databases	6	150	True	NM612
AIAPP	AI Application Development	8	50	False	TT663
AIFUN	AI Fundamentals	6	50	True	TT663
AJAX	AJAX Development	10	100	True	SC884
APPSTAT	Applied Statistics	8	50	False	TT663
BASPHY	Basic Python Programming	5	50	True	SC884
BIOCEL	Science of Materials	8	100	False	MP014
COMLIT	Intro to Computer Literacy	5	150	True	DG022
DATALG	Data Stuctures and Algorithms	5	100	False	MP014
DATANL	Study of Data Analytics	5	75	False	TJ225

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

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2023

MARKING GUIDELINES

MARKS: 150

Approved:

These marking guidelines consist of 24 pages.

GENERAL INFORMATION:

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

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

- **Annexures A, B, C** and **D** (pages 3 to 10) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 11 to 24) 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 10) should be made for each learner and completed during the marking session.

ANNEXURE A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
1.1	**Button [1.1 – Display name and age]** Retrieve name from edtQ1_1 and store in sName variable ✓ Retrieve age from spnQ1_1 and store in iAge variable ✓ Display using an output dialog box the name ✓ with #13/#10/sLineBreak ✓ for next line the age converted to string ✓	**5**	
1.2	**Button [1.2 – Hockey teams]** Extract the number of learners from the edit box, ✓ converted to an integer/real ✓ Calculate the number of teams Number of learners DIV ✓ PLAYERS ✓ (using constant) Calculate the number of reserves Number of learners MOD ✓ PLAYERS ✓ Display the number of teams converted to String ✓ in the memo memQ1_2 ✓ Also display the number of reserves ✓ **ALSO ACCEPT**: Alternative mathematical functions that will provide the correct answer. **NOTE**: The given Constant PLAYERS must be used at least once.	**9**	
1.3	**Button [1.3 – Calculate]** `Formula: d = Sqrt ✓ (power ✓ ((rX - rY),4)✓)` Display the value of **d** in edtQ1_3 ✓ formatted to 3 decimal places ✓ **ALSO ACCEPT**: Alternative mathematical functions that will provide the correct answer. **DO NOT ACCEPT** hardcoding instead of mathematical functions.	**5**	

1.4	**Button [1.4 – Marathon results]** case iPosition of ✓ 1: ✓ Display 'You receive a gold medal' ✓ 2,3: Display 'You receive a silver medal' ✓ 4..20: Display 'You receive a bronze medal' ✓ Else Display 'You receive a participation certificate' ✓ End // case **NOTE**: The first two marks for the structure of the case statement will be lost when multiple if statements are used.	**6**	
1.5	**Button [1.5 – Average mark]** Declare file variable (tFile) ✓ Initialise Total and iCount variables to 0 ✓ AssignFile (tFile, 'Details.txt') ✓ Reset (tFile) ✓ While not end of file ✓ Begin Read line from text file ✓ Increment iCount ✓ Find the position of the # delimiter ✓ Extract mark from line ✓ using correct indexes ✓ Convert mark to integer/real ✓ and add to total ✓ End while Close the file Calculate average using Total and iCount ✓ Round display to the nearest integer ✓ Display average mark in pnlQ1_5 ✓	**15**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – DATABASE PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**		

2.1.1	**Button [2.1.1 – Large enrolments]**
	`SELECT * ✓ FROM tblCourses ✓`
	`WHERE MaxStudents > 99 ✓`
	
	Alternative: `MaxStudents >= 100`
		3	

2.1.2	**Button [2.1.2 – Lecturer gender]**
	`SELECT LecturerName, LecturerSurname, ✓`
	`LEFT (Gender, 1) ✓ AS [Gender (M/F)] ✓`
	`FROM tblLecturers ✓`
	
	Alternative: `MID (Gender, 1, 1)`
		4	

2.1.3	**Button [2.1.3 – Multilingual lecturers]**
	`SELECT CourseID, CourseName`
	`FROM tblLecturers, tblCourses ✓`
	`WHERE tblLecturers.LecturerID ✓`
	`            = tblCourses.LecturerID ✓ AND ✓`
	`Multilingual = TRUE ✓`
	`ORDER BY CourseName ✓`
	
