**YOUR SCHOOL EMBLEM**

**YOUR SCHOOL NAME**

**SEPTEMBER EXAMINATION – 2019**

**INFORMATION TECHNOLOGY**

**PAPER 1 – PRACTICAL**

TIME: 3 HOURS MARKS: 150

**This paper consists of 24 pages, and there are 4 Delphi projects in your exam folder.**

***Make sure that you have a complete set.***

|  |  |  |
| --- | --- | --- |
|  | **INSTRUCTIONS AND INFORMATION** |  |
| 1. | This question paper is divided into FOUR sections. You must answer 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. | Make sure that you answer the questions according to the specifications that are given in each question. Marks will only be awarded according to the set of requirements. |  |
| 4. | Answer only what is asked for in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation. |  |
| 5. | 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. |  |
| 6. | **Routines such as search, sort and selection** must be developed from first principles. You may not use the built-in features of the programming language for any of these routines. |  |
| 7. | All data structures must be defined by you, the programmer, unless the data structures are supplied. |  |
| 8. | You must **save your work regularly** on the disk space allocated to you for the examination OR the flash you have been given. |  |
| 9. | Make sure that your **name and surname** appears as a comment in every program that you code. |  |
| 10. | At the end of the examination session, you must make sure that all your work has been saved on the disk space allocated to you for this examination session OR you must hand in the flash with all your work saved on it. **Ensure that all the** **files can be read**. |  |
| 11. | The files you need to complete this question paper have been given to you on the disk space allocated to you in the form of a password-protected executable file: **DelphiDataS19ENG.exe** |  |
|  | **Do the following:**   * Double click on the password-protected executable file. * Click on the ‘extract’ button. * Enter the following password: **!!++!!September** |  |
|  | |  |  |  |  | | --- | --- | --- | --- | | Question 1  Question1\_P.dpr  Question1\_P.res  Question1\_U.dcu  Question1\_U.dfm  Question1\_U.pas | Question 2  DB.mdb  DBBackup.mdb  Question2\_P.dpr  Question2\_P.res  Question2\_U.dcu  Question2\_U.dfm  Question2\_U.pas  TDBConnection\_U.dcu  TDBConnection\_U.pas | Question 3  PaintJob\_U.dcu  PaintJob\_U.pas  Question3\_P.dpr  Question3\_P.res  Question3\_U.dcu  Question3\_U.dfm  Question3\_U.pas | Question 4  MajorDams.txt  Question4\_p.dpr  Question4\_p.res  Question4\_u.dcu  Question4\_u.dfm  Question4\_u.pas | |  |
|  | **Change the name** of theLearner Data folder to your **Surname and Name**. |  |

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

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

|  |
| --- |
| **SCENARIO:**  A smart home is a residence that uses Internet-connected devices to enable the remote monitoring and management of appliances and systems, such as lighting and heating. A company wants to develop an interface to manage a smart home and needs developers to help with the basic design. |

|  |  |
| --- | --- |
| Do the following:   * Open the incomplete program in the **Question1** 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 currently has no functionality. * Follow the instructions below to complete the code for **EACH** section of **QUESTION** **1**, as described in **QUESTION 1.1** to **QUESTION 1.5**. |  |
| An example of the GUI is given below: |  |

