



# basic education

Department:  
Basic Education  
**REPUBLIC OF SOUTH AFRICA**

**NATIONAL  
SENIOR CERTIFICATE**

**GRADE 12**

**INFORMATION TECHNOLOGY P1**

**NOVEMBER 2024**

**MARKS: 150**

**TIME: 3 hours**

**This question paper consists of 25 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. The duration of this examination is three hours. Because of the nature of this examination, it is important to note that you will not be permitted to leave the examination room before the end of the examination session.
4. This question paper is set with programming terms that are specific to the Delphi programming language. The Delphi programming language must be used to answer the questions.
5. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements.
6. Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.
7. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper.
8. Routines, such as search, sort and selection, must be developed from first principles. You may NOT use the built-in features of the Delphi programming language for any of these routines.
9. All data structures must be defined by you, the programmer, unless the data structures are supplied.
10. You must save your work regularly on the disk/CD/DVD/flash disk you have been given, or on the disk space allocated to you for this examination session.
11. Make sure that your examination number appears as a comment in every program that you code, as well as on every event indicated.
12. If required, print the programming code of all the programs/classes that you completed. Your examination number must appear on all printouts. You will be given half an hour printing time after the examination session.
13. At the end of this examination session, you must hand in a disk/CD/DVD/flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read.

14. The files that you need to complete this question paper have been provided to you on the disk/CD/DVD/flash disk or on the disk space allocated to you. The files are provided in the form of password-protected executable files.

Do the following:

- Double click on the following password-protected executable file:  
**DataNOV2024.exe**
- Click on the 'Extract' button.
- Enter the following password: **#RUN4@New**

Once extracted, the following list of files will be available in the folder **DataNOV2024**:

**Question 1:**

Houses.txt  
Question1\_P.dpr  
Question1\_P.dproj  
Question1\_P.res  
Question1\_U.dfm  
Question1\_U.pas

**Question 2:**

ConnectDB\_U.pas  
MarathonsDB - Copy.mdb  
MarathonsDB.mdb  
Question2\_P.dpr  
Question2\_P.dproj  
Question2\_P.res  
Question2\_U.dfm  
Question2\_U.pas

**Question 3:**

MRecord\_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 limited 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 on the next page.

1.1 Write code for buttons **btnQ1\_1\_1** and **btnQ1\_1\_2** as specified below.

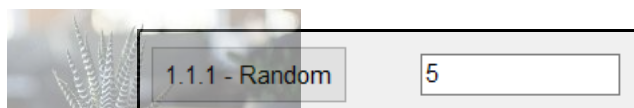


### 1.1.1 Button [1.1.1 - Random]

Write code to do the following:

- Generate a random number from 5 to 10, both numbers included.
- Display the random number in the edit box **edtQ1\_1\_1**.

Example of output if the random number 5 was generated:



**NOTE:** The output may differ due to the nature of random numbers.

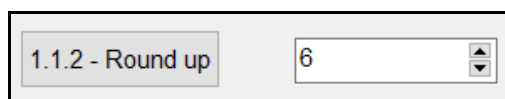
(4)

### 1.1.2 Button [1.1.2 - Round up]

You have been provided with a constant **NUMBER = 5.23247**.

Write code that uses a mathematical function to display the value of the constant, rounded up to the next whole number in the spin edit **spnQ1\_1\_2**.

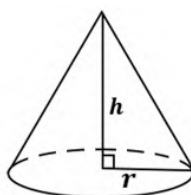
Example of output:



(3)

## 1.2 Button [1.2 - Surface area]

A right-angled solid cone is depicted in the image below.



The surface area of the cone is calculated using the following formula:

$$A = \pi r(r + \sqrt{h^2 + r^2})$$

Write code to do the following:

- Extract the values of **r** and **h** from the provided edit boxes **edtQ1\_2\_r** and **edtQ1\_2\_h** respectively.
- Use the values of **r** and **h** and appropriate mathematical functions to calculate the surface area of the cone represented by **A**.
- Display the value of **A** in the label **lblQ1\_2**, rounded to TWO decimal places.



