basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/
NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

MAY/JUNE 2025

MARKS: 150

TIME: 3 hours

**This question paper consists of 24 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 centre.

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: **DataJUN2025.exe**
- Click on the 'Extract' button.
- Enter the following password: **GAMES@J2025**

Once extracted, the following list of files will be available in the folder **DataJUN2025**:

Question 1:
DataQ1_3.txt
Question1_P.dpr
Question1_P.dproj
Question1_P.res
Question1_U.dfm
Question1_U.pas

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

Question 3:
Game_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

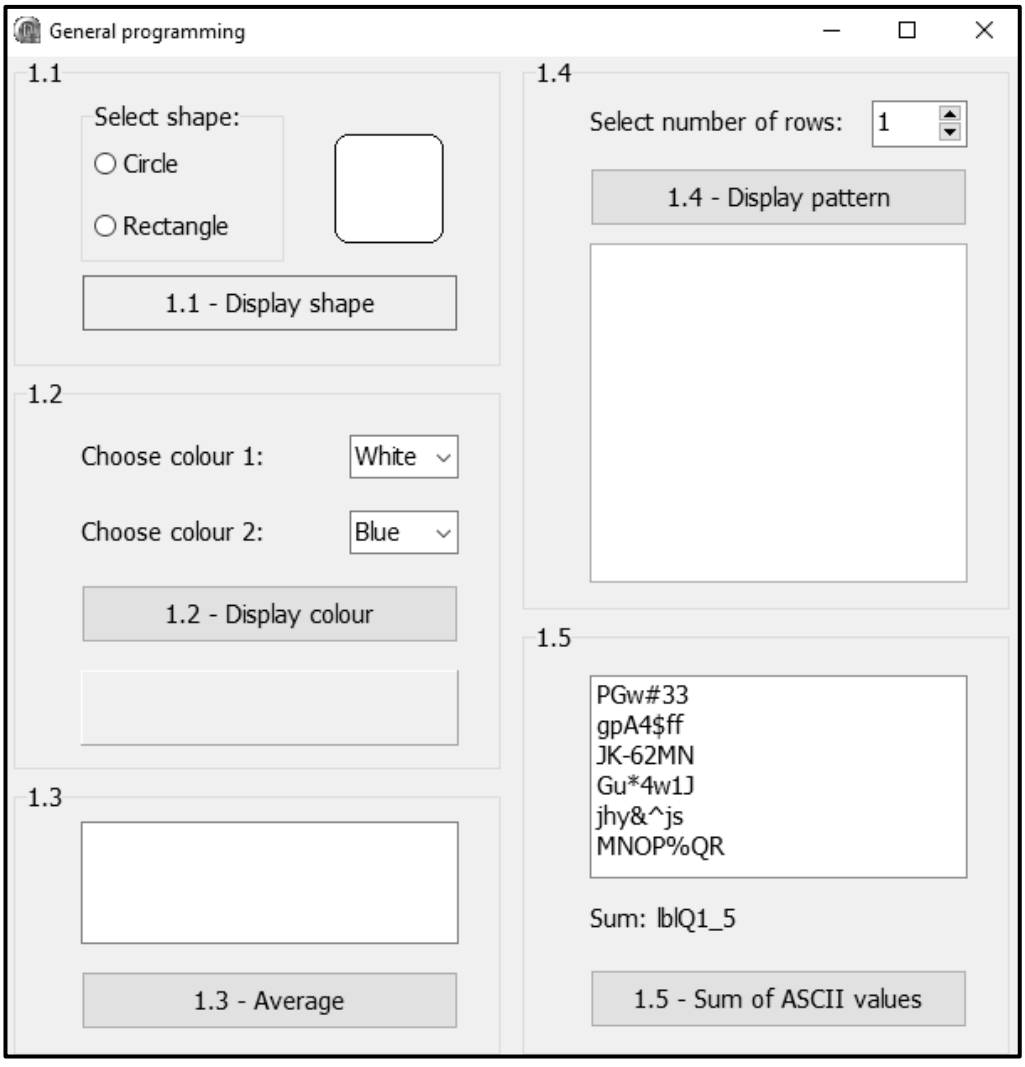
SECTION A

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 - Display shape]**

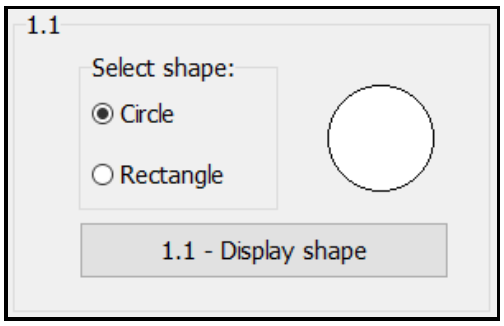
The user must select a shape from the radio group **rgpQ1_1**.

Code has been provided in the **OnCreate** event handler to set the visibility property of the **shpQ1_1** component to *true*.

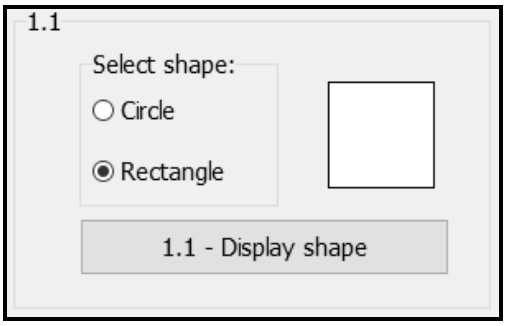
Write code to do the following:

- Extract the selected shape option from the radio group **rgpQ1_1**.
- Display the selected shape in the shape component **shpQ1_1**.

Example of output if the 'Circle' option is selected from **rgpQ1_1**:



Example of output if the 'Rectangle' option is selected from **rgpQ1_1**:



(5)

1.2 **Button [1.2 - Display colour]**

The user is required to select a colour from each of the provided combo boxes, **cmbColour1** and **cmbColour2**, respectively.

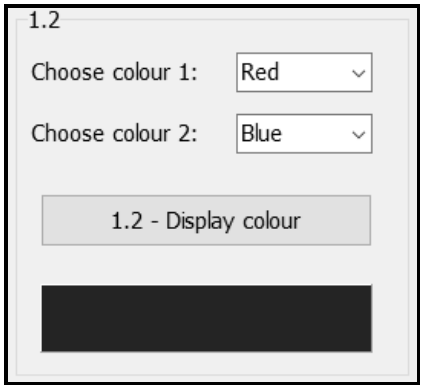
Code has been provided to do the following:

- Set the colour of the panel **pnlQ1_2** to `clBtnFace`
- Clear the caption of the panel **pnlQ1_2**

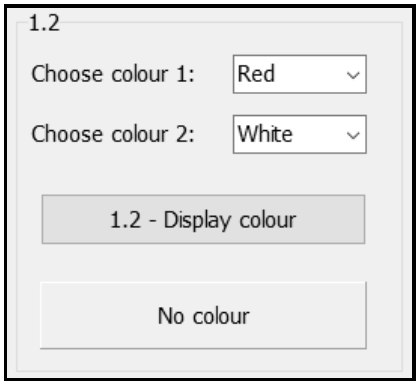
Write code to do the following:

- Extract and store the colours selected from the combo boxes, **cmbColour1** and **cmbColour2**.
- Determine the colour combination based on the following conditions:
  - If the combination of colours includes both 'Red' and 'Blue', set the colour of the panel **pnlQ1_2** to purple.
  - Otherwise, set the panel caption of **pnlQ1_2** to 'No colour'.

Example of output if the colours 'Red' and 'Blue' are selected. The colour of the panel changes to purple:



Example of output if any combination of colours, which excludes both 'Red' and 'Blue', are selected:



(8)

1.3 **Button [1.3 - Average]**

A text file called **DataQ1_3.txt** has been provided. The text file contains an unknown number of integer values.

Example of the first five lines of data in the text file **DataQ1_3.txt**:

```
22
21
23
42
124
```

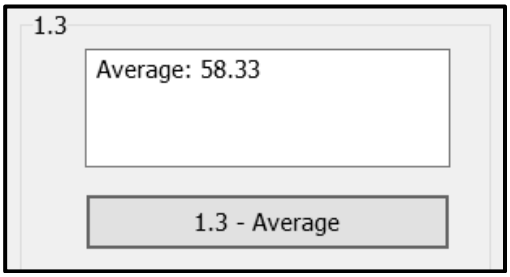
The average of the numbers in the text file must be calculated and displayed.

Write code to do the following:

Check whether or not the text file **DataQ1_3.txt** exists.

- If the file does NOT exist, display a suitable message in the memo component **memQ1_3** and close the program.
- If the file exists:
  - Use a conditional loop to read the numbers from the text file.
  - Determine the average of the numbers read from the text file.
  - Display the average rounded to TWO decimal places in the memo component **memQ1_3**.

Example of output:



(10)

1.4 **Button [1.4 - Display pattern]**

A pattern consisting of digits starting with the value of 1 in row 1, the value 2 in row 2, the value 3 in row 3 and so on, must be displayed. The number of rows and the number of digits per row will depend on the value selected by the user from the spin edit **spnQ1_4**.