	Alternatives: `Multilingual`
	`Multilingual LIKE True`
	`ORDER BY 2`
		6	

2.1.4	**Button [2.1.4 – Lecturer salaries]**
	`SELECT LecturerID,`
	`FORMAT(Count(*) ✓ * 10000 ✓, "CURRENCY") ✓`
	`AS Salary`
	`FROM tblCourses ✓`
	`GROUP BY LecturerID ✓`
	
	Note: Count can use any field name instead of *.
		5	

2.1.5	**Button [2.1.5 – Change online option]**
	`UPDATE tblCourses ✓ SET OnlineOption = FALSE ✓`
	`WHERE CourseName LIKE ✓ "%Programming%" ✓`
	
	Alternative: `CourseName LIKE "%Programming"`
		4	

	Subtotal:	**22**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**Database Manipulation**		

2.2.1	**Button [2.2.1 – Average duration of courses]**
	
	Go to the first record in tblLecturers ✓
	Loop through tblLecturers ✓
	    Display heading using LecturerID, LecturerName,
	    and LecturerSurname in the correct format ✓
	
	    Initialise Counter and Sum variables ✓
	
	    Go to the first record in tblCourses ✓
	    Loop through tblCourses ✓
	        Test if (tblLecturers ['LecturerID'] =
	        tblCourses['LecturerID']) ✓
	            Increment Counter ✓ and
	                    add duration to Sum ✓
	            Display the counter value and course name ✓
	        tblCourses.Next ✓
	    End loop (tblCourses)
	
	    Calculate average duration: Sum / Counter ✓
	    Display average duration formatted to two decimals ✓
	
	    tblLecturers.Next ✓
	End loop (tblLecturers)
		14	

2.2.2	**Button [2.2.2 – Register new lecturer]** tblLecturers.Insert; ✓ tblLecturers['LecturerID'] := 'ZT032'; tblLecturers['LecturerName'] := 'Zander'; tblLecturers['LecturerSurname'] := 'Thomas'; } ✓✓ tblLecturers['Gender'] := 'Male'; tblLecturers['Multilingual'] := True; tblLecturers.Post; ✓ **Alternatives:** Append instead of insert Any navigation command instead of Post **Allocating field values:** 1 mark for correctly using all field names 1 mark for using correct values Subtract 1 mark for each error to a maximum of two marks	**4**	
	Subtotal:	**18**	
	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**Constructor Create** Set attributes (fSchoolName, fTotalLearners, fPublicSchool) ✓ to correct parameters ✓ Assign 'Z' to fRating ✓	**3**	
3.1.2	**Function getPublicSchool** Function heading with Boolean value as return data type ✓ result = fPublicSchool ✓	**2**	

3.1.3	**Procedure updateRating**
	
	Procedure heading ✓ with integer parameter ✓
	
	passPercentage = parameter / fTotalLearners * 100 ✓
	
	if passPercentage >= 80 ✓
	    fRating = 'A' ✓
	else if passPercentage >= 60 ✓
	    fRating = 'B' ✓
	else
	    fRating = 'C' ✓
	
	Also accept other solutions
	
	Note: The range 79 – 80 can also be dealt with as a separate range – being included or excluded
		8	

3.1.4	**Function calcFunding** Function heading with real return data type ✓ Result ✓ = fTotalLearners ✓ * 145.50 ✓	**4**	

3.1.5	**Function toString** with string return data type Build string with fSchoolName and '-------------' on next line ✓ Add 'Total number of learners: ' and fTotalLearners to the string ✓ Add 'Rating: ' and fRating to the string ✓ If fPublicSchool ✓ Add 'Public school' ✓ Else Add 'Private school' ✓ Return string ✓	7	
	Subtotal: Object class	**24**	

QUESTION 3: MARKING GRID – FORM CLASS

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 – Instantiate Object]** Extract school name from edtQ3_2_1 ✓ Extract number of learners from spnQ3_2_1 ✓ Extract public school from chbQ3_2_1 ✓ objSchool ✓ := TSchool.Create ✓ Use three arguments in correct order ✓ `(sSchoolName, iNumLearners, bPublicSchool)` Display object objSchool in redQ3 using toString method ✓	7	
3.2.2	**Button [3.2.2 – Rating]** Extract number of learners that passed from spnQ3_2_2 ✓ Call updateRating ✓ with correct argument ✓ Display objSchool in redQ3 using toString method ✓	4	

3.2.3	**Button [3.2.3 – Funding]**
	
	Test if getPublicSchool = TRUE ✓
	    Display in redQ3 with message ✓
	    Using calcFunding method ✓
	            Formatted to currency ✓
	Else
	    Display message – 'No funding available' ✓
		5	