|  |  |  |
| --- | --- | --- |
|  | |  |
| 1.1 | **BUTTON - [Invite Home Member]** |  |
|  | Write code for the following:   * Display the text “**Welcome to the Smart House App**” on **lblMember**. * Change the text colour of **lblMember** to grey. * Change the date of the label **lblDate** to today’s date. * The date must be in the format “**ddd d/mm/yyyy**”. * Change **btnMembers** so that it does not display on the form.   **EXAMPLE OUTPUT:** |  |
|  |  | (5) |
|  |  |  |
| 1.2 | **BUTTON - [Question 1.2]** |  |
| 1.2.1 | One of the most common questions regarding a smart home is about the electricity usage.  Write code to do the following:   * Extract the number of hours from **spnLights**. * Calculate the amount of **kilowatts** used during the **month**. * Kilowatts per month is calculated as follows:   Kilowatts per month = 0.06 \* ***hours*** \* 30   * Display the **kilowatts** formatted to **TWO** decimalson **lblWatts** in the following format:   <calculated kilowatts> kWh/m | (4) |
|  |  |  |
| 1.2.2 | A power company charges 10 cents per kWh, or R 0.10.  Write code to do the following:   * Calculate the amount charged by multiplying 0.10 to the kilowatts per month. * Display the result on **lblAmount** in **CURRENCY** format.   **EXAMPLE OUTPUT:** | (2) |
|  |  |  |
| 1.3 | **BUTTON - [Question 1.3]** |  |
|  | The developers want to include a weather report on the Smart Home interface to provide the user with the option to choose between Celsius and Fahrenheit.  Write code for the following:   * Check that the user selected a unit of temperature and display an appropriate message when nothing is selected. * Generate **TWO** random values in **Celsius** for the minimum and the maximum temperatures for the day’s weather forecast. Use the upper and lower limits as indicated in the table on the next page.  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Min Temperature** | |  | **Max Temperature** | | | Lower limit | Upper limit |  | Lower limit | Upper limit | | 18 | 22 |  | 32 | 38 |  * When the user selects Fahrenheit, use the following formula to convert the minimum and maximum values to Fahrenheit:   **Fahrenheit** = (**Celsius** × 9/5) + 32   * Display the minimum and maximum values on **lblMin** and **lblMax**. Both these values need to be **ROUNDED DOWN** to the nearest whole number. * Display the corresponding unit in **lblDegrees.**   **EXAMPLE OUTPUT:** |  |
|  | **Note:** Due to the fact that random numbers are used, your output might be different. | (9) |
|  |  |  |
| 1.4 | **BUTTON - [Question 1.4]** |  |
|  | The developers want to encourage users to be more active and decided to add directions for a jogging route on the Smart Home interface.  Write code for the following:   * Extract the number of sides for the shape from **cmbShape**. * Determine whether the user wants to run clockwise or anti-clockwise from the **chkClock** component. * Extract the route’s total distance from **edtDistance**. * The distance before a turn is calculated using the following formula:   **distance before turn = total distance / number of sides**   * Display the route’s directions on the **redMap** component in the following format: |  |
|  | Start Route  Run for <distance> km  Turn Left (Anti Clock wise) / Turn Right (Clock wise)  ...  Route Finished |  |
|  | **EXAMPLE OUTPUT:** | (13) |
|  |  |  |
| 1.5 | **BUTTON - [Question 1.5]** |  |
|  | The developers want to add an alarm to the Smart Home interface. |  |
|  | Write code to compile the alarm’s name:   * Obtain the radio station or song’s name from **edtMusic**. * Ensure the user enters a value and display an appropriate error message when nothing was entered. * The user must be prompted to enter a time via an input dialog, set the default value of the input dialog to 8:00am.      * Check whether a radio station or song is selected using the **chkFm** component. * The format for the alarm name if **chkFm** has been selected is as follows:   <alarm time><first four characters of radio station>FM   * The format for the alarm name if **chkFM** has NOT been selected is as follows:   <alarm time><first four characters of song>mp3   * Generate the alarm name and display it on the **pnlInfo** component. |  |
|  |  | |
|  | **EXAMPLE 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** |  |  |
|  |  |  |

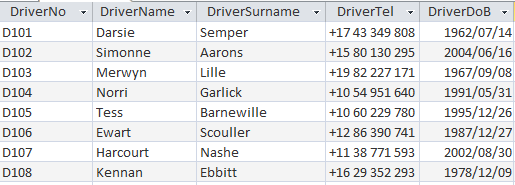
|  |  |  |
| --- | --- | --- |
| **QUESTION 2: SQL AND DATABASE**  This section consists of two questions. The following important notes are applicable to both questions:   * You are **NOT** allowed to modify or add to the supplied data in any way. * Good programming techniques must be applied when coding your solutions. * **NO marks will be assigned for hardcoding**. Use control structures and variables where necessary. |  |  |

|  |
| --- |
| The database, **DB.mdb**, contains the information of the drivers and their exotic and rare cars. The database contains two tables, namely **tblDrivers** and **tblCars**. |