Example of the pattern if the user selects the value of 4:

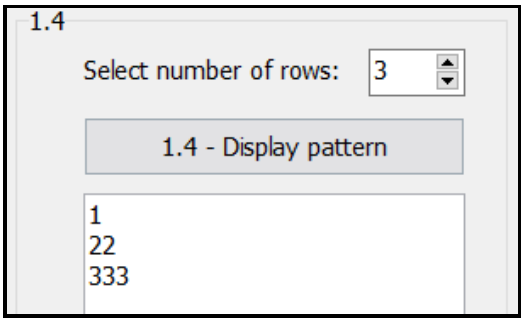
```
1
22
333
4444
```

Code has been provided to clear the rich edit component **redQ1_4**.

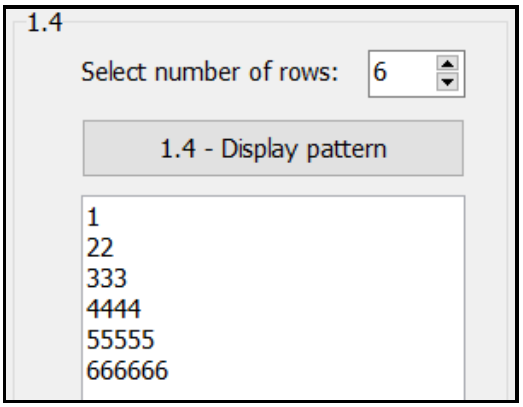
Write code to do the following:

- Extract the number of rows selected from the spin edit **spnQ1_4**.
- Use nested loops and the selected number of rows to compile and display the pattern described above, in the **redQ1_4** component.

Example of input and output if the number of rows selected was three:



Example of input and output if the number of rows selected was six:



(8)

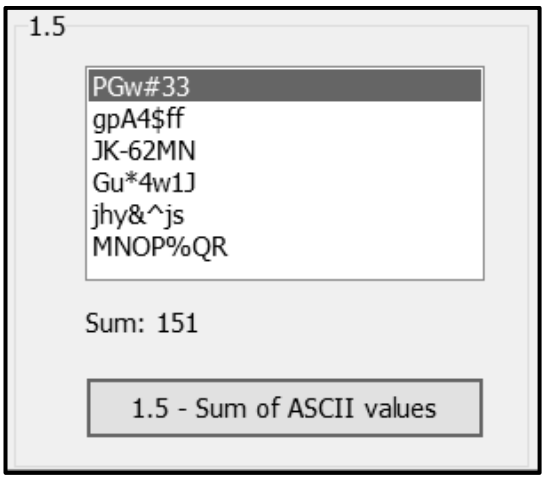
1.5 **Button [1.5 - Sum of ASCII values]**

A user must select a string from the provided list box **lstQ1_5**. The sum of the ASCII values associated with only the uppercase letters in the string must be calculated and displayed.

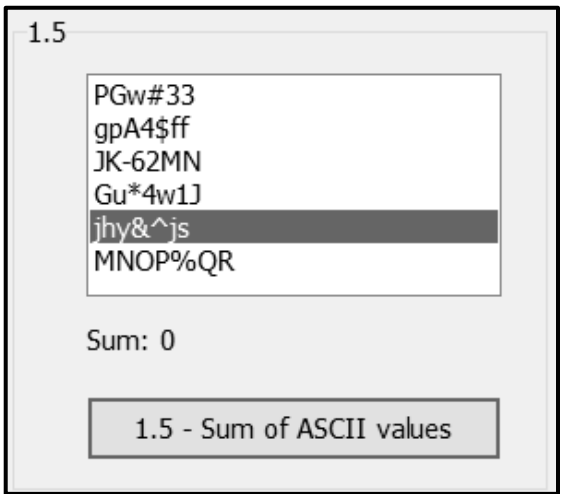
Write code to do the following:

- Declare a variable to store the sum of the values.
- Extract and store the string selected from the list box **lstQ1_5**.
- Use a loop to step through the characters in the string.
  If a character is an uppercase letter:
  - Determine the ASCII value of the uppercase letter
  - Add the value to the sum variable
- Display the sum of the ASCII values of the uppercase letters in the label **lblQ1_5**.

Example of output if the string 'PGw#33' was selected from the list box:



Example of output if the string 'jhy&^js' was selected from the list box:



(9)

- 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 gaming company requires assistance in generating queries using a comprehensive database consisting of popular video games.

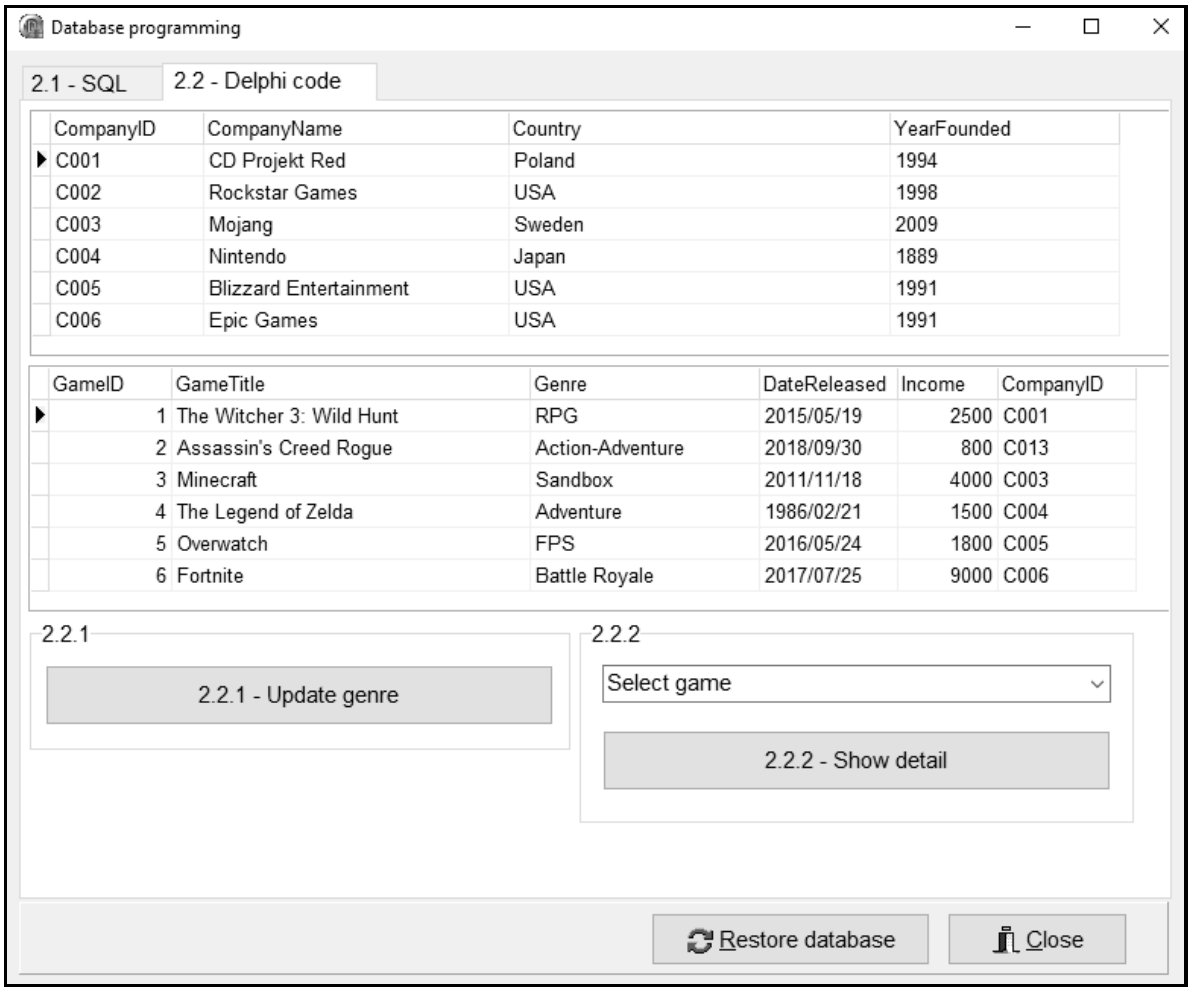
A database called **CompGamesDB.mdb**, which contains information about popular video games released before the year 2020, has been created.

The database contains two tables called **tblCompanies** and **tblGames**.

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

Do the following:

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



- 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.