	Subtotal Form class:	**16**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX MARKS**	**LEARNER'S MARKS**
4.1	**Button [4.1 – Codes]** Loop from 1 to length of array (or 5) ✓ Initialise sLine ← blank string ✓ Loop from 1 to ✓ the length of code ✓ Test if character in the code ✓ is a letter ('a'..'z' OR 'A'..'Z') ✓ or a digit ('0'..'9') ✓ Join the character to the sLine output string ✓ End inner loop Determine the number of special characters removed Length(arrCodes[i]) ✓ – Length(sLine) ✓ // Or use a counter in the inner loop Add the sLine code ✓ to the list box in the correct format with brackets and number of characters removed ✓ End outer loop **Concepts**: Outer i loop (1 to 5) (1) Inner j loop (1 to length(arrCodes[i])) (2) Test character [i][j] (1) is digit OR letter (2) Remove characters / build string (2) Counting characters removed (2) Add new code to lstQ4_1 (1) and number of characters removed (1)	**12**	
4.2.1	**Button [4.2.1 – Extra IT periods]** Loop iCnt from 1 to 4 (Monday – Thursday) ✓ Set counter to 1 ✓ While the cell is not blank ✓ Increment counter by 1 ✓ Assign 'IT' ✓ to arrTimeTable[iCnt, counter] ✓ End loop **Concepts**: Loop through the days 1 to 4 (1) // Also accept 1 to 5 Conditional loop/Break statement with for loop (1) Determine index of (1) First empty space in a row (1) Assign a new value to empty space (2)	**6**	

4.2.2	**Button [4.2.2 – Group IT]**
	```
	Loop I from 1 to 4 (Monday–Thursday) ✓
	   Initialise Counter ✓
	   Loop J from 1 to length of arrTimeTable[I] ✓
	      If arrTimeTable[I, J] = 'IT' ✓
	         If Counter = 1 ✓
	            Store index J ($J_1$) at first occurrence of 'IT' ✓
	         If Counter = 2 ✓
	            Store index J ($J_2$)  at second occurrence of 'IT' ✓
	      // swap other subject code with IT
	      Set sTemp ✓ to arrTimeTable[I, $J_1$+1] ✓
	      Set arrTimeTable[I, $J_1$+1] to arrTimeTable[I,$J_2$] ✓
	      Set arrTimeTable[I, $J_2$] to sTemp ✓
	   End inner loop
	End outer loop
	```
	Concepts:
	
	Loop through the rows (1 to 4) // 1
	  Determine the position of the first occurrence of IT:
	  Initialise/Create variable to store first position // 1
	  Loop through the columns // 1
	    Test if the cell value = 'IT' // 1
	      Save/Determine the index/position of first occurrence // 2
	  Determine the position of the second occurrence of IT:
	      Save/Determine the index/position of second occurrence // 2
	
	Swap the subject after the first occurrence of IT with the second occurrence of IT // 4
		12	

	TOTAL SECTION D:	**30**	

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, StdCtrls, ExtCtrls, Spin, pngimage, Math;

type
  TfrmQuestion1 = class(TForm)
    grpQ1_2: TGroupBox;
    btnQ1_2: TButton;
    grpQ1_1: TGroupBox;
    edtQ1_1: TEdit;
    spnQ1_1: TSpinEdit;
    lblQ1_1_Name: TLabel;
    lblQ1_1_Age: TLabel;
    btnQ1_1: TButton;
    grpQ1_3: TGroupBox;
    imgQ1_3: TImage;
    btnQ1_3: TButton;
    edtQ1_3: TEdit;
    Label3: TLabel;
    grpQ1_5: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    cmbQ1_5: TComboBox;
    btnQ1_5: TButton;
    grpQ1_: TGroupBox;
    btnQ1_4: TButton;
    pnlQ1_5: TPanel;
    Label6: TLabel;
    edtQ1_2: TEdit;
    memQ1_2: TMemo;
    lblQ1_4: TLabel;
    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}
```

```
// ==========================================================================
// 1.1 Display name and age                                          5 marks
// ==========================================================================

procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
var
  sName: String;
  iAge: integer; // Provided code
begin
  sName := edtQ1_1.Text;
  iAge := spnQ1_1.Value;
  ShowMessage(sName + #13 + IntToStr(iAge));
end;