The **tblDrivers** table is structured with the following fields:

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATA TYPE** | **DESCRIPTION** |
| DriverNo (PK) | Text | Unique number for each driver. |
| DriverName | Text | Stores the driver’s first name. |
| DriverSurname | Text | Stores the driver’s surname. |
| DriverTel | Text | Stores the driver’s telephone number. |
| DriverDoB | Date/Time | Date of birth of the driver. |

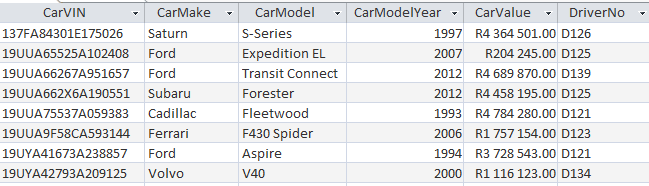
Example data from **tblDrivers** table:



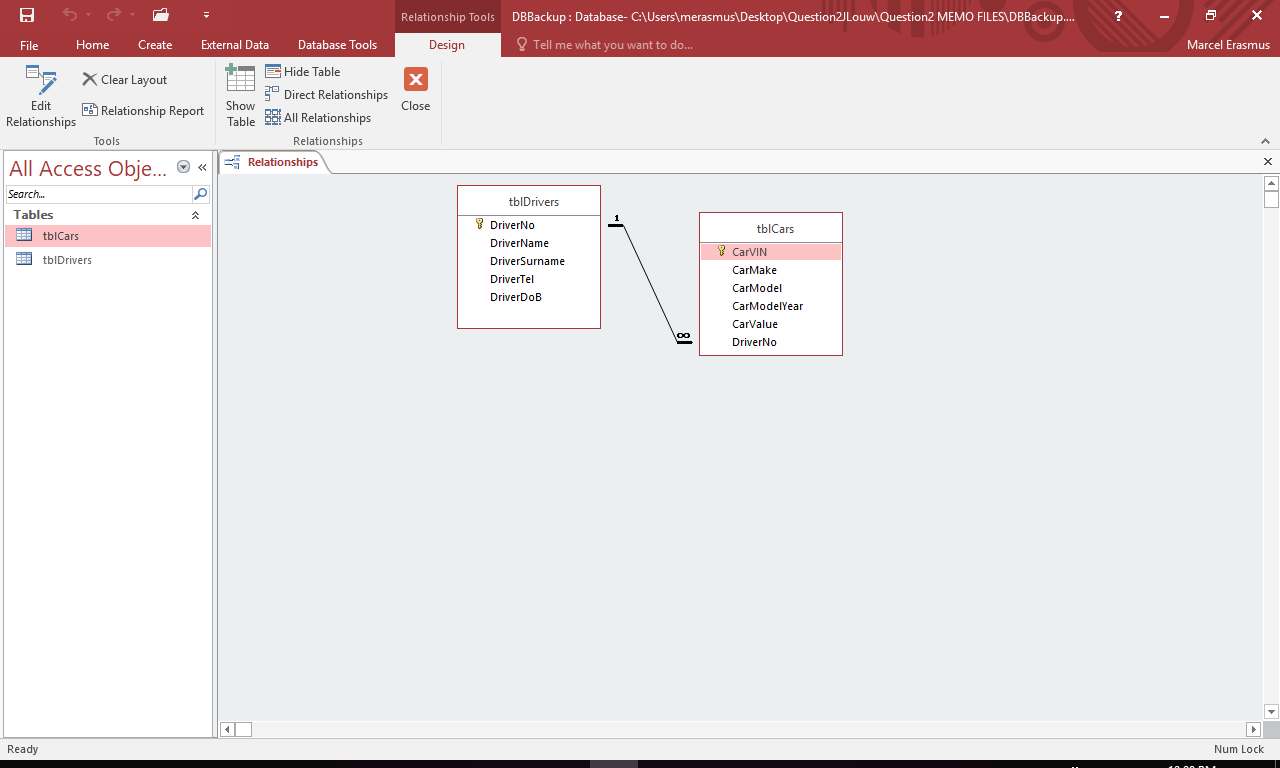
The **tblCars** table is structured with the following fields:

|  |  |  |
| --- | --- | --- |
| **FIELD** | **DATA TYPE** | **DESCRIPTION** |
| CarVIN (PK) | Text | The car’s unique number. |
| CarMake | Text | The fabricate of the car. |
| CarModel | Text | The model of the car. |
| CarModelYear | Text | The year the car was made. |
| CarValue | Currency | The current value of the car. |
| DriverNo (FK) | Text | A reference to a driver’s unique number. |

Example data from **tblCars** table:



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



|  |
| --- |
| **NOTE:**   * Connection code has been provided. * The database is password-protected - therefore you will not be able to access the database directly. * When the **btnDBRestore** button is clicked, the data in the database will be restored to the original data. |

|  |
| --- |
| Do the following: |
| * Compile and execute the program in the **Question2** folder. The program currently has limited functionality. * Complete the code for each question as described in **QUESTION 2.1** and **QUESTION** **2.2**. |
|  |

|  |
| --- |
| The program contains a graphical user interface with two tab sheets labelled  **Question 2.1** and **Question 2.2**. |

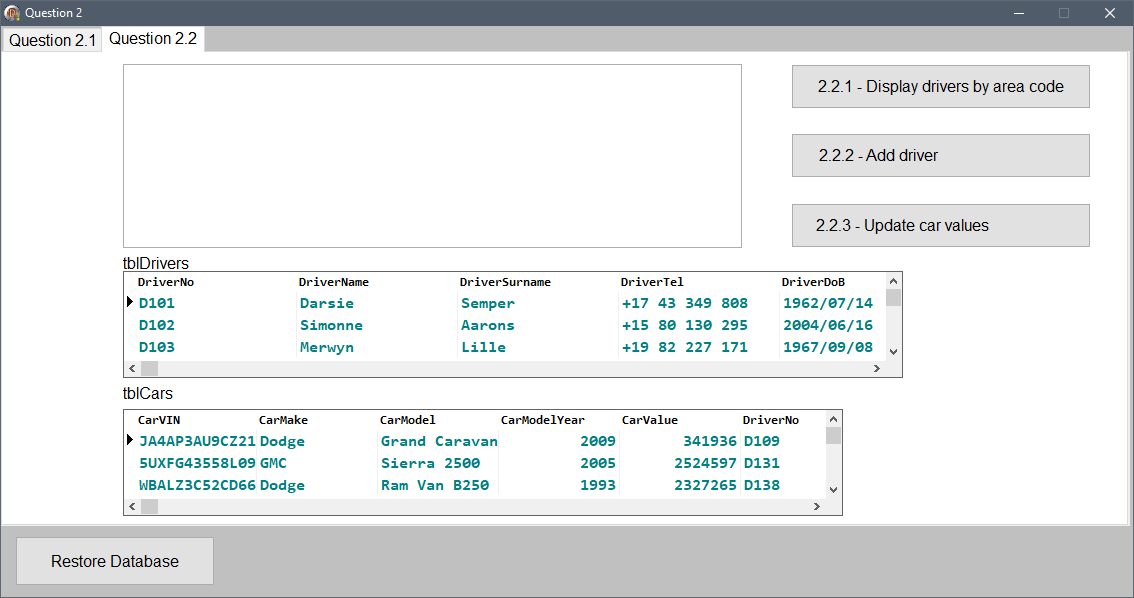
|  |  |
| --- | --- |
| **QUESTION 2.1 STRUCTURED QUERY LANGUAGE**  Select tab sheet **Question 2.1**, which displays the following user interface: |  |
| Complete the SQL code to meet the requirements specified in **QUESTION 2.1.1 to QUESTION 2.1.5**. |  |

