A blue shield with yellow and white text

Description automatically generated

**Protea Heights Academy**

**GRADE 12**

**INFORMATION TECHNOLOGY P1**

**PRELIMINARY EXAMINATION - PRACTICAL**

**AUGUST 2024**

**MARKS: 150**

**TIME: 3 HOURS**

**This question paper consists of 28 pages including 2 data pages and 2 pages for planning purposes.**

|  |  |  |
| --- | --- | --- |
| **INSTRUCTIONS AND INFORMATION** |  |  |

|  |  |
| --- | --- |
| 1. | This paper is divided into FOUR sections. Candidates must answer ALL the questions from all FOUR sections. |
|  |  |
| 2. | The duration of this examination is three hours. Because of the nature of this examination, it is important to note that you will not be permitted to leave the examination room before the end of the examination session. |
|  |  |
| 3. | This question paper is set with programming terms that are specific to Delphi programming language. The Delphi programming language must be used to answer the questions. |
|  |  |
| 4. | Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements. |
|  |  |
| 5. | Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation. |
|  |  |
| 6. | Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper. |
|  |  |
| 7. | Routines, such as search, sort and selection, must be developed from first principles. You may NOT use the built-in features of the Delphi programming language for any of these routines. |
|  |  |
| 8. | All data structures must be defined by you, the programmer, unless the data structures are supplied. |
|  |  |
| 9. | You must save your work regularly on the disk/CD/DVD/flash disk you have been given, or on the disk space allocated to you for this examination session. |
|  |  |
| 10. | Make sure that your name and surname appears as a comment in every program that you code, as well as on every event indicated. |
|  |  |
| 11. | If required, print the programming code of all the programs/classes that you completed. Your name and surname must appear on all the printouts. You will be given half an hour printing time after the examination session. |
|  |  |
| 12. | At the end of this examination session, you must hand in a disk/CD/DVD/ flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read. |

|  |  |
| --- | --- |
| 13. | The files that you need to complete this question paper have been provided to you on the disk/CD/DVD/flash disk or on the disk space allocated to you. The files are provided in the form of password-protected executable files.  Do the following:   * Double click on the following password-protected executable file:   **DataENGSept2024.exe**.   * Click on the extract button. * Enter the following password: **Pass@W#2024Prelim**   Once extracted, the following list of files will be available in the folder  **DataENGSept2024**: |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Question1:** | **Question3:** |  |  |
|  | Question1\_P.dpr  Question1\_P.dproj  Question1\_P.res  Question1\_U.dfm  Question1\_U.pas | clsMonitor.pas  Question3\_P.dpr  Question3\_P.dproj  Question3\_P.res  Question3\_U.dfm  Question3\_U.pas |  |  |
|  |  |  |  |  |
|  | **Question2:** | **Question4:** |  |  |
|  | ConnectDB\_U.dfm  ConnectDB\_U.pas  OrdersDB.mdb  OrdersDB - Copy.mdb Question2\_p.dpr  Question2\_p.dproj  Question2\_p.res  Question2\_p\_Icon.ico  Question2\_u.dfm  Question2\_u.pas | Question4\_P.dpr  Question4\_P.dproj  Question4\_P.res  Question4\_P\_Icon.ico  Question4\_U.dfm  Question4\_U.pas  Storm1.txt  Storm2.txt  Storm3.txt  Storm4.txt  Storm5.txt |  |  |