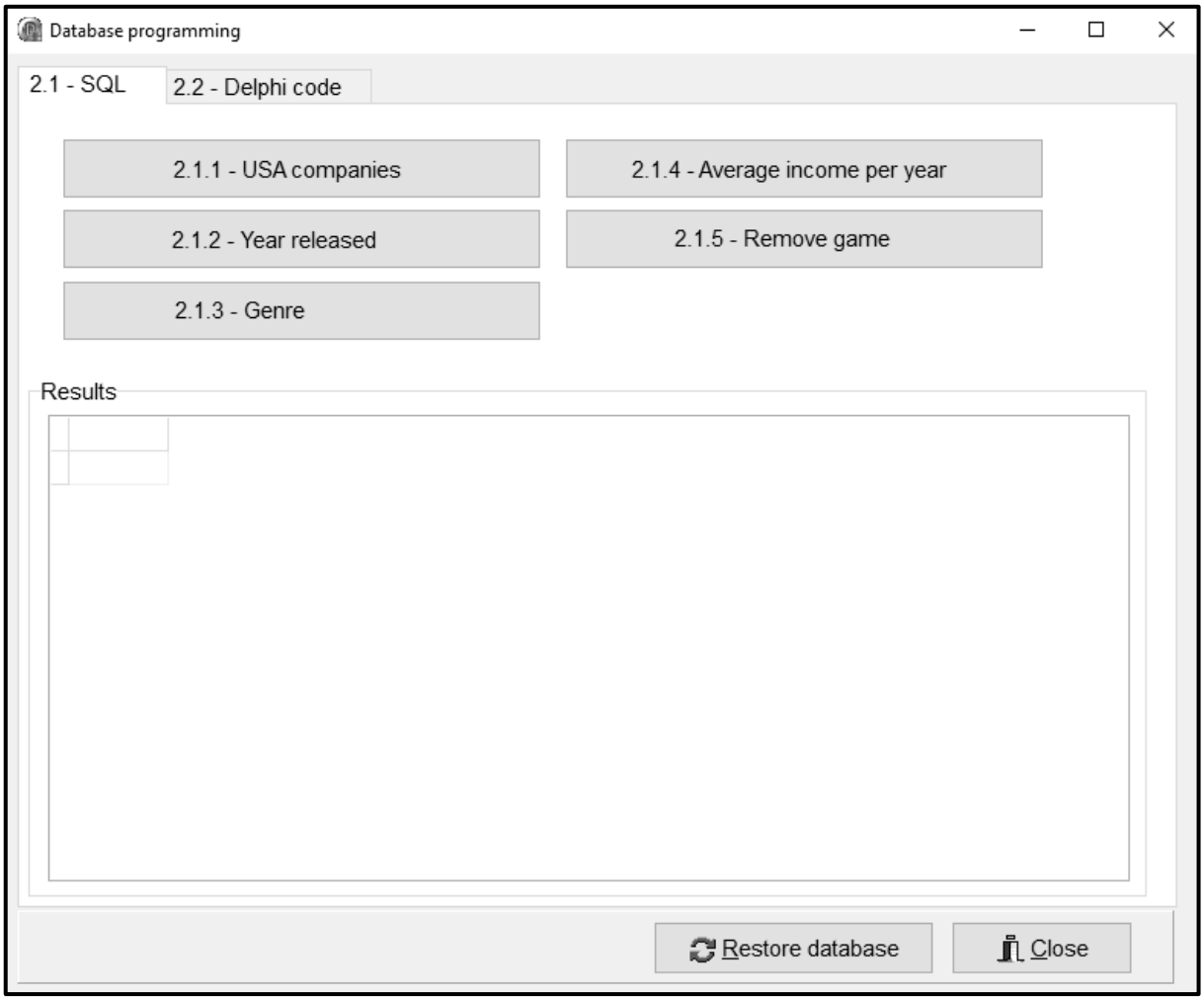
NOTE:

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

Variable	Data type	Description
`tblCompanies`	`TADOTable`	Refers to the table **tblCompanies**
`tblGames`	`TADOTable`	Refers to the table **tblGames**

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 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 - USA companies]**

Display ALL the fields of ALL the companies in the **tblCompanies** table, that were founded in the USA.

Example of output of the first four records:

CompanyID	CompanyName	Country	YearFounded
C002	Rockstar Games	USA	1998
C005	Blizzard Entertainment	USA	1991
C006	Epic Games	USA	1991
C007	InnerSloth	USA	2015

(3)

2.1.2 **Button [2.1.2 - Year released]**

Display the **GameTitle**, **Genre** and the **DateReleased** of all games from the **tblGames** table that were released since the year 2019.

Example of output of the first five records:

GameTitle	Genre	DateReleased
Call of Duty: Warzone	Battle Royale	2020/03/10
FIFA 21	Sports	2020/10/09
Assassin's Creed Valhalla	Action-Adventure	2020/11/10
Cyberpunk 2077	RPG	2020/12/10
Apex Legends	Battle Royale	2019/02/04

(4)

2.1.3 **Button [2.1.3 - Genre]**

The user must enter the genre of a game to search for, using an input dialogue box.

Code has been provided to extract the genre entered by the user and store the genre in a variable called **sGenre**.

Display the **GameTitle** and **DateReleased** of all games that match the genre entered by the user, sorted according to the date released, from the latest to the oldest game.

Example of output if the genre 'Sports' was entered:

GameTitle	DateReleased
FIFA 21	2020/10/09
Rocket League	2015/07/07

(4)

2.1.4 **Button [2.1.4 - Average Income per year]**

Determine and display the average income per year, considering the year in which the company was founded.

HINT: Divide the total income of all the games developed by the company by the number of years since the company was founded.

NOTE: The values in the **Income** field in the **tblGames** table is in millions. Example: The income from the game 'Among Us' is represented as 500, which means R500 000 000.

Display the name of the company and the average income per year in millions. The calculated average income per year must be displayed in currency format in a new field called **AverageIncomePerYear**.

Example of output of the first four companies:

CompanyName	AverageIncomePerYear
Behavior Interactive	R15 625 000.00
Bethesda Game Studios	R86 956 521.74
BioWare	R20 689 655.17
Blizzard Entertainment	R212 121 212.12

(8)

2.1.5 **Button [2.1.5 - Remove game]**

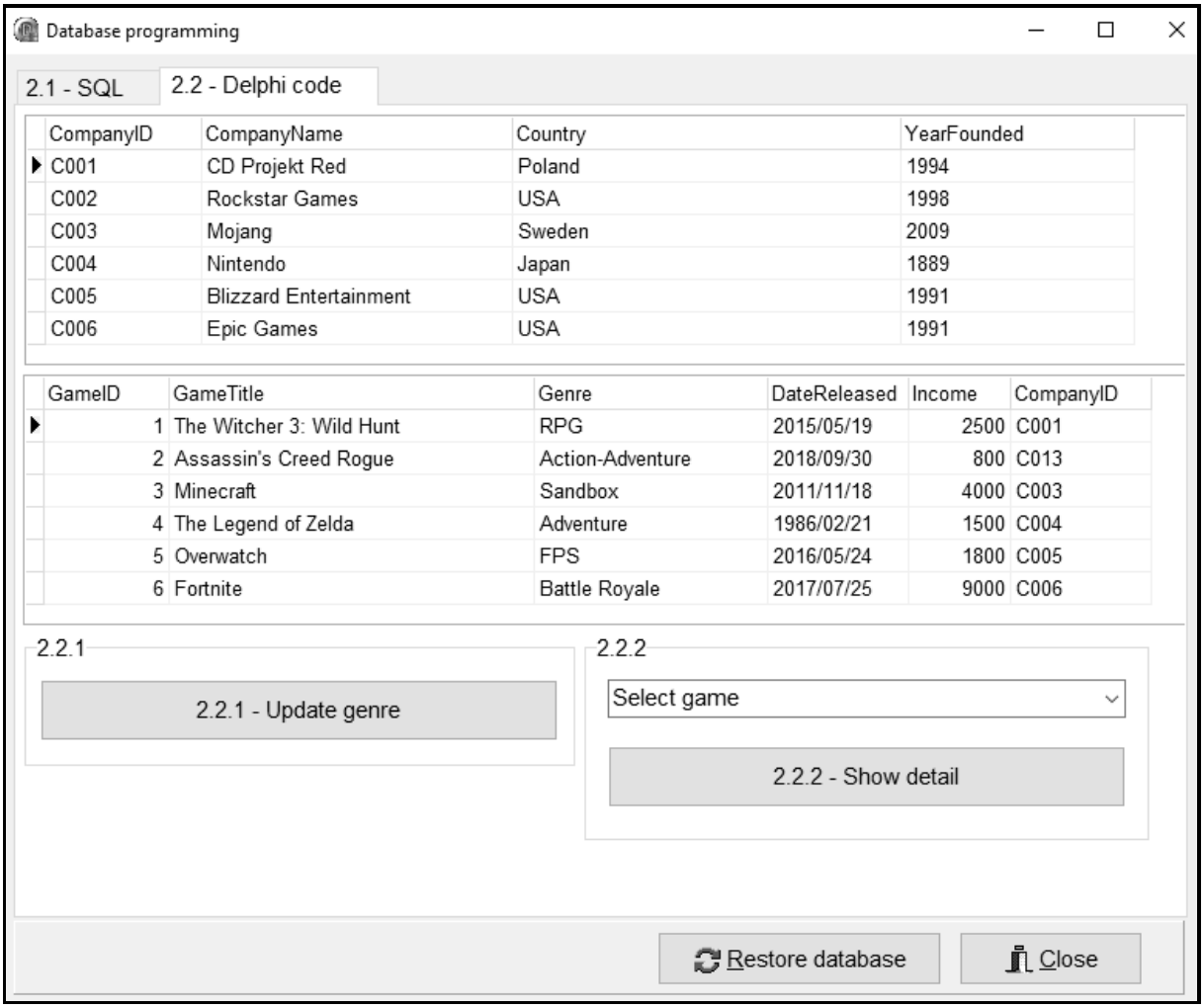
Write code to remove the game called **Apex Legends** from the **tblGames** table.

HINT: Click **Button 2.1.2** to see whether the record has been removed or not.