Example of input and output where  $r = 5$  and  $h = 10$ :

$r =$ <input type="text" value="5"/>	$h =$ <input type="text" value="10"/>
1.2 - Surface area	
254.16	

Example of input and output where  $r = 7.9$  and  $h = 22.4$ :

$r =$ <input type="text" value="7.9"/>	$h =$ <input type="text" value="22.4"/>
1.2 - Surface area	
785.56	

(8)

### 1.3 Button [1.3 - Read file]

A text file called **Houses.txt** contains the street address and the number of bedrooms for each house in a suburban area.

The format of the content of the text file is as follows:

```
<street address>
<number of bedrooms>
```

Example of the first four lines from the **Houses.txt** text file:

```
61 Loop street
4
83 Church street
2
```

Code has been provided for the declaration of the text file variable **tFile**.

Write code to do the following:

- Assign the text file **Houses.txt** to the variable **tFile**.
- Ensure that reading from the text file starts from the first line in the text file.
- Read through the text file using a loop and save the street address and number of bedrooms into separate variables.
- Display the street address and the number of bedrooms in the rich edit **redQ1\_3** in the format below.

```
<street address> - <number of bedrooms> -bedroom house
```

Example of output:

61 Loop street - 4-bedroom house  
 83 Church street - 2-bedroom house  
 17 School street - 3-bedroom house

(9)

#### 1.4 Button [1.4 - Add name]

Learners at a school are going on an excursion. Two rich edit components **redQ1\_4\_P** and **redQ1\_4\_NP** have been provided to display the names of the learners who have paid their fees and the learners who have not paid their fees respectively.

The combo box **cmbQ1\_4** contains the names of all the learners going on the excursion.

Write code to do the following:

- Extract the name selected from the combo box **cmbQ1\_4**.
- If the 'Paid' check box **chbQ1\_4** is checked, the name of the learner must be added to the **redQ1\_4\_P** component, otherwise the name must be added to the **redQ1\_4\_NP** component.
- Remove the selected name from the combo box **cmbQ1\_4**.

Example of input and output if Bianca was selected and her excursion fees were already paid:

Name: Bianca	Paid: <input checked="" type="checkbox"/>
--------------	---

Paid	Not paid
Bianca	

Example of input and output if Godfrey was selected and his excursion fees were not paid:

Name: Godfrey	Paid: <input type="checkbox"/>
---------------	--------------------------------

Paid	Not paid
Bianca	Godfrey



Example of the content of **cmbQ1\_4** after Bianca and Godfrey were selected and their names have been removed from the combo box:



(7)

### 1.5 Button [1.5 - Replace]

A password must be compiled for each learner attending the excursion.

The user must enter the name and surname of the learner in the edit box **edtQ1\_5**.

Stanmorephysics.com

The following code has been provided:

- The content of the edit box **edtQ1\_5** has been saved in the variable **sNameSurname**.
- A string **sCharacters** that contains six special characters

```
sCharacters := '@#$$%^&';
```

Write code to do the following to create the password:

- Reverse the characters and remove all the spaces in the **sNameSurname** string.
- Display the reversed string, without spaces, in the **memQ1\_5** output area.
- Replace every third character in the reversed string with a special character randomly selected from the provided **sCharacter** string.
- Display the password in the **memQ1\_5** output area.

Example of input and output:

(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

The organisers of athletic events require an application for running enthusiasts to provide them with information about marathons and the locations.

A database called **MarathonsDB.mdb**, which contains information about marathon names and the cities where they are held, has been developed. It allows users to find marathons in their preferred cities.

The database contains two tables called **tblLocations** and **tblMarathons**.

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

Do the following:

- Open the incomplete project file called **Question2\_P.dpr** in the **Question 2** folder.
- Enter your examination number as a comment in the first line of the **Question2\_U.pas** unit file.
- Compile and execute the program. The program has no functionality currently. The contents of the tables are displayed, as shown below on the selection of tab sheet **2.2 - Delphi code**.