|  |  |  |
| --- | --- | --- |
| **SECTION A** |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **QUESTION 1: GENERAL PROGRAMMING SKILLS** |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Do the following: |  |  |
| * Open the incomplete program in the **Question 1** folder. * Enter your name and surname as a comment in the first line of the **Question1\_U.pas** file. * Compile and execute the program. The program has no functionality currently.   Example of graphical user interface (GUI): |  |  |
| * Complete the code for each section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.5. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.1 | **Button [1.1 – Format and display]** |  |  |
|  |  |  |  |
|  | Write code to do the following: |  |  |
|  |  |  |  |
|  | * Add the text 'Web design' to be displayed in the panel **pnlQ1\_1** * Change the font of the text on the panel **pnlQ1\_1** to 'Consolas' * Change the text on the panel **pnlQ1\_1** to be displayed in bold and underlined * Change the text on the panel **pnlQ1\_1** to 24 point font size |  |  |
|  |  |  |  |
|  | Example of output: |  |  |
|  |  |  | (5) |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.2 | **Button [1.2 – Loop types]** |  |  |
|  |  |  |  |
|  | You are required to write code that evaluates the type of loop based on user input from a combo box **cmbQ1\_2**. The combo box allows the user to select either "Conditional" or "Unconditional".  Based on the selected item, the program should display an appropriate message using the ShowMessage procedure:   * If the user selects "Conditional", the message should display "While/Repeat Until loop". * If the user selects "Unconditional", the message should display "For loop". * If the user selects any other value, the message should display "Incorrect response". |  |  |
|  |  |  |  |
|  | Example of output if the option unconditional was selected: |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Example of output if an invalid option was selected: |  |  |
|  |  |  |  |
|  |  |  | (7) |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.3 | **Button [1.3 – Display sum of even numbers]** |  |  |
|  | You are required to write code that calculates the sum of even numbers entered by the user.  The program should prompt the user, using an input dialog box, to enter five integer numbers and calculate the sum of only the even numbers.  The final sum should be displayed in an edit box **edtQ1\_3**. |  |  |
|  |  |  |  |
|  | Example of output if the following numbers were entered:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Input numbers: | 1 | 5 | 7 | 2 | 4 | |  |  |
|  |  |  |  |
|  |  |  | (10) |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.4 | **Button [1.4 – Guessing game]**  A random number is generated and a user will then try and correctly guess what the number is. The user has three chances to guess what the number is before losing the game.  Write code to do the following:   * Randomly generate a value between 1 and 10 (both values included) * Use a loop and an input dialog box to allow the user three attempts to guess the random number generated between 1 and 10 * Display appropriate messages using ShowMessage dialog boxes to inform the user if their guess is 'Too high', 'Too low', or 'Correct' * After the game ends (either by finding the correct number or using up all three guesses), show a message indicating whether the user won or lost and display the number of guesses made | |  |  |
|  |  | |  |  |
|  | Example of input and output when user has correctly guessed the number (1) on the first attempt: | |  |  |
|  |  | |  |  |
|  | Input | Output |  |  |
|  |  | |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Example of input and output when user has correctly guessed the number (5) on the third attempt: | |  |  |
|  |  | |  |  |
|  | Input | Output |  |  |
|  |  | |  |  |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Example of output when user has guessed a number lower than the value generated: |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Example of output when user has guessed a number higher than the value generated: |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Example of output when user has incorrectly guessed the number (6) after the third attempt: |  |  |
|  |  |  |  |
|  |  |  | (11) |
|  |  |  |  |

|  |  |
| --- | --- |
| 1.5 | **Button [1.5 – Fibonacci sequence]** |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | You are required to write a Delphi program that generates the first ten numbers of the Fibonacci sequence and displays them in a list box **lstQ1\_5**.  Code has been provided to clear the list box **lstQ1\_5**.  The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding numbers, starting from 0 and 1.  Use a loop to generate the next eight numbers of the sequence. Update variables accordingly to get the next number in the sequence. Add each Fibonacci number to the list box **lstQ1\_5**. |  |  |
|  |  |  |  |
|  | Example of output: |  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | (7) |

|  |  |  |
| --- | --- | --- |
| * Enter your name and surname as a comment in the first line of the program file. * Save your program. * Print the code if required. |  |  |

|  |  |  |
| --- | --- | --- |
| **TOTAL SECTION A:** |  | **40** |

|  |  |  |
| --- | --- | --- |
| **SECTION B** |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **QUESTION 2: SQL AND DATABASE** |  |  |