|  |  |  |
| --- | --- | --- |
| 2.1.1 | **BUTTON [2.1.1 – Driver alphabetical list]**  Display ALL the details of the drivers in the **tblDrivers** table, sorted alphabetically according to the driver’s surname.  Example output of first four records: | (4) |
|  |  |  |
| 2.1.2 | **BUTTON [2.1.2 – Car models on selected date]**  Display the car make and car model of all the cars manufactured in 1959.  Example of output: | (3) |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 2.1.3 | **BUTTON [2.1.3 - Cars with high value]**  Display the car make, model and value of the cars with a value of R 2 000 000 or more. Format the car’s value as currency and display the field name as **CurrentValue**.  Example output of the first four records: | (6) |
|  |  |  |
| 2.1.4 | **BUTTON [2.1.4 - Car makes]**  Display a list of all the car makes, along with the number of times they feature in the **tblCars** table.  Example of output: | (3) |
|  |  |  |
| 2.1.5 | **BUTTON [2.1.5 - Ford owners]**  Display the name, surname and car model of all the drivers that own a **Ford** car make.  Example output of first four records: | (5) |
|  |  |  |
|  |  | **[21]** |

**QUESTION 2.2 DATABASE MANIPULATION (NO SQL)**

Select tab sheet **Question 2.2**, which displays the following user interface:



**NOTE:**

* NO marks will be allocated for the use of SQL statements for **QUESTION 2.2**.
* The names of the tables to be used in your code must be **tblDrivers** and **tblCars**, which are **TADOTable** objects connected to the database.

Complete the code to meet the requirements specified in **QUESTION 2.2.1** to **QUESTION 2.2.3**.

|  |  |  |
| --- | --- | --- |
| 2.2.1 | **BUTTON [2.2.1 - Area code]**  Write code to display the name, surname and date of birth of all the drivers whose telephone numbers begin with **+15**.  The data must be displayed in the **redQ2** component in neat columns as shown in the example of output.  **NOTE:** Code for the headings are provided.  Example of output for the first four lines of output: | (7) |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 2.2.2 | **BUTTON [2.2.2 - Add driver]**  Write code to add a record to the **tblDrivers** table.  The data of the driver to be added is provided below:  **Driver no**: D150  **Driver name**: Jule  **Driver surname**: Orris  **Driver telephone**: +51 87 580 3783  **Driver date of birth**: 1972/11/25  Example of the last few records in **tblDrivers** after adding the record for Jule: | (3) |
|  |  |  |
| 2.2.3 | **BUTTON [2.2.3 - Update value]**  Write code to increase all the cars values according to the table below:   |  |  | | --- | --- | | **Car year model** | **% value needs to increase** | | Before 2010 | 15% | | 2010 and later | 10% |   Example of the last few records in **tblCars** after updating the values: | (9) |
|  |  |  |
|  |  | **[19]** |

|  |  |  |
| --- | --- | --- |
| * 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** |

**QUESTION 3: OBJECT-ORIENTED PROGRAMMING**

|  |
| --- |
| **SCENARIO**  A company specialising in painting marine parts for boats and ships requires software to calculate the price of a given paint job they have to complete. |

The incomplete project **Question3\_P** is provided in your **Question3** exam folder. It contains:

* The incomplete main form unit called **Question3\_U.pas**
* The incomplete object class called **PaintJob\_U.pas**

Currently the program has no functionality.