NOTE: Restore the database to be able to use the original data when testing your code. (2)

2.2 Tab sheet [2.2 - Delphi code]

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



NOTE:

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

2.2.1 Button [2.2.1 - Update genre]

There are various *Tetris* games available such as *Tetris 1*, *Tetris 2*, *Tetris 99*, etc. The **Genre** of all the *Tetris* games have been stored incorrectly and will need to be changed.

Write code to change the genre of all games that have 'Tetris' as part of the **GameTitle** field, to 'Puzzle'. (7)

2.2.2 **Button [2.2.2 - Show detail]**

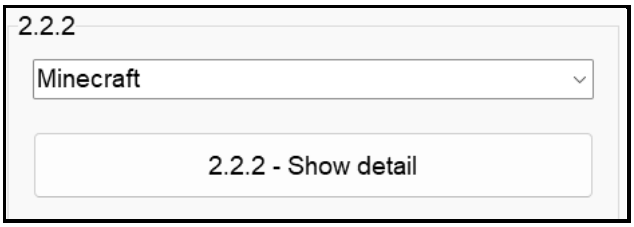
The user must select a title of a game from the combo box **cmbQ2_2_2**.

Code has been provided to extract the title of the game selected by the user and store the title in a variable called **sGame**.

Display the game title selected, the genre, the company that developed the game and the country that the company is from, in a **ShowMessage** dialogue box.

The **GameTitle** and **Genre** of the game must be displayed in the first line and the **CompanyName** and **Country** in the second line in the ShowMessage dialogue box, as shown in the example below.

Example of input:



Example of output:

(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 B: 40

SECTION C

QUESTION 3: OBJECT-ORIENTED PROGRAMMING

Gamers use an application that provide them with information about exclusive online/ offline games that are available for a single platform only.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Open the incomplete object class **Game_U.pas**.
- Enter your examination number as a comment in the first line of the **Question3_U.pas** file and the **Game_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.

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

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

The UML class diagram for the **TGame** class is given below.

TGame object	
- fName: String	// Name of game
- fOnline: Boolean	// True if internet access is required to play and false if not
- fDownloadCount: Integer	// Number of times the game was downloaded
- fPlatform: String	// Which platform the game runs on (Xbox/PlayStation/PC)
+ constructor create(sName: String, bOnline: Boolean, iDownloadCount: Integer, sPlatform: String)	
+ onlineStatus: String	
+ updateDownloadCount(iNumberDownloads: Integer)	
+ determinePopularity: String	
+ getName: String (provided)	
+ getOnline: Boolean (provided)	
+ getPlatform: String (provided)	
+ toString : String (provided - incomplete)	

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

Use the UML diagram provided to answer QUESTION 3.1.1 to QUESTION 3.1.5.

3.1.1 Write code for the **constructor create** method as defined in the provided class diagram. The attributes must be initialised using the received parameter values. (5)

3.1.2 Write code for the **onlineStatus** method that will return a String value, 'Yes' or 'No' to indicate whether the game requires internet access or not. (4)

3.1.3 Write code for the **updateDownloadCount** method, which will increase the value of the **fDownloadCount** attribute using the value received as a parameter. (3)

3.1.4 Write code for a method called **determinePopularity** that uses the information below to determine the popularity of a game, only if it is an **online** game. Return the result (popularity) as a String value, based on the information in the table below. Return the string 'Not an online game' if the game is not an online game.

The popularity of an **online** game is determined by the download count, e.g. a game with a download count of 500 000 or more is considered 'Very popular'.

Download count	Popularity
Less than 100 000	'Not popular'
Greater than or equal to 100 000 and smaller than 500 000	'Popular'
Greater than or equal to 500 000	'Very popular'

(10)

3.1.5 An incomplete **toString** method has been provided. Complete the code for the **toString** method to indicate whether internet access is required or not for the game ('Yes'/'No').

Format of the **toString** method:

```
Name: <Name of the game>
Internet required: <Yes/No>
Number of downloads: <Download count>
Platform: <Platform>
```

(2)

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 **objGame**.

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

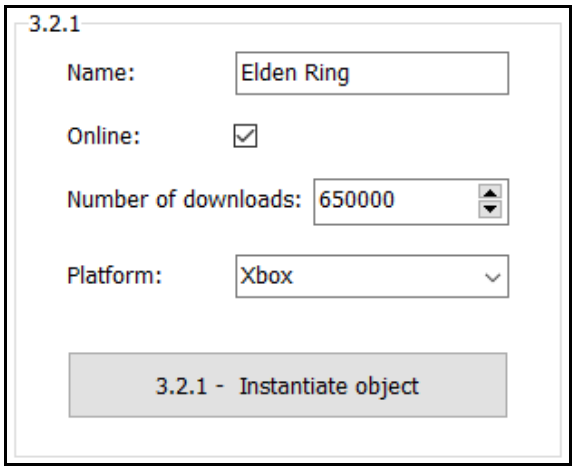
3.2.1 **Button [3.2.1 - Instantiate object]**

Code has been provided to extract and store the name, online status, number of downloads and the platform of a game from the components in variables.

Write code to do the following:

- Use the variables **sName**, **bOnline**, **iNumberOfDownloads** and **sPlatform** to instantiate a new **objGame** object.
- Call the **toString** method to display the information of the **Game** object in the rich edit **redQ3**.

Example of input:



Example of output:

```
Name: Elden Ring
Internet required: Yes
Number of downloads: 650000
Platform: Xbox
```

(5)

3.2.2 **Button [3.2.2 - Game compatibility]**

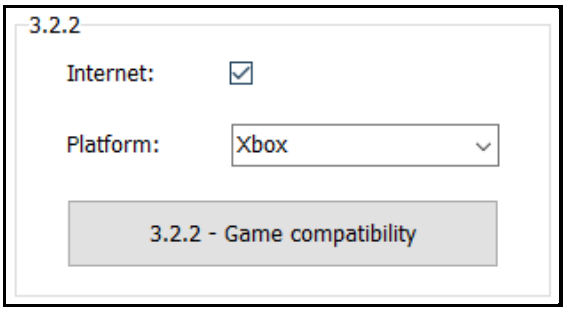
The check box **cbxQ3_2_2** must be ticked if the user has access to the internet, and the platform being used must be selected from the combo box **cmbQ3_2_2**.

Code has been provided to extract and store the values selected by the user in variables **bOnline** and **sPlatform**.

Write code to do the following:

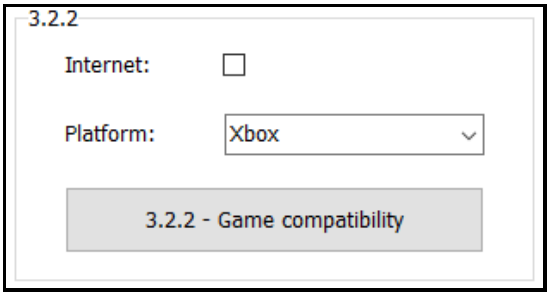
- Call the **getOnline** and **getPlatform** methods to check whether the user is able to play the game or not.
- Use a message dialogue box to display an appropriate message indicating whether the user can play the game or not.

Example of input:

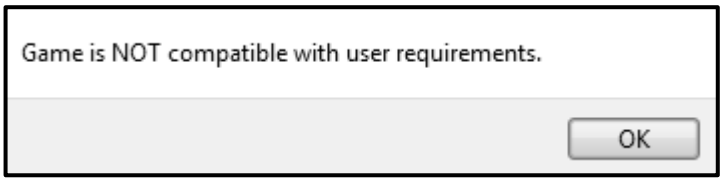


Example of output:

Example of input:



Example of output:



(4)

3.2.3 Button [3.2.3 - Update downloads]

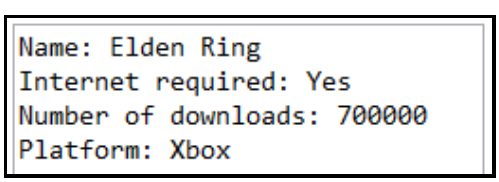
When a game is downloaded, the relevant attribute of the **Game** object must be updated accordingly.

The user must enter the number of new downloads of the game using an input dialogue box.

Write code to do the following:

- Call the **updateDownloadCount** method using the value extracted from the input dialogue box as an argument.
- Call the **toString** method to display the updated information of the **Game** object in the rich edit **redQ3**.

Example of output for the game *Elden Ring* that had 650 000 downloads, and another 50 000 downloads were added:



(5)

3.2.4 **Button [3.2.4 - Popularity]**

Write code to call the relevant methods to display the name and popularity level of the game in the rich edit **redQ3**.