|  |
| --- |
| Innovate Web Solutions (IWS) is a well-established web development company. The company specialises in creating custom websites, e-commerce platforms, and mobile applications for clients ranging from small businesses to large corporations.  IWS uses a database called **OrdersDB.mdb** to manage all their clients and orders. |

|  |  |  |
| --- | --- | --- |
| The data pages attached at the end of the question paper provide information on the design of the database and its contents. |  |  |
|  |  |  |
| Do the following: |  |  |
|  |  |  |
| * Open the incomplete project file called **Question2\_P.dpr** in the **Question 2** folder * Add your name and surname as a comment in the first line of the **Question2\_U.pas** unit file * Compile and execute the program. The program currently has limited functionality. The contents of the tables is displayed as shown below on the selection of **Tabsheet Question 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. | |  | |  |
| * 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** | | tblUsers | TADOTable | Refers to the table **tblUsers** | | tblOrders | TADOTable | Refers to the table **tblOrders** | |  | |  |
|  |  | |  |  |
| **2.1** | **Tabsheet [Question 2.1 - SQL]** | |  |  |
|  | Example of graphical user interface (GUI) for Question 2.1: | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  | **NOTE:**   * ONLY SQL statements may be used 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** on the next page: | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2.1.1 | **Button [Q2.1.1 All Users]** |  |  |
|  |  |  |  |  |
|  |  | Display all fields of all users in the **tblUsers** table sorted in alphabetical order according to the **Username**. |  |  |
|  |  |  |  |  |
|  |  | Example of output of the first five records: |  |  |
|  |  |  |  |  |
|  |  |  |  | (3) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 2.1.2 | **Button [Q2.1.2 Orders]** |  |  |
|  |  |  |  |  |
|  |  | Display the **OrderID**, **OrderDate** and **TotalAmount** of all orders that were placed by a user with a specific **UserID** (e.g., UserID = 1).  Code has been provided to extract and store the **UserID** in a variable called **sUserID**. |  |  |
|  |  |  |  |  |
|  |  | Example of output if the **UserID** entered was 1: |  |  |
|  |  |  |  |  |
|  |  |  |  | (4) |
|  |  |  |  |  |
|  | 2.1.3 | **Button [Q2.1.3 Average orders]** |  |  |
|  |  |  |  |  |
|  |  | Statistics about the average order amount are required by the managers of the company. |  |  |
|  |  |  |  |  |
|  |  | Calculate and display the **UserName** and average order amount stored in a new field called **Ave** **amount**, for each customer, of all orders placed in June 2023. The Ave amount field must be formatted to currency. |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Example of output: |  |  |
|  |  |  |  |  |
|  |  |  |  | (10) |
|  |  |  |  |  |
|  | 2.1.4 | **Button [Q2.1.4 Orders not completed]** |  |  |
|  |  |  |  |  |
|  |  | Clients with many orders are highly valued at the company. |  |  |
|  |  |  |  |  |
|  |  | Active orders are orders that has not been completed yet. Display all the clients that have orders and the total number of active orders of these clients. Display the result in a calculated field called **Active** **orders**. |  |  |
|  |  |  |  |  |
|  |  | Example output of the first five records: |  |  |
|  |  |  |  |  |
|  |  |  |  | (7) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2.1.5 | **Button [Q2.1.5 Remove orders]** |  |  |
|  |  |  |  |  |
|  |  | Remove all the records from the **tblOrders** table that have orders that are already completed. |  |  |
|  |  |  |  |  |
|  |  | Example output of the first five records after removing completed orders successfully: |  |  |
|  |  |  |  |  |
|  |  |  |  | (3) |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2.2 | **Tabsheet [Question 2.2 – Delphi code]** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Example of graphical user interface (GUI) for Question 2.2: |  |  |