Complete the code for each section of **QUESTION 3** as specified in **QUESTION 3.1** and **QUESTION 3.2** below.

|  |  |  |
| --- | --- | --- |
| 3.1 | An incomplete class called **PaintJob\_U.pas** has been provided. Complete the code in the provided class called **TPaintJob** as described in **QUESTION 3.1.1** to **QUESTION 3.1.4**.  The following attributes have been provided for the **TPaintJob** class: |  |
|  | |  |  | | --- | --- | | **ATTRIBUTE** | **DESCRIPTION** | | **fCustomer** | Full name (first name and last name) of the customer requesting the paint job. | | **fPaintType** | Type of paint to be used. | | **fSurfaceArea** | Surface size that needs to be painted in m2. | | **fCode** | Unique code for the job being done. | |  |
|  | **NOTE:**  A completed **toString** method has been provided in the **TPaintJob** class. |  |
| 3.1.1 | Write code for a constructor method that will receive the full name of the customer, the type of paint to be used and the area to be painted as parameters.  Assign the received values to the respective attributes. | (4) |
|  |  |  |
| 3.1.2 | Write code for a method called **calculateCost** that receives an **Integer** value as a parameter and returns the **cost of the job**.   * The received integer value is the number of layers that must be painted. * Multiply the number of layers by the surface area to calculate the total amount of area to paint. * The resulting area must be multiplied by the cost of the type of paint per square meter (shown in the table below).  |  |  | | --- | --- | | **PAINT TYPE** | **COST PER SQUARE METER (m2)** | | Vinyl | R1254.60 | | Polyurethane | R748.35 | | CoalTar | R322.62 | |  |
|  | Calculate and return the paint job’s cost as result. | (8) |
|  |  |  |
| 3.1.3 | Write code for a method called **generateCode** that compiles and stores a **string** in the **fCode** attribute in the following format:  **%%$$\*\***  The description for the different sections in the format of the code is as follows:   * **%%** represents the first **TWO** letters of the paint type in **LOWER CASE**. * **$$** represents the **FIRST** and **LAST** letters of the customer’s full name. * **\*\*** represents **TWO** random numbers from 0 to 9 (both inclusive).   Example of a generated code with the following attribute values:   |  |  | | --- | --- | | **PAINT TYPE:** | Polyurethane | | **CUSTOMER FULL NAME:** | Floyd Khanyiso | | **GENERATED CODE:** | poFo54 |   Example of a generated code with the following attribute values:   |  |  | | --- | --- | | **PAINT TYPE:** | Coaltar | | **CUSTOMER FULL NAME:** | Barry Davids | | **GENERATED CODE:** | coBs20 |   **Note:** Due to the fact that random numbers are used, your output might be different. | (8) |
|  |  |  |
| 3.1.4 | Write code for a method called **getCode** that will return the paint job’s unique code stored in the **fCode** attribute (generated in **QUESTION 3.1.3**) as a **string**. | (2) |

|  |  |  |
| --- | --- | --- |
|  | An incomplete form class **Question3\_U** is provided with the following graphical user interface: |  |
| 3.2 | Complete the form class by doing the following: |  |
| 3.2.1 | **BUTTON [3.2.1 – Create paint job object]**  The user must enter the customer’s full name, the surface area of the job and the type of paint needed using the components provided.  The following input data has been provided in the design of the graphical user interface to test the program:   * Customer Name: Floyd Khanyiso * Surface Area: 37 * Paint Type: Vinyl   A global **TPaintJob** variable called **objPaintJob** has been provided.  Write code to do the following:   * Use the data that has been entered by the user to instantiate the **objPaintJob** object. * Use the **toString** method to display the details of the object in the rich edit component **redQ3**. | (7) |
|  |  |  |
| 3.2.2 | **BUTTON [3.2.2 - Identification code]**  Write code to do the following:   * Call the **generateCode** method to generate a unique code for the paint job. * Display the compiled identification code in the edit box **edtCode** by using the appropriate method.   **EXAMPLE OUTPUT:**  Example of output if the test data provided as input values in **QUESTION 3.2.1** was used: |  |
|  |  | (3) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3.2.3 | **BUTTON [3.2.3 – Calculate Cost]**  The user must choose the number of layers to be painted in the edit box **edtLayers**.  Code has been provided that declares a constant variable called **iMaxLayers** which represents the maximum number of layers a paint job can be painted with. It is currently set to the value of 7.  Write code to do the following:   * Extract the number of layers chosen from the **edtLayers** component. * Determine if the extracted number of layers can be added without exceeding the maximum number of layers that can be used. * In case the number of layers does not exceed the maximum allowed, call the relevant method to calculate the cost of the paint job. * Display the total price in the **lblCost** component, formatted to **TWO** decimal places. * If the number of layers exceeds the maximum, display a suitable error message on the **lblCost** component.   For each of the tests below click **BUTTON [3.2.1 – Create paint job object]** then **BUTTON [3.2.3 – Calculate Cost]**  **EXAMPLE 1:**  Test data:   |  |  | | --- | --- | | **SURFACE AREA:** | 37 | | **PAINT TYPE:** | Vinyl | | **LAYERS:** | 3 |   Example of output on **lblCost** based on the test data above:    **EXAMPLE 2:**  Test data:   |  |  | | --- | --- | | **SURFACE AREA:** | 37 | | **PAINT TYPE:** | Polyurethane | | **LAYERS:** | 5 |   Example of output on **lblCost** based on the test data above:    **EXAMPLE 3:**  Test data:   |  |  | | --- | --- | | **SURFACE AREA:** | 37 | | **PAINT TYPE:** | CoalTar | | **LAYERS:** | 7 |   Example of output on **lblCost** based on the test data above:    **EXAMPLE 4:**  Test data:   |  |  | | --- | --- | | **SURFACE AREA:** | 37 | | **PAINT TYPE:** | Vinyl | | **LAYERS:** | 8 |   Example of output on **lblCost** based on the test data above: |  |
|  |  | (8) |