Database programming

2.1 - SQL 2.2 - Delphi code

LocationID	City	Province	Altitude	Population
1	Cape Town	Western Cape	42	4618000
2	Durban	KwaZulu-Natal	8	3721000
3	Bloemfontein	Free State	1395	598000
4	Pretoria	Gauteng	1339	2473000

MarathonID	MarathonName	MarathonDate	Distance	PrizeMoney	Organiser	LocationID
1	East London Marathon	2024/09/12	28.4	R50 000.00	Running South Africa	15
2	Joburg Run	2024/10/26	55.2	R12 000.00	Active Events	6
3	Cape Town Challenge	2025/01/18	40.6	R20 000.00	Runners United	1
4	Durban Ultra Marathon	2025/02/19	45.8	R10 000.00	Endurance Sports SA	2

2.2.1

2.2.1 - Remove marathons

2.2.2

2.2.2 - Qualifying marathons

Restore database Close

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

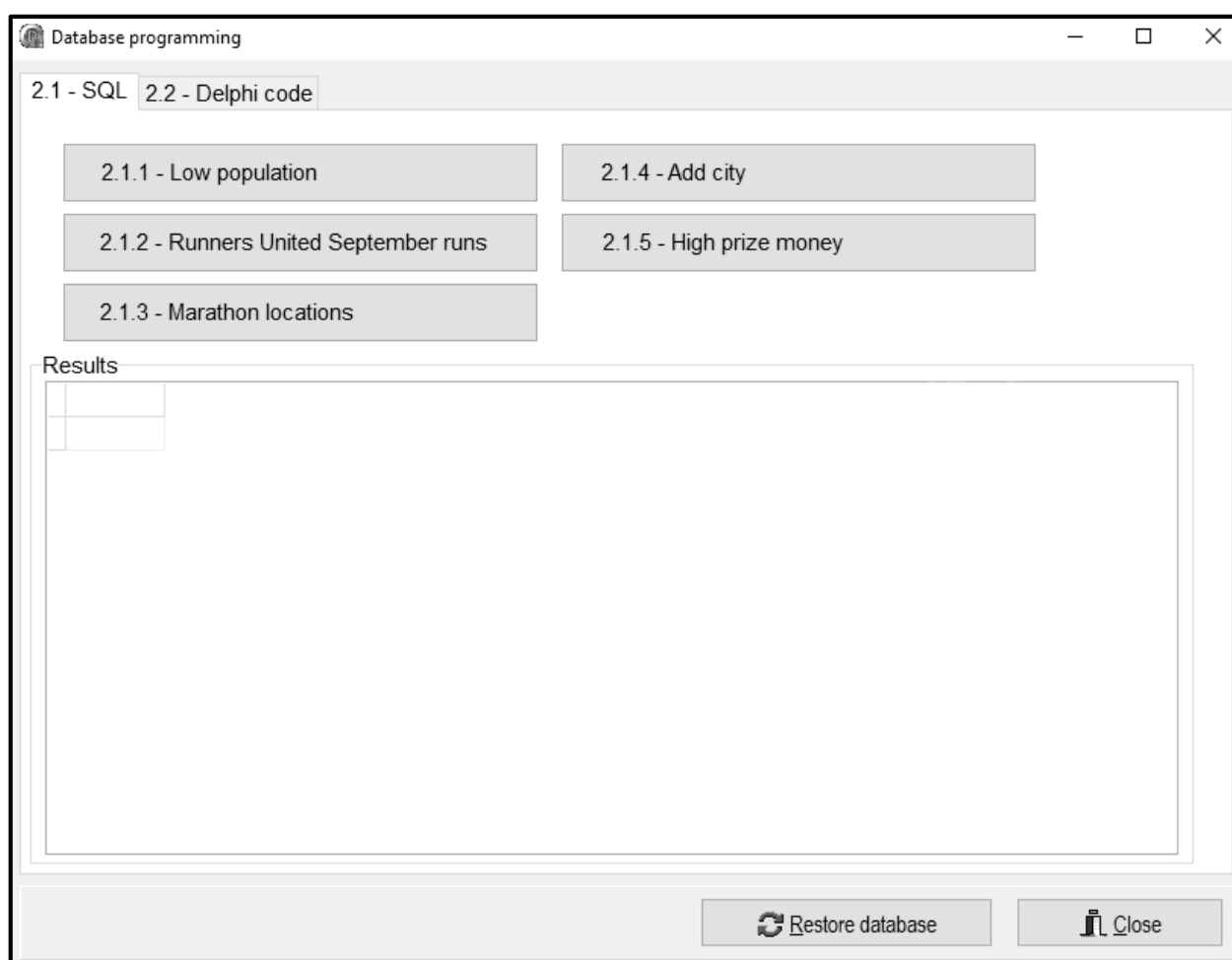
**NOTE:**

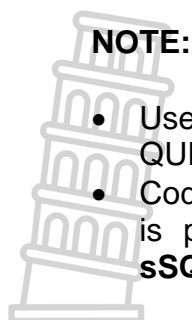
- The 'Restore database' button is provided to restore the data contained in the database to the original content.
- The content of the database is password-protected, i.e. you will NOT be able to gain access to the content of the database using Microsoft Access.
- Code is provided to link the GUI components 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
tblLocations	TADOTable	Refers to the table <b>tblLocations</b>
tblMarathons	TADOTable	Refers to the table <b>tblMarathons</b>

**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 - Low population]**

Display all fields of the records in the **tblLocations** table with a **population** of less than 200 000 people.

Example of output:

LocationID	City	Province	Altitude	Population
9	Mbombela	Mpumalanga	1653	110159
11	Potchefstroom	North West	1127	67372
12	Kroonstad	Free State	1532	107514
17	Upington	Northern Cape	2742	87301

(3)

**2.1.2 Button [2.1.2 - Runners United September runs]**

Display the **MarathonID**, **MarathonDate** and **Distance** of all the marathons organised by Runners United taking place during September.

Example of output:

MarathonID	MarathonDate	Distance
13	2025/09/14	22.7
26	2025/09/27	70.2

(4)

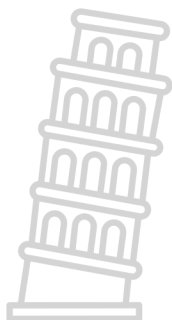
**2.1.3 Button [2.1.3 - Marathon locations]**

The location of each marathon is required by the organisers.