|  |  |
| --- | --- |
|  |  |

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **NOTE:**   * Only Delphi programming code may be used to answer **QUESTION 2.2**. * NO marks will be awarded for SQL statements in **QUESTION 2.2**. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2.2.1 | **Button [2.2.1 – Display clients with orders]** | |  |  |
|  |  |  |  |  |
|  | It is important to make sure that the company keeps track of all orders placed.  Loop through the database tables and display each user with information on all the orders they have placed with the company.  Display all fees as currency. | |  |  |
|  | Display the result neatly in the rich edit component **redQ2\_2** as illustrated on the next page: | |  |  |
|  |  | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Example of output of the first two users: |  |  |
|  |  |  |  |
|  |  |  | (13) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| * Enter your name and surname 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** |  |  |

|  |  |  |
| --- | --- | --- |
| A website authoring company plans to offer hosting plans to their clients. Websites will be charged for the bandwidth they consume. The amount charged to the owners of the sites will therefore vary based on the number of visits the website receives. |  |  |

|  |  |  |
| --- | --- | --- |
| Do the following: |  |  |
| * Open the incomplete program in the **Question 3** folder. * Open the incomplete object class **clsMonitor**. * Enter your name and surname as a comment in the first line of the **Question3\_U.pas** file. * Compile and execute the program. The program has limited functionality currently.   Example of 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 explicitly stated in the question. |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | The incomplete object class (**clsMonitor**) provided contains the declaration of five attributes which describe a **Monitor** object.  These attributes for a **Monitor** object have been declared as follows:   |  |  | | --- | --- | | Attribute | Description | | fIP | Internet Protocol Address of the website. | | fCountryOfOrigin | The country from where the website is authored. | | fVisits | The number of times the site has been requested or viewed indirectly via a search engine. | | fDirectVisits | The number of times the site has been requested or viewed via its URL. | | fStatus | Indicating whether the website is active or accessible (TRUE status) or inactive and inaccessible (FALSE status) |   Code has been provided for the **getStatus**, **getIP** accessor methods and the **changeStatus** mutator method.  Complete the code in the object class as described in QUESTION 3.1.1 to QUESTION 3.1.4. |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Write code for a constructor method that will receive the IP address and country of origin as parameters.  Set the initial number of visits as well as the number of direct visits to 0 and the initial status to false. |  | (4) |
|  |  |  |  |  |
|  |  | Write code for a method called **addVisits** that will receive two parameter values:   * an integer value indicating the number of times the site was accessed either indirectly via a search engine or directly via a URL * A Boolean parameter when true, indicating that the website was accessed directly via a URL and when false, indicating that the website was accessed indirectly via a search engine.   Increment the appropriate attribute with the integer parameter received.  Direct and indirect visits can only be considered if the status of the website is true. |  | (6) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Write code for a method called **checkValidIP** that will determine if an IP address is in valid decimal notation.  A valid IP address will look as follows:   * An integer value each between 0 and 255 * A dot(.) * An integer value each between 0 and 255 * A dot(.) * An integer value each between 0 and 255 * A dot(.) * An integer value each between 0 and 255   Example: 125.235.12.199  The method must return a Boolean value indicating whether the IP address is valid or not. | |  | (8) |