// ==========================================================================
// 1.2 Hockey teams                                                  9 marks
// ==========================================================================

procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
const
  PLAYERS = 11;
var
  iNumLearners, iNumTeams, iNumReserves: integer;
begin
  // Provided code
  memQ1_2.Clear;

  iNumLearners := StrToInt(edtQ1_2.Text);
  iNumTeams := iNumLearners DIV PLAYERS;
  iNumReserves := iNumLearners MOD PLAYERS;
  memQ1_2.Lines.Add('Number of hockey teams: ' + IntToStr(iNumTeams));
  memQ1_2.Lines.Add('Number of learners on reserve list: ' + IntToStr
      (iNumReserves));
end;

// ==========================================================================
// 1.3 Calculate                                                     5 marks
// ==========================================================================

procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  rX, rY: real; // Provided code
  rD: real;
begin
  // Provided code
  rX := 12.46;
  rY := 8.54;

  rD := sqrt(power((rX - rY), 4));
  edtQ1_3.Text := FloatToStrF(rD, ffFixed, 8, 3);
end;
```

```
// ============================================================================
// 1.4 Marathon results                                                6 marks
// ============================================================================

procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  iPosition: integer; // Provided code
begin
// Provided code
  iPosition := StrToInt(InputBox('Marathon results',
      'Enter the position the athlete achieved', '1'));
// -----------------------------------------------------
  case iPosition of
    1:        lblQ1_4.Caption := 'You receive a gold medal.';
    2, 3:     lblQ1_4.Caption := 'You receive a silver medal.';
    4 .. 20: lblQ1_4.Caption := 'You receive a bronze medal.'
    else lblQ1_4.Caption := 'You receive a participation certificate.';
  end;
end;

// ============================================================================
// 1.5 Average mark                                                   15 marks
// ============================================================================

procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  tFile: TextFile;
  sLine: String;
  iTotal, iMark, iCount, iPosHash: integer;
  rAverage: real;
begin
  iTotal := 0;
  iCount := 0;
  AssignFile(tFile, 'Details.txt');
  Reset(tFile);
  while NOT(eof(tFile))do
   begin
      readln(tFile, sLine);
      iPosHash := pos('#', sLine);
      iMark := StrToInt(copy(sLine, iPosHash + 1));
      iTotal := iTotal + iMark;
      inc(iCount);
   end;
  closeFile(tFile);
  rAverage := iTotal / iCount;
  pnlQ1_5.Caption := FloatToStrF(rAverage, ffFixed, 3, 0);
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
// ============================================================================
// 2.1 - Section: SQL statements
// ============================================================================

// ============================================================================
//  2.1.1 Large courses                                                3 marks
// ============================================================================

    sSQL1 := 'SELECT * ' +
             'FROM tblCourses ' +
             'WHERE MaxStudents > 99';

// ============================================================================
//  2.1.2 Lecturer gender                                              4 marks
// ============================================================================

    sSQL2 := 'SELECT LecturerName, LecturerSurname, ' +
             'Left(Gender, 1) AS [Gender (M/F)]' +
             'FROM tblLecturers';

// ============================================================================
//  2.1.3 Multilingual lecturers                                      6 marks
// ============================================================================

    sSQL3 := 'SELECT CourseID, CourseName ' +
             'FROM tblLecturers , tblCourses ' +
             'WHERE (tblLecturers.LecturerID = tblCourses.LecturerID) AND
              (Multilingual = True) ' +
             'ORDER BY CourseName';
// ============================================================================
//  2.1.4 Lecturer salaries                                            5 marks
// ============================================================================

    sSQL4 := 'SELECT LecturerID, ' +
             'FORMAT(Count(*)*10000, "CURRENCY") ' +
             'AS [Salary] ' +
             'FROM tblCourses ' +
             'GROUP BY LecturerID';

// ============================================================================
//  2.1.5 Change online option                                         4 marks
// ============================================================================

    sSQL5 := 'UPDATE tblCourses Set OnlineOption = FALSE ' +
             'WHERE CourseName Like "%Programming%"';
```

```
// ============================================================================
// 2.2 - Section: Delphi code
// ============================================================================