|  |  |  |
| --- | --- | --- |
| * 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 C:** | **[40]** |

# QUESTION 4: PROBLEM SOLVING

|  |
| --- |
| **SCENARIO:**  The City of Cape Town uses a system called Western Cape Water Supply System (WCWSS) to monitor the water levels of the dams that supply the city with water.  There are six main dams and eight smaller dams that supply water to the City of Cape Town. This question will focus on the six main dams. |
|  |
| * Compile and execute the program in the **Question4** folder. * The program currently has no functionality. * The following graphical user interface is displayed: |

The program contains the declarations of three arrays called **arrDams**, **arrDamLevels** and **arrCapacities**. All three of the arrays are declared with a maximum size of 6 elements:

* **arrDams** is declared for the **SIX MAIN DAMS** that supply Cape Town with water.
* **arrDamLevels** is declared for current water level of each dam as a percentage.
* **arrCapacities** is declared for the individual water storing capacities of each dam in **MILLIONS OF CUBIC METERS**.

The following two arrays are declared with a maximum size of 5 elements:

* **arrYears** contains the past five years (2015 – 2019).
* **arrAverages** contains the average percentage of the water levels for each of the main dams as indicated at the end of July of each year.

|  |
| --- |
| var  arrYears: array [1..5] of string = ('2019', '2018', '2017', '2016', '2015');  arrAverages: array [1..5] of real = (53.1, 25.1, 42.4, 53.5, 0); |

|  |
| --- |
| **NOTE:**  **arrTempDams**, **arrTempDamLevels** and **arrTempCapacities** are three constant arrays that are populated with values and must ONLY be used to answer **QUESTION 4.2** and **QUESTION 4.3** if your code to populate **arrDams**, **arrDamLevels** and **arrCapacities** in **QUESTION 4.1** was unsuccessful. |

A supplied text file called **MajorDams.txt** contains 6 lines of data. Each line contains the name of a dam, the dam’s full capacity and the percentage water level of the dam. The data in each line is provided in the following format:

<dam name>;<full capacity>,<water level>

Example of the contents of the text file:

|  |
| --- |
| Berg River;130.010,89.5  Steenbras Lower;33.517,53.2  Steenbras Upper;31.767,80.8  Theewaterskloof;480.188,50.5  Voelvlei Dam;164.095,64.0  Wemmershoek;58.644,51.4 |

Explanations of the first two lines of data in the **MajorDams.txt** text file:

* Berg River dam’s full capacity is 130.010 million cubic meters and its water level is 89.5%.
* Steenbras Lower dam’s full capacity is 33.517 million cubic meters and its water level is 53.2%.