|  |  |  | |  |  |
|  | 3.1.4 | Write code for a **toString** method that will return the attributes as a string in the following format:   * The IP and country of origin are displayed, separated by a colon. <IP>:<Country of Origin> * On a new line, display the number of visits via a search engine * On a new line, display the number of direct visits * On a new line, display whether the website is active or inactive   Example:  <IP address>:<Country of origin>  Number of visits via search engine: <Number of visits>  Number of visits via URL: <Number of direct visits>  Website accessibility: Active | |  |  |
|  |  | **NOTE**: | The website accessibility must indicate 'Active' when the <Status> is true, otherwise indicate 'Not active' |  | (5) |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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 **objMonitor**.  Write code to perform the tasks described in QUESTION 3.2.1 to QUESTION 3.2.4 below: |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Button [3.2.1 Instantiate]** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | An IP address for the websites consists of 4 numbers between 0 and 255 (both values included), separated by a full stop. E.g., 123.123.123.123  A combo box has also been provided preloaded with an extensive list of countries.  The following GUI is provided: |  |  |
|  | The user must enter the IP address in the edit field provided and select a country of origin.  Do the following:   * Extract the IP address from the edit box **edtQ3\_2\_1** * Extract the country of origin from the combo box **cmbQ3\_2\_1**. * Use the information extracted to instantiate the object **objMonitor**. * Display a message indicating that the object has been instantiated.   Example of output: |  | (5) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Button [3.2.2 Toggle website state]** |  |  |
|  | |  |  |  |
|  | | The circular shape in the GUI below is used to indicate the status of the website. A lime green circle indicates a TRUE status whereas a red circle will indicate a FALSE status.    Do the following:   * Call the **changeStatus** method of the object. * Use the appropriate method to determine the status of the object. * If the status of the object is TRUE, change the colour of the circle to lime green. * If the status of the object is FALSE, change the colour of the circle to red. |  | (4) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 3.2.3 | **Button [3.2.3 Valid IP]** |  |  |
|  | |  |  |  |
|  | | A relevant method must be called to determine whether the IP address is valid or invalid. An appropriate message must be displayed in the panel **pnlQ3\_2\_3** to indicate whether the IP address is valid or not, using an appropriate method to display the IP address in the message.  Example of output if the IP address (300.123.123.123) is not valid: |  | (4) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 3.2.4 | **Button [3.2.4 Add Visit]** |  |  |
|  | |  |  |  |
|  | | The user will be able to increase the number of visits, by using an input box. If the website includes direct access, the checkbox **cbxQ3\_2\_4** must be checked.  Code has been provided to extract and store the number of visits using a dialog box in the variable called **iNumOfVisits**. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Do the following:   * Extract the boolean value from the checkbox. * Call the **addVisits** method on the object using the provided **iNumOfVisits** variable and the extracted Boolean value as arguments. * Clear the lines property of the rich edit. * Call the **toString** method on the object and display the resulting string in the rich edit. |  |  |
|  |  |  |  |
|  | Example of input: |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Example output if the website was accessed 100 times directly: |  | (4) |
|  |  |  |  |
|  |  |  |  |
| * Enter your name and surname 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** |  |  |