// ============================================================================
// 2.2.1 Average duration of courses                                   14 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  iCountCourses, iSumDuration : integer;
  rAverageDuration : real;
begin

  redQ2_2_1.Clear;

  tblLecturers.First;
  while NOT tblLecturers.Eof do
    begin
      redQ2_2_1.Lines.Add(tblLecturers['LecturerID'] + ': ' +
                          tblLecturers['LecturerName'] + ' ' +
                          tblLecturers['LecturerSurname']);
      tblCourses.First;
      iCountCourses := 0;
      iSumDuration := 0;
      while NOT tblCourses.Eof do
        begin
          if tblLecturers['LecturerID'] =
                           tblCourses['LecturerID'] then
            begin
              inc(iCountCourses);
              redQ2_2_1.Lines.Add(IntToStr(iCountCourses) + '. ' +
                                   tblCourses['CourseName']);
              iSumDuration := iSumDuration +tblCourses['Duration'];
            end;
          tblCourses.Next;
        end;
      rAverageDuration := iSumDuration / iCountCourses;
      redQ2_2_1.Lines.Add('Average duration of courses: ' + #9 +
                     FloatToStrF(rAverageDuration, ffFixed, 8, 2) + #13);
      tblLecturers.Next;
    end;
end;
// ============================================================================
// 2.2.2 Register new lecturer                                          4 marks
// ============================================================================
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
begin
  tblLecturers.Insert;
  tblLecturers['LecturerID'] := 'ZT032';
  tblLecturers['LecturerName'] := 'Zander';
  tblLecturers['LecturerSurname'] := 'Thomas';
  tblLecturers['Gender'] := 'Male';
  tblLecturers['Multilingual'] := True;
  tblLecturers.Post;
end;
```

```
// ==============================================================================
// {$ENDREGION}
// ==============================================================================
// {$REGION 'Provided code: Setup DB connections - DO NOT CHANGE!'}
// ==============================================================================

procedure TfrmQuestion2.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  // Disconnects from database and closes all open connections
  dbCONN.dbDisconnect;
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  redQ2_2_1.Paragraph.TabCount := 2;
  redQ2_2_1.Paragraph.Tab[0] := 100;
  redQ2_2_1.Paragraph.Tab[1] := 150;
  redQ2_2_1.Paragraph.Tab[2] := 200;
end;

procedure TfrmQuestion2.FormShow(Sender: TObject);
begin
  // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblLecturers := dbCONN.tblOne;
  tblCourses := dbCONN.tblMany;
  dbCONN.setupGrids(dbgLecturers, dbgCourses, dbgrdSQL);
  pgcDBAdmin.ActivePageIndex := 0;
end;
// ==============================================================================
// {$ENDREGION}

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
unit School_U;

interface

type
  TSchool = class(TObject)
  private
  var
    fSchoolName: String;
    fTotalLearners: Integer;
    fPublicSchool: boolean;
    fRating: char;
  public
    // Provide code
    constructor create(sSchoolName: String; iTotalLearners: integer;
      bPublicSchool: Boolean);
    // Code here

    function getPublicSchool: boolean;
    procedure updateRating(iLearnersPassed: integer);
    function calcFunding: real;
    function toString: String;
  end;

implementation

uses
  SysUtils, Math;
// ==========================================================================
// 3.1.1 Constructor Create                                          3 marks
// ==========================================================================

constructor TSchool.create(sSchoolName: String; iTotalLearners: integer;
  bPublicSchool: boolean);
begin
  // 3.1.1 Contructor Create
  fSchoolName := sSchoolName;
  fTotalLearners := iTotalLearners;
  fPublicSchool := bPublicSchool;
  fRating := 'Z';
end;
// ==========================================================================
// 3.1.2 Function getPublicSchool                                  2 marks
// ==========================================================================

function TSchool.getPublicSchool: boolean;
begin
  Result := fPublicSchool;
end;
```

```
// ============================================================================
// 3.1.3 Procedure updateRating                                         8 marks
// ============================================================================

procedure TSchool.updateRating(iLearnersPassed: integer);
var
  rPassPer: real;
  cRating: char;
begin
  rPassPer := iLearnersPassed / fTotalLearners * 100;

  if rPassPer >= 80 then
  begin
    fRating := 'A';
  end
  else if (rPassPer >= 60) AND (rPassPer < 80) then
  begin
    fRating := 'B';
  end
  else
  begin
    fRating := 'C';
  end;
end;
// ============================================================================
// 3.1.4 Function calcFunding                                           4 marks
// ============================================================================

function TSchool.calcFunding: real;
begin
  Result := 145.50 * fTotalLearners;
end;