Concatenate the **City** field and the first three letters of the **Province** field of all the locations in the **tblLocations** table, in a new field called **Location**.

The content of the new **Location** field must be in the following format:

<City> - <First three letters of province>



Example of output of the first five records:

Location
Polokwane - Lim
Mbombela - Mpu
eMalahleni - Mpu
Potchefstroom - Nor
Kroonstad - Fre

(4)

#### 2.1.4 Button [2.1.4 - Add city]

A city must be added to the **tblLocations** table. The details are provided in the table below.



Field	Value
LocationID	19
City	Welkom
Province	Free State
Altitude	1198
Population	423016

Example of output of the last five records after the city has been added:

15	East London	Eastern Cape	18	394034
16	Kimberly	Northern Cape	1116	284509
17	Upington	Northern Cape	2742	87301
18	Klerksdorp	North West	1297	226215
19	Welkom	Free State	1198	423016

(4)

#### 2.1.5 Button [2.1.5 - High prize money]

Display the city, the number of marathons and the total prize money of all the marathons held in the city, only if the total prize money is more than R50 000.

The new field for the number of marathons is **NumMarathons** and the new field for the total prize money is **Total Prize Money**.

Example of output:

City	NumMarathons	Total Prize Money
East London	2	64500
Mbombela	3	68500

(7)

## 2.2 Tab sheet [2.2 - Delphi code]

### NOTE:

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

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

LocationID	City	Province	Altitude	Population
1	Cape Town	Western Cape	42	4618000
2	Durban	KwaZulu-Natal	8	3721000
3	Bloemfontein	Free State	1395	598000
4	Pretoria	Gauteng	1339	2473000

MarathonID	MarathonName	MarathonDate	Distance	PrizeMoney	Organiser	LocationID
1	East London Marathon	2024/09/12	28.4	R50 000.00	Running South Africa	15
2	Joburg Run	2024/10/26	55.2	R12 000.00	Active Events	6
3	Cape Town Challenge	2025/01/18	40.6	R20 000.00	Runners United	1
4	Durban Ultra Marathon	2025/02/19	45.8	R10 000.00	Endurance Sports SA	2

2.2.1

2.2.1 - Remove marathons

2.2.2

2.2.2 - Qualifying marathons

Restore database Close

### 2.2.1 Button [2.2.1 - Remove marathons]

Some organisers withdrew from the marathons and these marathons' records must be removed as they will no longer take place.