|  |
| --- |
| You are responsible for the Storm Monitoring Application of a remote island in the Pacific Ocean. |

|  |  |  |
| --- | --- | --- |
| Do the following: |  |  |
|  |  |  |
| * Open the incomplete program in the **Question 4** folder. * Enter your name and surname number as a comment in the first line of the **Question4\_U.pas** file. |  |  |
| * Compile and execute the program. Currently, the program has no functionality. |  |  |

Example of graphical user interface (GUI):

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| You have been provided with the following array: |  |  |
|  |  |  |
| |  |  |  | | --- | --- | --- | | Array | Size | Description | | **arrWindSpeed** | [1 .. 5, 1 .. 2] | This two-dimensional array contains the category of a hurricane and the wind speeds in mph. | |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| You have been provided with five text files that contain maps of islands namely: **Storm1.txt**, **Storm2.txt**, **Storm3.txt**, **Storm4.txt** and **Storm5.txt**.  Example of content of Storm1.txt:    The '#' represents the island.  The '–' represents the sea.  The '1' represents the storm. |  |  |

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Button [4.1 - Level]** |  |  |
|  |  |  |  |
|  | You have been provided with a combo box, **cmbQ4\_1**, that contains the names of the island maps with the storm that is approaching.  Example:    The user must select a map from the combo box **cmbQ4\_1**.  The category of the storm must be determined by counting the number of '1's that are found in the selected map.  Write code to do the following:   * Loop through the selected text file   + Display the content of each line of the text file (map) selected in the memo **memQ4**   + Determine the category of the storm * Display the category of the storm in the rich edit **redQ4** and set the spin edit **spnQ4\_1** to the storm category. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Example of the output in the memo **memQ4** if **Storm1** is selected from the combo box **cmbQ4\_1**. |  | (15) |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Button [4.2 - Wind]** |  |  |
|  |  |  |  |
|  | You have been provided with the array **arrWindSpeed** thatcontains the category of the storm and the wind speed range of the storm in mph.  arrWindSpeed: array [1 .. 5, 1 .. 2] of String =  (('1', '74 to 95 mph'),  ('2', '96 to 110 mph'),  ('3', '111 to 129 mph'),  ('4', '130 to 156 mph'),  ('5', '157 mph or higher'));  The miles per hour needs to be converted to kilometres per hour.  Write code to do the following:   * Extract the category of the storm from the spin edit **spnQ4\_1**. * Extract the mph from the corresponding index of the array **arrWindSpeed** and convert the wind speed to km/h. * Display the km/h values in **redQ4** rounded off to two decimals. |  |  |
|  | Code has been provided to clear the rich edit component **redQ4**. |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **NOTE**: | Hardcoding will not receive any marks. The mph needs to be extracted from the correct index in the array **arrWindSpeed**. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Example of output if **Storm1** is selected: |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Example of output if **Storm4** is selected: |  |  |
|  |  |  |  |
|  |  |  | (15) |

|  |  |  |
| --- | --- | --- |
| * Enter your name and surname 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:**  **GRAND TOTAL:** |  | **30**  **150** |

**INFORMATION TECHNOLOGY P1**

**DATABASE INFORMATION QUESTION 2:**

The design of the database tables is as follows:

Table: **tblUsers**

This table contains the information about the users.

|  |  |  |
| --- | --- | --- |
| **Field name** | **Data type** | **Description** |
| UserID | Number | A unique code assigned to each user. |
| Username | Text (10) | The username of the client. |
| Password | Text (10) | The password of the client. |
| Email | Text (30) | The e-mail address of the client. |
| RegistrationDate | Text (20) | The registration date of a client using the service. |

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

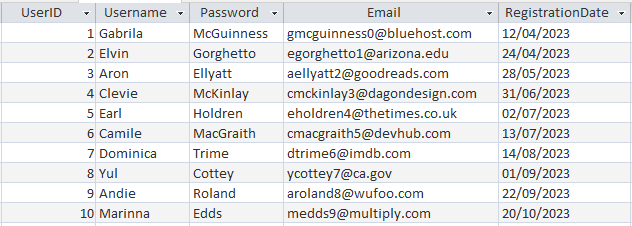
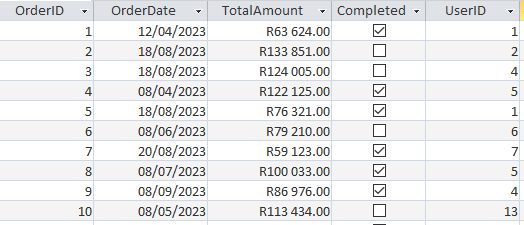


Table: **tblOrders**

This table contains the information about each order as they were received by the company.

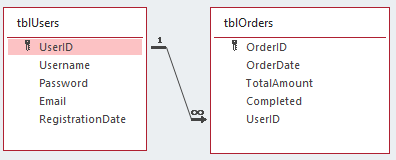
|  |  |  |
| --- | --- | --- |
| **Field name** | **Data type** | **Description** |
| OrderID | Number | A unique number assigned to each order. |
| OrderDate | Date/Time | The date when the order was first received. |
| TotalAmount | Date/Time | The total amount payable from each order. |
| Completed | Yes/No | Indicates whether the order has been completed or not. |
| UserID | Text (5) | Indicates with which user the order is associated. |

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



**NOTE**: Connection code has been provided.

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



**Pages for planning purposes**

**Page 1**

**Pages for planning purposes**

**Page 2**