Example of output for the game *Elden Ring* that has 700 000 downloads:

```
Elden Ring: Very popular
```

Example of output for the game *Orange Box* that has 50 000 downloads:

```
Orange Box: Not popular
```

(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

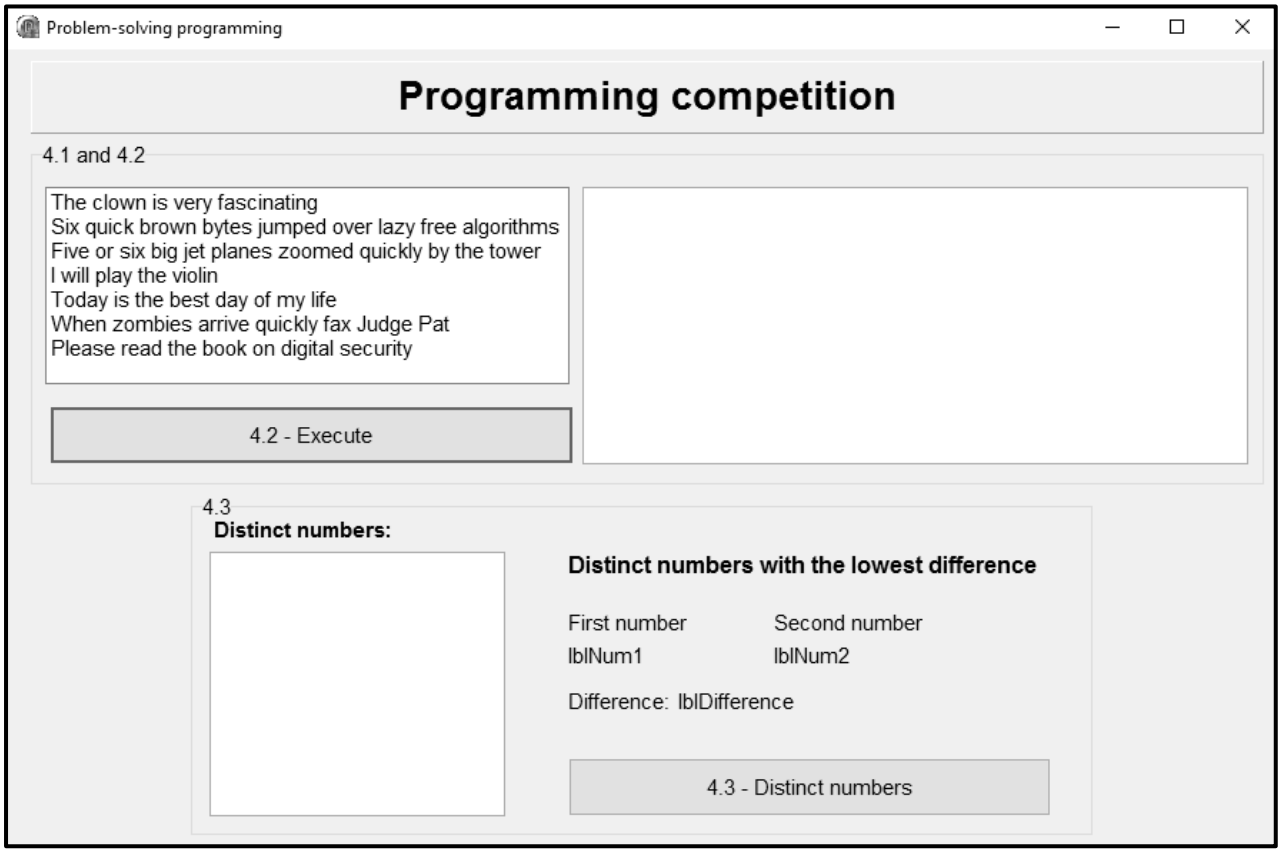
QUESTION 4: PROBLEM-SOLVING PROGRAMMING

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.

Complete the code for each section of QUESTION 4, as described in QUESTION 4.1, QUESTION 4.2 and QUESTION 4.3.

Example of graphical user interface (GUI) for QUESTION 4:



A list box, **lstQ4_2**, which contains several sentences, has been provided in the program:

Example of the sentences in list box **lstQ4_2**:

- `The clown is very fascinating`
- `Six quick brown bytes jumped over lazy free algorithms`
- `Five or six big jet planes zoomed quickly by the tower`

4.1 **Function isPangram**

A pangram is a string that contains all the letters of the alphabet at least once.

An example of a pangram: The quick brown fox jumped over the lazy dog.

Create a function called **isPangram** that will receive a sentence as a parameter and determine if the sentence is a pangram or not. Return the value TRUE if the sentence is a pangram, or otherwise, return the value FALSE. (8)

4.2 **Button [4.2 - Execute]**

Each of the sentences provided in the list box **lstQ4_2** must be read and evaluated to determine if each sentence is a pangram or not.

Write code to display the sentences and the results of the pangrams in the rich edit **redQ4_2**, as shown in the example below.

NOTE: Your code must work for any set of sentences, not only for the sentences provided in the list box.

Example of output:

Original sentence	Pangram
The clown is very fascinating	No
Six quick brown bytes jumped over lazy free algorithms	Yes
Five or six big jet planes zoomed quickly by the tower	Yes
I will play the violin	No
Today is the best day of my life	No
When zombies arrive quickly fax Judge Pat	Yes
Please read the book on digital security	No

(8)

4.3 **Button [4.3 - Distinct numbers]**

You have been provided with the following global array:

```
arrNumbers: array [1 .. 20] of Real = (10.39, 5.33, 5.48,
            9.53, 5.82, 4.21, 5.33, 2.26, 4.21, 8.48, 4.82,
            9.53, 2.17, 5.33, 9.53, 8.46, 9.53, 10.26, 5.33,
            9.21);
```

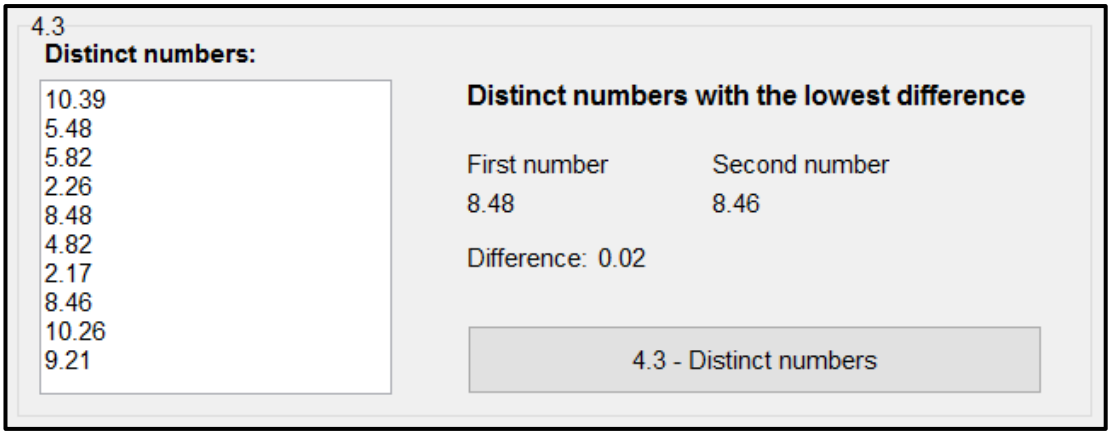
The array **arrNumbers** contains twenty real numbers.

Some of the numbers in the array appear more than once. A distinct/unique element in an array is an element that is not repeated.

Write code to do the following:

- Identify the distinct/unique numbers in the **arrNumbers** array and display these numbers in the rich edit **redQ4_3**, as shown in the example of output below.
- Determine which two of the distinct/unique numbers have the lowest difference between them, and display the two numbers in the labels **lblNum1** and **lblNum2** and the difference between them in the label **lblDifference**, formatted to TWO decimal places.

Example of output:



(14)

- 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 QUESTION 2:

The design of the database tables is as follows:

Table: **tblCompanies**

This table contains the details of each company:

Field name	Data type	Description
CompanyID	Text(10)	Unique ID of the company
CompanyName	Text(30)	Name of the company
Country	Text(15)	Country where the company is from
YearFounded	Number	Year when the company was established

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

CompanyID	CompanyName	Country	YearFounded
C001	CD Projekt Red	Poland	1994
C002	Rockstar Games	USA	1998
C003	Mojang	Sweden	2009
C004	Nintendo	Japan	1889
C005	Blizzard Entertainment	USA	1991
C006	Epic Games	USA	1991
C007	InnerSloth	USA	2015
C008	Infinity Ward	USA	2002
C009	Riot Games	USA	2006
C010	Valve Corporation	USA	1996

Table: **tblGames**

This table contains the details of the games developed by each company:

Field name	Data type	Description
GameID	Number	Unique ID for the game
GameTitle	Text(40)	Title of the game
Genre	Text(20)	Genre of the game
DateReleased	DateTime	Date the game was released
Income	Number	Total income since the release date in millions
CompanyID	Text(10)	The ID of the company that developed the game

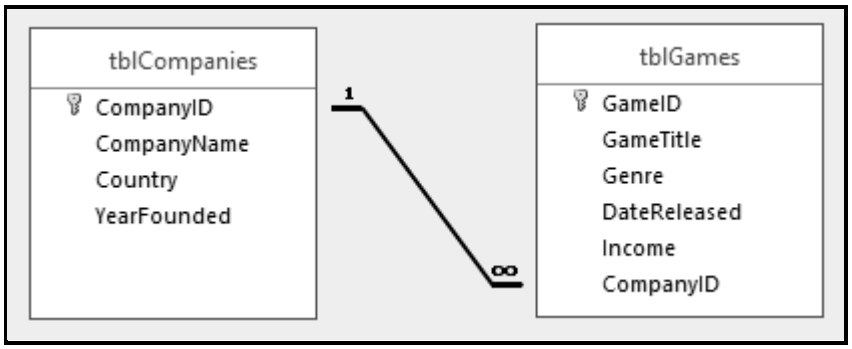
Example of the first ten records of the **tblGames** table:

GameID	GameTitle	Genre	DateReleased	Income	CompanyID
1	The Witcher 3: Wild Hunt	RPG	2015/05/19	2500	C001
2	Assassin's Creed Rogue	Action-Adventure	2018/09/30	800	C013
3	Minecraft	Sandbox	2011/11/18	4000	C003
4	The Legend of Zelda	Adventure	1986/02/21	1500	C004
5	Overwatch	FPS	2016/05/24	1800	C005
6	Fortnite	Battle Royale	2017/07/25	9000	C006
7	Red Dead Redemption 2	Action-Adventure	2018/10/26	3500	C002
8	Among Us	Social Deduction	2018/06/15	500	C007
9	Call of Duty: Warzone	Battle Royale	2020/03/10	2000	C008
10	League of Legends	MOBA	2009/10/27	3000	C009

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 – MAY/JUNE 2025
INFORMATION SHEET *(to be completed by the candidate AFTER the 3-hour exam session)*

CENTRE NUMBER: _____

EXAMINATION NUMBER: _____

WORK STATION NUMBER: _____

Version of Delphi used during the INFORMATION TECHNOLOGY SC/NSC May/June 2025 Examinations:

Mark appropriate box with a cross (**X**)	Delphi 10	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	Filename	Saved (✓)	Attempted (✓)	Maximum Mark	Mark Achieved	Marker Code
1	Question1_P.dproj			**40**		
2	Question2_P.dproj			**40**		
3	MGame_U.pas			**24**		
	Question3_P.dproj			**16**		
4	Question4_P.dproj			**30**		
TOTAL				**150**		

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

_____

_____

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/
NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

MAY/JUNE 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 11) 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 shape]** Check if ✓ itemIndex = 0 / item is circle ✓ Change shape property of shpQ1_1 ✓ to stCircle Else ✓ Change shape property of shpQ1_1 to stRectangle ✓ **Also accept:** • Case can be used instead of if statement **Concepts**: • Test using if / case (1) • Using itemIndex / item (1) • Changing the shape property (1) • Else (1) • Correctly assigning both shapes (1)	**5**	
1.2	**Button [1.2 - Display colour]** Correct use of the colours selected from the combo boxes ✓ Check if the colours selected in either of the combo boxes is red and blue: if ((sColor1 = 'Red') ✓ AND (sColor2 = 'Blue')) ✓ OR ✓ ((sColor1 = 'Blue') AND (sColor2 = 'Red')) ✓ then Set the colour of pnlQ1_2 to clPurple ✓ Else ✓ Set the caption of pnlQ1_2 to 'No colour' ✓ **Concepts**: • Extract colours selected (1) • Use of if-statement / case statement to test for the first colour (1) • Test for second colour in combination (1) • Correct use of AND (1), correct use of OR in both combinations (1) • Set colour to clPurple (1) • Else (1) • Set caption to No colour (1)	**8**	

1.3	**Button [1.3 - Average]** Initialise variables (sum and counter) Test if the text file does not exist (Try/FileExists) ✓ Display error message and exit program ✓ AssignFile (tFile, 'DataQ1_3.txt') ✓ Reset file ✓ Loop through the text file ✓ Read value from file into variable ✓ Add value to sum ✓ Increase counter ✓ Calculate the average ✓ and display formatted to two decimal places in memQ1_3 ✓	**10**	

1.4	**Button [1.4 - Display pattern]**
	
	Extract the number of rows selected from spnQ1_4 ✓
	Outer loop from 1 to the number of rows ✓
	Initialise string ✓
	Nested ✓ inner loop from 1 to outer loop ✓
	Add outer loop counter value ✓ to string ✓
	// End of inner loop
	Display output string in redQ1_4 ✓
	// End of outer loop
		8	

1.5	**Button [1.5 - Sum of ASCII values]** Extract word from list box ✓ Initialise sum ✓ Loop ✓ from 1 to length of word ✓ Test if character ✓ is uppercase alphabet character ✓ Add the ASCII value of the character ✓ to sum ✓ Display sum ✓	**9**	
	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 - USA companies]**
	`SELECT * ✓`
	`FROM tblCompanies ✓`
	`WHERE Country = "USA" ✓`
		3	

2.1.2	**Button [2.1.2 - Year released]**
	`SELECT GameTitle, Genre, DateReleased ✓`
	`FROM tblGames ✓`
	`WHERE YEAR ✓ (DateReleased) >= 2019 ✓`
	
	Also accept:
	`Left (DateReleased,4) >= 2019`
	`Mid (DateReleased,1,4) >= 2019`
	`Year (DateReleased) > 2018`
	`DateReleased >= #2019/01/01#`
		4	

2.1.3	**Button [2.1.3 - Genre]**
	`SELECT GameTitle, DateReleased`
	`FROM tblGames ✓`
	`WHERE Genre = ✓"' + sGenre + '" ✓`
	`ORDER BY DateReleased DESC ✓`
	
	Also accept:
	`WHERE Genre = ' + QuotedStr(sGenre) + '`
		4	

2.1.4	**Button [2.1.4 – Average income per year]**
	`SELECT CompanyName,`
	`Format((Sum(Income)*1000000) ✓/`
	`(Year(Date())✓ - YearFounded - 1) ✓,"Currency"✓)`
	`AS AverageIncomePerYear ✓`
	`FROM tblCompanies, tblGames ✓`
	`WHERE tblCompanies.CompanyID =`
	`tblGames.CompanyID ✓`
	`GROUP BY CompanyName, YearFounded ✓`
	
	Also accept:
	`Year(Now)`
		8	

2.1.5	**Button [2.1.5 - Remove game]**
	`DELETE FROM tblGames ✓`
	`WHERE GameTitle = "Apex Legends" ✓`
		2	

	Subtotal:	**21**	

.QUESTION 2: MARKING GRID (CONT.)

2.2	**Database Manipulation**		
2.2.1	**Button [2.2.1 - Update genre]** Go to the first record in tblGames ✓ Loop through tblGames ✓ Test if Tetris ✓ is part of the field tblGames['GameTitle'] ✓ tblGames.Edit; ✓ tblGames['Genre'] := 'Puzzle' ✓ tblGames.Post; tblGames.Next; ✓	**7**	
2.2.2	**Button [2.2.2 - Show detail]** Go to the first record in tblGames ✓ Loop through tblGames ✓ Test if tblGames['GameTitle'] = sGame ✓ Go to the first record in tblCompanies ✓ Loop through tblCompanies ✓ Test if (tblGames ['CompanyID'] = ✓ tblCompanies ['CompanyID']) ✓ Use a ShowMessage dialog box ✓ to display: tblGames['GameTitle'] + ' ' + tblGames['Genre'] ✓ + #13 + 'Developed by ' + tblCompanies ['CompanyName'] + ', ' + tblCompanies ['Country'] ✓ tblCompanies.Next ✓ End loop (tblCompanies) tblGames.Next ✓ End loop (tblGames)	**12**	
	Subtotal:	**19**	
	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 method:** Heading with four parameter values (String, Boolean, Integer, String) ✓ Set `fName` to String parameter ✓ Set `fOnline` to Boolean parameter ✓ Set `fDownloadCount` to integer parameter ✓ Set `fPlatform` to String parameter ✓	**5**	

3.1.2	**onlineStatus function:**
	
	function heading with String return type ✓
	
	Check if fOnline ✓ = True
	  Result = 'Yes' ✓
	Else
	  Result = 'No' ✓
		4	

3.1.3	**updateDownloadCount procedure:** procedure heading with integer parameter ✓ fDownloadCount = fDownloadCount ✓ + parameter ✓	**3**	

3.1.4	**determinePopularity function:**
	
	function heading with String return type ✓
	
	Check if fOnline ✓ = True
	  if fDownloadCount < 100000 ✓
	  Result := 'Not popular' ✓
	  Else ✓
	  if fDownloadCount < 500000 ✓
	  Result := 'Popular' ✓
	  Else ✓
	  Result := 'Very popular' ✓
	Else
	  Result := 'Not an online game'; ✓
		10	

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.5	**toString function** Result := 'Name: ' + fName + #13 + 'Internet required: ' + **onlineStatus** + #13 + 'Number of downloads: ' + IntToStr(fDownloadCount) + #13 + 'Platform: ' + fPlatform; Calling onlineStatus method ✓ at the correct position. ✓	**2**	
	Subtotal: Object class	**24**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 - Instantiate object]** objGame := ✓ TGame.create ✓ (sName, bOnline, iNumberOfDownloads, sPlatform) ✓ Display objGame information in redQ3 ✓ using toString method. ✓	**5**	
3.2.2	**Button [3.2.2 - Game compatibility]** Check if (bOnline = objGame.getOnline) ✓ AND (sPlatform = objGame.getPlatform) ✓ ShowMessage('Game is compatible with user requirements.') ✓ Else ShowMessage('Game is NOT compatible with user requirements.'); ✓	**4**	
3.2.3	**Button [3.2.3 – Update downloads]** Extract number entered using a dialog box ✓ and typecast to integer ✓ Call the updateDownloadCount method ✓ with the extracted number as argument ✓ Display objGame information in redQ3 using toString method. ✓	**5**	
3.2.4	**Button [3.2.4 - Popularity]** Call the getName and the determinePopularity methods ✓ Display the name of the game as well as the popularity in redQ3 ✓ in the correct format	**2**	
	Subtotal Form class:	**16**	
	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	**Function isPangram** Function heading with string parameter ✓ with Boolean return type ✓ Loop ✓ or mechanism to check each letter of the alphabet OR build frequency/count structure Mechanism to check presence of all 26 letters: (3) • Alphabet letters are individually tracked or counted ✓ • A correct uppercase / lowercase check is applied to each letter ✓ • To ensure that the letter appears in the string ✓ Checking counter / Boolean ✓ Return TRUE if all alphabet letters are found Return FALSE if any alphabet letter is missing ✓	**8**	
4.2	**Button [4.2 - Execute]** Loop ✓ through sentences ✓ Extract the sentence from lstQ4_2 ✓ If statement – call isPangram ✓ with argument ✓ Display sentence with 'Yes' ✓ separated by #9 ✓ in redQ4_2 Else Display sentence with 'No' separated by #9 in redQ4_2✓	**8**	