Complete the code for the requirements described in **QUESTION 4.1** to **QUESTION 4.3**.

|  |  |  |  |
| --- | --- | --- | --- |
| 4.1 | **BUTTON [4.1 - Load and display dam data]**  Write code for the following:   * Perform a test to determine if the text file **MajorDams.txt** is accessible. If it is not accessible, display a suitable message indicating that the text file is not accessible and leave the procedure. * Make use of the text file **MajorDams.txt** to populate the arrays **arrDams**, **arrDamCapacities** and **arrDamLevels** with the appropriate data. * Code has been provided to display the headings as indicated in the example of output.      * Display the contents of **arrDams**, **arrDamCapacities** and **arrDamLevels** in the **redQ4\_1** component as shown in the example of output. * Determine the total capacity of the main dams. * Display the calculated total capacity on the **redQ4\_1** component as indicated in the example of output. |  | |
|  | **EXAMPLE OF OUTPUT:** |  | (14) | |
|  |  |  |  | |
| 4.2 | **BUTTON [4.2 - Display water level]**  The user has to select a dam from the combo box **cmbDams**.  The purpose of the program is to calculate and display the contents of the selected dam in **millions of cubic meters**.   |  | | --- | | **NOTE:**  The Steenbras dam consists of two sub dams, namely Steenbras Lower and Steenbras Upper. |   Write code for the following:   * Calculate the contents of the dam(s) that has been selected by the user from the **cmbDams** component. * Use the values stored in **arrDamLevels** and **arrCapacities** to calculate the contents of the dam(s) in **millions of cubic meters**. * Display the percentage of the water level(s) between brackets (see the examples of output). * If a dam consists of sub dams, all of the sub dams’ contents must be calculated and displayed (see the examples of output). * Display the resulting output with appropriate headings on the **redQ4\_2** component as indicated in the example of output below.   **EXAMPLES OF OUTPUT:** |  |  | |
|  |  |  |  | |
|  |  |  | (8) | |
|  |  |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| 4.3.1 | **BUTTON [4.3.1 - Display yearly averages]**  The average dam level percentages are kept for five years.  The years and average water level percentages are stored in the **arrYears** and **arrAverages** arrays.  The data in the **arrAverages** array has been captured incorrectly:   * The average percentage for 2019 contains the data for 2018. * 2018 contains the data for 2017. * 2017 contains the data for 2016. * 2016 contains the data for 2015. * The data for 2015 is currently 0%. |  |  |
|  | The average dam level percentage for 2019 is 59.8%.  Write code for the following:   * Adapt the **arrAverages** array so that the correct water level percentages are stored for the correct corresponding years:   + The percentage water level for 2019 must be stored in the first element of the array.   + The rest of the values must be stored correctly in elements 2 to 5. * Display the years and average water level percentages on the **redQ4\_3\_1** output component.   **EXAMPLE OF OUTPUT** |  | (4) |
|  |  |  |  |
| 4.3.2 | **BUTTON [4.3.2 - Determine and display days]** |  |  |
|  | The number of days that water can be provided to the City of Cape Town must be calculated according to the **TOTAL** water level percentage as indicated in the **rTOT** (59.8%) constant that has been provided.  The city’s daily water usage is 0.38% of the **TOTAL** water level percentage. A local constant, called **rDaily**, has been provided and contains the value for the City’s daily water usage (0.38%).  **NOTE:**   * The last 10% of a dam’s water is not suitable for human consumption.   Write code for the following:   * Calculate the total number of days for which the City of Cape Town can be supplied with water if the total water level percentage remains constant, rounded down to the lowest integer. * Display the calculated number of days with a suitable message on **pnlDays**.   **EXAMPLE OF OUTPUT:** |  |  |
|  |  |  | (4) |

|  |  |  |
| --- | --- | --- |
| * 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 (QUESTION 4): 30**

**GRAND TOTAL: 150**