// ============================================================================
// 3.1.5 Function toString                                              7 marks
// ============================================================================

function TSchool.toString: String;
var
  sOutStr : String;
begin
  sOutStr := fSchoolName + #13 + '-----------------------------' +#13;
  sOutStr := sOutStr + 'Total number of learners: ' +
                         IntToStr(fTotalLearners) + #13;
  sOutStr := sOutStr + 'Rating: ' + fRating + #13;
  if fPublicSchool then
      sOutStr := sOutStr + 'Public school '  + #13
  else
      sOutStr := sOutStr + 'Private school '  + #13;
  result := sOutStr;
end;

end.
```

Main Form Unit

```
unit Question3_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, CheckLst, ExtCtrls, Buttons, Spin, ComCtrls, jpeg;

type
  TfrmQuestion3 = class(TForm)
    gbxQ3_2_1: TGroupBox;
    gbxQ3_2_3: TGroupBox;
    redQ3: TRichEdit;
    btnQ3_2_1: TButton;
    gbxQ3_2_2: TGroupBox;
    btnQ3_2_2: TButton;
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ3_2_3: TButton;
    Image1: TImage;
    Label6: TLabel;
    edtQ3_2_1: TEdit;
    Label2: TLabel;
    spnQ3_2_1: TSpinEdit;
    chbQ3_2_1: TCheckBox;
    Label1: TLabel;c
    sedQ3_2_2: TSpinEdit;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
  private

  public

  end;

var
  frmQuestion3: TfrmQuestion3;

implementation

{$R *.dfm}

uses
  School_U;

var
  objSchool: TSchool;
```

```
// ============================================================================
// 3.2.1 Instantiate object                                            7 marks
// ============================================================================

procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
  sSchoolName : String;
  iNumLearners : integer;
  bPublicSchool : boolean;
begin
  // Provided code
  redQ3.Clear;

  // 3.2.1 Instantiate object
  sSchoolName := edtQ3_2_1.Text;
  iNumLearners := spnQ3_2_1.Value;
  bPublicSchool := chbQ3_2_1.Checked;

  objSchool := TSchool.create(sSchoolName, iNumLearners, bPublicSchool);
  redQ3.Lines.Add(objSchool.toString);
end;
// ============================================================================
// 3.2.2 Rating                                                        4 marks
// ============================================================================

procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  iNumPassed : integer;
begin
  // Provided code
  redQ3.Clear;

  // 3.2.2 Rating
  iNumPassed := spnQ3_2_2.Value;
  objSchool.updateRating(iNumPassed);
  redQ3.Lines.Add(objSchool.toString);
end;

// ============================================================================
// 3.2.3 Funding                                                       5 marks
// ============================================================================

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
begin

  // 3.2.3 Funding
  if objSchool.getPublicSchool then
    redQ3.Lines.Add('Public school will receive ' + FloatToStrF
        (objSchool.calcFunding, ffCurrency, 8, 2))
  else
      redQ3.Lines.Add('No funding available ');
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;

interface

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

type
  TfrmQuestion4 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ4_2_2: TButton;
    redQ4: TRichEdit;
    GroupBox1: TGroupBox;
    btnQ4_2_1: TButton;
    pgcQ4: TPageControl;
    tshQ4_1: TTabSheet;
    tshQ4_2: TTabSheet;
    btnQ4_1: TButton;
    lstQ4_1: TListBox;
    GroupBox2: TGroupBox;
    procedure btnQ4_2_2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure btnQ4_2_1Click(Sender: TObject);
    procedure btnQ4_1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    procedure populate;
    procedure display;
  end;
var
  frmQuestion4: TfrmQuestion4;

  // Provided code for Question 4.1
  arrCodes: array [1 .. 5] of String =
              ('An7J*Q#D&N', 'pL78K#$.%BV',
              '89@FGh0&Y56#$Q','Bn4m321&*#T','P2QwER%$#a' );