4.3	**Button [4.3 - Distinct numbers]**
	
	Concepts:
	
	Create String / temporary array with unique numbers:
	Outer loop ✓ Inner loop ✓
	Determine if number is unique ✓✓✓
	Add unique number to string / temporary array ✓✓
	Display unique number ✓
	
	Determine lowest difference:
	Nested looping ✓
	Calculate difference as a positive number ✓
	Determine if current difference is the lowest ✓
	Store two numbers and the lowest difference ✓✓
	
	Display numbers with lowest difference in labels and the lowest difference correctly formatted ✓
	
	Possible solution:
	
	`Initialise index of unique array to 0`
	`Loop outer from 1 to length or arrNumbers`
	`  Initialise counter to 0`
	`   Nested loop inner from 1 to length or arrNumbers`
	`      Check if arrNumbers[outer] = arrNumbers[inner]`
	`            Increment counter`
	`   //End inner loop`
	
	`//Check if duplicate elements not found`
	`    Test if counter = 1`
	`        Increment index of unique array`
	`        arrUnique[index] = arrNumbers[outer]`
	`         Display unique number/arrUnique[index]`
	`    //End outer loop`
	
	`Initialise lowest difference variable`
	`Loop I from 1 to length of temp array - 1`
	`    Nested loop J from I + 1 to length of temp array`
	`        Current difference =`
	`                    Abs(arrTemp[I] - arrTemp[J])`
	`        Check if (rCurrDiff < rLowestDiff) AND`
	`     (rCurrDiff <> 0)`
	`            rNum1 := arrTemp[I];`
	`            rNum2 := arrTemp[J];`
	`            rLowestDiff:= rCurrDiff;`
	
	`Display numbers with lowest difference and the lowest difference on labels correctly formatted`
		14	

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

SUMMARY OF LEARNER'S MARKS:

CENTER 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, ComCtrls, Spin, ExtCtrls, Math;

type
  TfrmQuestion1 = class(TForm)
    grpQ1_1: TGroupBox;
    grpQ1_2: TGroupBox;
    grpQ1_3: TGroupBox;
    grpQ1_4: TGroupBox;
    grpQ1_5: TGroupBox;
    rgpQ1_1: TRadioGroup;
    shpQ1_1: TShape;
    btnQ1_1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    cmbColour1: TComboBox;
    cmbColour2: TComboBox;
    btnQ1_2: TButton;
    pnlQ1_2: TPanel;
    Label3: TLabel;
    spnQ1_3: TSpinEdit;
    btnQ1_3: TButton;
    redQ1_3: TRichEdit;
    btnQ1_4: TButton;
    pnlQ1_4: TPanel;
    btnQ1_5: TButton;
    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                                                        5 marks
// =============================================================================

procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  // Question 1.1

  if rgpQ1_1.ItemIndex = 0 then
    shpQ1_1.Shape := stCircle
  else
    shpQ1_1.Shape := stRectangle;
end;

//=============================================================================
// Question 1.2                                                        8 marks
// =============================================================================

procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  sColour1, sColour2: String;
begin
  // Provided code
  pnlQ1_2.Color := clBtnFace;
  pnlQ1_2.Caption := '';

  // Question 1.2
  sColour1 := cmbColour1.Text;
  sColour2 := cmbColour2.Text;

  if ((sColour1 = 'Red') or (sColour2 = 'Red')) AND
    ((sColour1 = 'Blue') or (sColour2 = 'Blue')) then
    pnlQ1_2.Color := clPurple
  else
    pnlQ1_2.Caption := 'No colour';

end;
```

```
//=============================================================================
// Question 1.3                                                        10 marks
// ============================================================================

procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  tFile: textFile;
  iNum, iSum, iCnt: Integer;
  rAverage: real;
begin
  AssignFile(tFile, 'DataQ1_3.txt');
  try
    reset(tFile);
  except
    ShowMessage('File not found');
    Application.Terminate;
  end;
  iSum := 0;
  iCnt := 0;
  while not eof(tFile) do
  begin
    readln(tFile, iNum);
    iSum := iSum + iNum;
    Inc(iCnt);
  end;
  rAverage := iSum / iCnt;
  memQ1_3.Lines.Add('Average: ' + FloatToStrF(rAverage, ffFixed, 8, 2));
  CloseFile(tFile);
end;

//=============================================================================
// Question 1.4                                                         8 marks
// ============================================================================

procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  iRows, I, J, K: Integer;
  sConcat: String;
begin
  // Provided code
  redQ1_4.Clear;

  // Question 1.4
  iRows := spnQ1_4.Value;

  for I := 1 to iRows do
  begin
    sConcat := '';
    for J := 1 to I do
      sConcat := sConcat + IntToStr(I);
    redQ1_4.Lines.Add(sConcat);
  end;

end;
```

```
//=============================================================================
// Question 1.5                                                      9 marks
// =============================================================================

procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  sWord: String;
  iCnt, iTot: Integer;

begin
  iTot := 0;
  sWord := lstQ1_4.Items[lstQ1_4.ItemIndex];
  for iCnt := 1 to length(sWord) do
  begin
    if sWord[iCnt] IN ['A' .. 'Z'] then
    begin
      iTot := iTot + Ord(sWord[iCnt])
    end;
  end;
  lblQ1_4.Caption := IntToStr(iTot);
end;

procedure TfrmQuestion1.FormCreate(Sender: TObject);
begin
  shpQ1_1.Visible := False;
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

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

//============================================================================
//  Question 2.1.1                                                    3 marks
//============================================================================
      sSQL1 := 'SELECT * ' +
               'FROM tblCompanies ' +
               'WHERE Country = "USA"';

//============================================================================
//  Question 2.1.2                                                    4 marks
//============================================================================
      sSQL2 := 'SELECT GameTitle, Genre, DateReleased ' +
               'FROM tblGames ' +
               'WHERE YEAR (DateReleased) >= 2019';

//============================================================================
//  Question 2.1.3                                                    4 marks
//============================================================================
      sSQL3 := 'SELECT GameTitle, DateReleased ' +
               'FROM tblGames ' +
               'WHERE Genre = "' + sGenre + '" ' +
               'ORDER BY DateReleased DESC';

//============================================================================
//  Question 2.1.4                                                    8 marks
//============================================================================
      sSQL4 := 'SELECT CompanyName, format((sum(Income)*1000000)/
                (year(date()) - YearFounded - 1),"Currency") ' +
                'AS IncomePerYear ' +
                'FROM tblCompanies, tblGames ' +
                'WHERE tblCompanies. CompanyID = tblGames. CompanyID ' +
                'GROUP BY CompanyName, YearFounded';

//============================================================================
//  Question 2.1.5                                                    2 marks
//============================================================================
      sSQL5 := 'DELETE FROM tblGames ' +
               'WHERE GameTitle = "Apex Legends"';
```

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

//=============================================================================
//  Question 2.2.1                                                    7 marks
//=============================================================================
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
begin
  tblGames.First;
  while (NOT tblGames.Eof) do
  begin
    if pos('Tetris', tblGames['GameTitle']) > 0 then
    begin
      tblGames.Edit;
      tblGames['Genre'] := 'Puzzle';
      tblGames.Post;
    end;
    tblGames.Next;
  end;
end;

//=============================================================================
//  Question 2.2.2                                                   12 marks
//=============================================================================
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  sGame,sDetail : String;
begin
  // Provided code
  sGame := cmbQ2_2_2.Text;

  // 2.2.2 - Show detail
  tblGames.First;
   while (NOT tblGames.Eof) do
     begin
       if (sGame = tblGames['GameTitle']) then
         begin
           tblCompanies.First;
           while (NOT tblCompanies.EOF) do
             begin
               if (tblGames['CompanyID'] =
                   tblCompanies['CompanyID']) then
                 begin
                   ShowMessage(tblGames['GameTitle'] + ' ' +
                               tblGames['Genre'] + #13 +
                               'Developed by ' +
                              tblCompanies['CompanyName'] + ' ' +
                              tblCompanies['Country']);
                 end;
               tblCompanies.Next;
             end;
         end;
       tblGames.Next;
     end;
end;
```

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

procedure TfrmQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  // Restores the Database
  dbCONN.RestoreDatabase;
  dbCONN.SetupGrids(dbgDevelopers, dbgGames, dbgrdSQL);
end;

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

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

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
unit Game_U;

interface

Uses SysUtils;

Type
  TGame = class(TObject)
  private
    fName: String;
    fOnline: Boolean;
    fDownloadCount: Integer;
    fPlatform: String;
  public
    constructor create(sName: String; bOnline: Boolean;
                       iDownloadCount: Integer; sPlatform: String);
    function onlineStatus: String;
    procedure updateDownloadCount(iNumberDownloads: Integer);
    function determinePopularity: String;
    // Provided code
    function getName: String;
    function getPlatform : String;
    function getOnline: Boolean;
    function toString : String;
  end;

implementation

{ TGame }

// ============================================================================
// Question 3.1.1                                                    5 marks
// ============================================================================
constructor TGame.create(sName: String; bOnline: Boolean;
                         iDownloadCount: Integer; sPlatform: String);
begin
  // Question 3.1.1
  fName := sName;
  fOnline := bOnline;
  fDownloadCount := iDownloadCount;
  fPlatform := sPlatform;
end;
```

```
// ============================================================================
// Question 3.1.2                                                    4 marks
// ============================================================================
function TGame.onlineStatus: String;
begin
  // Question 3.1.2
  if fOnline then
    Result := 'Yes'
  else Result := 'No';
end;

// ============================================================================
// Question 3.1.3                                                    3 marks
// ============================================================================
procedure TGame.updateDownloadCount(iNumberDownloads: Integer);
begin
  // Question 3.1.3
  fDownloadCount := fDownloadCount + iNumberDownloads;
end;

// ============================================================================
// Question 3.1.4                                                   10 marks
// ============================================================================
function TGame.determinePopularity: String;
begin
  // Question 3.1.4
  if fOnline then
  begin
    If fDownloadCount < 100000 then
      Result := 'Not popular'
    else if fDownloadCount < 500000 then
      Result := 'Popular'
    else
      Result := 'Very popular';
  end
  else
    Result := 'Not an online game';
end;

// ============================================================================
// Question 3.1.5                                                    2 marks
// ============================================================================
function TGame.toString: String;
begin
  // Question 3.1.5
  Result := 'Name: ' + fName + #13 + 'Internet required: ' + onlineStatus +
            #13 + 'Number of downloads: ' + IntToStr(fDownloadCount) +
            #13 + 'Platform: ' + fPlatform;
end;
```

```
// ============================================================================
// Provided code - do not change
// ============================================================================
function TGame.getName: String;
begin
  Result := fName;
end;

function TGame.getOnline: Boolean;
begin
  Result := fOnline;
end;

function TGame.getPlatform: String;
begin
  Result := fPlatform;
end;{$ENDREGION}

// ============================================================================
end.
```

Main Form Unit

```
// ============================================================================
// Question 3.2.1                                                    5 marks
// ============================================================================
procedure TTfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
  sName, sPlatform : String;
  bOnline: Boolean;
  iNumberOfDownloads: integer;
begin
  // Provided code
  redQ3.Lines.Clear;

  sName := edtQ3_2_1.Text;
  bOnline := cbxQ3_2_1.Checked;
  iNumberOfDownloads := spnQ3_2_1.Value;
  sPlatform := cmbQ3_2_1.Text;

  // End of provided code

  // 3.2.1 Instantiate object
  objGame := TGame.create(sName, bOnline,
                          iNumberOfDownloads, sPlatform);
  redQ3.Lines.Add(objGame.toString);
end;

// ============================================================================
// Question 3.2.2                                                    4 marks
// ============================================================================
procedure TTfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  bOnline : Boolean;
  sPlatform : String;
begin
  // Provided code
  redQ3.Lines.Clear;

  bOnline := cbxQ3_2_2.Checked;
  sPlatform := cmbQ3_2_2.Text;

  // 3.2.2 - Game compatibility
  if (bOnline = objGame.getOnline) AND (sPlatform = objGame.getPlatform)
then
    ShowMessage('Game is compatible with user requirements.')
  else
    ShowMessage('Game is NOT compatible with user requirements.');
end;
```

```
// ============================================================================
// Question 3.2.3                                                    5 marks
// ============================================================================
procedure TTfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
var
  iNumberOfDownloads: integer;
begin
  // Provided code
  redQ3.Lines.Clear;

  // 3.2.3  - Update number of downloads
  iNumberOfDownloads := StrToInt(InputBox('Number of downloads', 'Please
enter number of downloads', ''));
  objGame.updateDownloadCount(iNumberOfDownloads);
  redQ3.Lines.Add(objGame.toString);
end;

// ============================================================================
// Question 3.2.4                                                    2 marks
// ============================================================================
procedure TTfrmQuestion3.btnQ3_2_4Click(Sender: TObject);
begin
  // Provided code
  redQ3.Lines.Clear;

  // 3.2.4 -  Determine popularity of a game
  redQ3.Lines.Add(objGame.getName + ': ' + objGame.determinePopularity);
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;

interface

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

type
  TfrmQuestion4 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    btnQ4_2: TButton;
    redQ4_2: TRichEdit;
    lstQ4_2: TListBox;
    grpQ4_3: TGroupBox;
    redQ4_3: TRichEdit;
    btnQ4_3: TButton;
    lblNum1 : Tlabel;
    lblNum2 : Tlabel;
    lblDifference : TLabel;
    procedure btnQ4_2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnQ4_3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    function isPangram(sSentence: String): boolean;
  end;

var
  frmQuestion4: TfrmQuestion4;
  arrNumbers: array [1..20] of real;

implementation

{$R *.dfm}

// ============================================================================
// Question 4.1                                                      8 marks
// ============================================================================

function TfrmQuestion4.isPangram(sSentence: String): boolean;
var
  iCount: integer;
  cChar: char;
begin
  iCount := 0;
  for cChar := 'A' to 'Z' do
  begin
    if pos(cChar, UpperCase(sSentence)) > 0 then
      Inc(iCount);
  end;
  Result :=  iCount = 26;
end;
```

```
// ============================================================================
// Question 4.2                                                      8 marks
// ============================================================================

procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  sSentence, sTemp, sWord, sOut: String;
  I : integer;
  bFlag : boolean;
begin
  // Provided code
  redQ4_2.Lines.Add('Original sentence' + #9 + 'Pangram' + #13);

  // Question 4.2
  I := 0;
  lstQ4_2.Items.Add('');

  repeat
      sSentence := lstQ4_2.Items[I];
      inc(I);
      if sSentence <> '' then
      begin
        if isPangram(sSentence) then
          sOut := sSentence + #9 + 'Yes'
        else if (isPangram(sSentence) = False) then
          sOut := sSentence + #9 + 'No';
        redQ4_2.Lines.Add(sOut);
      end;
  until (sSentence = '');
end;

// ============================================================================
// Question 4.3                                                     14 marks
// ============================================================================

procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  iLoop1, iLoop2: integer;
  rNum, rLowest, rDiff, rNum1, rNum2: real;
  iCount, iUnique: integer;
  arrUnique: array [1..20] of real;
begin
  iCount := 0;
  for iLoop1 := 1 to 20 do
    begin
      iUnique := 0;
      for iLoop2 := 1 to 20 do
        if arrNumbers[iLoop1] = arrNumbers[iLoop2] then
          inc(iUnique);

      if iUnique = 1 then
        begin
          inc(iCount);
          arrUnique [iCount] := arrNumbers[iLoop1];
          redQ4_3.Lines.Add(FloatToStr(arrUnique [iCount]));
        end;
    end;
```

```
  rLowest := ABS(arrUnique[1] - arrUnique[2]);

  for iLoop1 := 1 to iCount - 1 do
  begin
    for iLoop2 := iLoop1+1 to iCount do
    begin
      rDiff := ABS(arrUnique[iLoop1] - arrUnique[iLoop2]);
      if rDiff < rLowest then
      begin
        rLowest := rDiff;
        rNum1 := arrUnique[iLoop1];
        rNum2 := arrUnique[iLoop2];
      end;
    end;
  end;
  lblNum1.Caption := FloatToStr(rNum1);
  lblNum2.Caption := FloatToStr(rNum2);
  lblDifference.Caption := FloatToStrF(rLowest, ffFixed,8,2);
end;

// ============================================================================
// Provided code
// ============================================================================
procedure TfrmQuestion4.FormCreate(Sender: TObject);
begin
  arrNumbers[1] := 10.39;
  arrNumbers[2] := 5.33;
  arrNumbers[3] := 5.48;
  arrNumbers[4] := 9.53;
  arrNumbers[5] := 5.82;
  arrNumbers[6] := 4.21;
  arrNumbers[7] := 5.33;
  arrNumbers[8] := 2.26;
  arrNumbers[9] := 4.21;
  arrNumbers[10] := 8.48;
  arrNumbers[11] := 4.82;
  arrNumbers[12] := 9.53;
  arrNumbers[13] := 2.17;
  arrNumbers[14] := 5.33;
  arrNumbers[15] := 9.53;
  arrNumbers[16] := 8.46;
  arrNumbers[17] := 9.53;
  arrNumbers[18] := 10.26;
  arrNumbers[19] := 5.33;
  arrNumbers[20] := 9.21;
  redQ4_2.Paragraph.TabCount := 2;
  redQ4_2.Paragraph.Tab[0] := 290;
  redQ4_2.Paragraph.Tab[1] := 390;
end;

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

end.
```