  // Provided code for Question 4.2
  arrDays: array [1 .. 5] of String = ('Mon.', 'Tue.', 'Wed.', 'Thu.',
'Fri.');
  arrSubjectCodes: array [1 .. 5] of String =
                                      ('IT','HL', 'ACC', 'PHY', 'MAT' );
  arrTimeTable: array [1 .. 5, 1 .. 7] of String;

implementation
```

```
// =============================================================================
// 4.1 Codes                                                          12 marks
// =============================================================================
procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  I, J, iNumSpecChars: integer;
  sLine: String;
begin
  // 4.1 Codes
  for I := 1 to length(arrCodes) do
  begin
    sLine := '';
    for J := 1 to length(arrCodes[I]) do
    begin
      if arrCodes[I][J] IN ['A' .. 'Z', 'a' .. 'z', '0' .. '9'] then
      begin
        sLine := sLine + arrCodes[I][J];
      end;
    end;
    iNumSpecChars := length(arrCodes[I]) - length(sLine);
    lstQ4_1.Items.Add(sLine + '(' + intToStr(iNumSpecChars) + ')');
  end;
end;

// =============================================================================
// 4.2.1  Extra IT periods                                           6 marks
// =============================================================================
procedure TfrmQuestion4.btnQ4_2_1Click(Sender: TObject);
var
  iRow, iCol: integer;
begin
  // 4.2.1 Extra IT periods
  for iRow := 1 to 4 do
  begin
    iCol := 1;
    While NOT(arrTimeTable[iRow, iCol] = '') do
    begin
      inc(iCol);
    end;
    arrTimeTable[iRow, iCol] := 'IT';
  end;
  // Provided code
  display;
end;

// =============================================================================
// 4.2.2 Group IT                                                     12 marks
// =============================================================================
procedure TfrmQuestion4.btnQ4_2_2Click(Sender: TObject);
var
  I: integer;
  J: integer;
  iCount, iFirst, iSecond: integer;
  sTemp: String;

begin
```

```
  // 4.2.2 Group IT
  for I := 1 to 4 do
  begin
    iCount := 0;
    for J := 1 to 7 do
    Begin
      if arrTimeTable[I, J] = 'IT' then
      begin
        inc(iCount);
        if iCount = 1 then
          iFirst := J + 1;
        if iCount = 2 then
        begin
          iSecond := J;
          sTemp := arrTimeTable[I, iFirst];
          arrTimeTable[I, iFirst] := arrTimeTable[I, iSecond];
          arrTimeTable[I, iSecond] := sTemp;
        end;
      end;
    end;
  end;
  // Provided code
  display;
end;

// ==========================================================================
// Provided code - Do not change
// ==========================================================================

procedure TfrmQuestion4.populate;
var
  sSubjCode: String;
  iPeriod, iRand, iRow, iCol, iCnt: integer;
  arrLocal: array [1 .. 5] of String;
begin
  for iCnt := 1 to 5 do
  begin
    repeat
      iRand := RandomRange(1, 6);
      if length(arrLocal[iRand]) = 0 then
        arrLocal[iCnt] := arrSubjectCodes[iCnt];
    until length(arrLocal[iCnt]) > 0;
  end;
  for iCol := 1 to 5 do
  begin
    for iRow := 1 to 5 do
    begin
      repeat
        iRand := RandomRange(1, 8);
      until (arrTimeTable[iRow, iRand] = '');
      arrTimeTable[iRow, iRand] := arrLocal[iCol];
    end;
  end;
  display;
end;
```

```
procedure TfrmQuestion4.FormShow(Sender: TObject);
begin
  redQ4.Paragraph.TabCount := 9;
  redQ4.Paragraph.Tab[0] := 50;
  redQ4.Paragraph.Tab[1] := 100;
  redQ4.Paragraph.Tab[2] := 150;
  redQ4.Paragraph.Tab[3] := 200;
  redQ4.Paragraph.Tab[4] := 250;
  redQ4.Paragraph.Tab[5] := 300;
  redQ4.Paragraph.Tab[6] := 350;
  redQ4.Paragraph.Tab[7] := 400;
  redQ4.Paragraph.Tab[8] := 450;
  display;
  populate;
end;

procedure TfrmQuestion4.display;
var
  iRow, iCol, iCnt: integer;
  sLine: String;
begin
  sLine := #9;
  for iCnt := 1 to 7 do
    sLine := sLine + intToStr(iCnt) + #9;
  redQ4.Clear;
  redQ4.Lines.Add(sLine);
  for iRow := 1 to 5 do
  begin
    sLine := arrDays[iRow];
    for iCol := 1 to 7 do
    begin
      sLine := sLine + #9 + arrTimeTable[iRow, iCol];
    end;
    redQ4.Lines.Add(sLine);
  end;
end;

// ============================================================
// End of provided code
// ============================================================

end.
```