The organiser's name must be entered by the user.

Code has been provided to extract and store the name of the organiser, using an input dialog box. The name of the organiser is stored in the variable **sOrganiser**.

Write code to remove the records from the **tblMarathons** table, where the name of the organiser is the same as the name of the organiser entered.



Example of the first four records in the **tblMarathons** table before the organiser, Active Events, is deleted:

MarathonID	MarathonName	MarathonDate	Distance	PrizeMoney	Organiser	LocationID
1	East London Marathon	2024/09/12	28.4	R50 000.00	Running South Africa	15
2	Joburg Run	2024/10/26	55.2	R12 000.00	Active Events	6
3	Cape Town Challenge	2025/01/18	40.6	R20 000.00	Runners United	1
4	Durban Ultra Marathon	2025/02/19	45.8	R10 000.00	Endurance Sports SA	2

Example of the first four records in the **tblMarathons** table after the organiser, Active Events, is deleted:

MarathonID	MarathonName	MarathonDate	Distance	PrizeMoney	Organiser	LocationID
1	East London Marathon	2024/09/12	28.4	R50 000.00	Running South Africa	15
3	Cape Town Challenge	2025/01/18	40.6	R20 000.00	Runners United	1
4	Durban Ultra Marathon	2025/02/19	45.8	R10 000.00	Endurance Sports SA	2
5	Johannesburg Run Fest	2025/03/15	75.9	R13 000.00	Outdoor Adventures	6

(5)

### 2.2.2

#### Button [2.2.2 - Qualifying marathons]

The runners training for long-distance marathons must compete in qualifying marathons to be eligible for entry. A qualifying marathon is any marathon that is run over a distance of 40 km or more.

A city name must be entered using an input dialog box and then all the qualifying marathons in that city must be displayed.

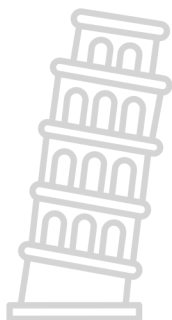
Code has been provided to:

- Clear the content of the **redQ2\_2\_2** component
- Extract and store the city name that was entered in the variable **sCity**

Use the **redQ2\_2\_2** component to display the output.

Write code to do the following:

- Display the marathon name and distance of all the qualifying marathons from the **tblMarathons** table, for the city name that was entered, using the input box provided.
- Display a suitable message if the city name entered cannot be found in the **tblLocations** table.



Example of output if Paarl is entered as input:

2.2.2	
2.2.2 - Qualifying marathons	
Paarl Peak Pursuit	40.2
Worcester Wild Run	48.1

Example of output if Sasolburg is entered as input:

2.2.2	
2.2.2 - Qualifying marathons	
Sasolburg was not found.	

(13)

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

**TOTAL SECTION B: 40**



## SECTION C

### QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

You have been tasked by the World Marathon Association to assist with managing the information of record holders in marathons.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Open the incomplete object class **MRecord\_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question3\_U.pas** file and the **MRecord\_U.pas** file.
- Compile and execute the program. The program has limited functionality currently.

Example of the graphical user interface (GUI):

The screenshot shows a graphical user interface titled "World Marathon Association". It is divided into three main sections: 3.2.1, 3.2.2, and 3.2.3. Section 3.2.1 contains input fields for "Marathon name:" (Karoo Marathon), "Record holder:" (Sam Peterson), "Record date:" (2024/06/13), "Record time:" (02:08:38), and "Distance:" with radio buttons for "21.1 km" and "42.2 km" (the latter is selected). Below these is a button labeled "3.2.1 - Instantiate object". Section 3.2.2 contains a button labeled "3.2.2 - Pace". Section 3.2.3 contains input fields for "Name:" and "Time:", and a button labeled "3.2.3 - Check record". At the bottom right, there is a small image of a marathon runner crossing a finish line.

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

**NOTE:** You are NOT allowed to add any additional attributes or user-defined methods, unless explicitly stated in the question.

- 3.1 The provided incomplete object class (**MRecord**) contains the declaration of five attributes, which describe a **Marathon Record** object.

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

Attribute	Type	Description
fMarathonName	String	The name of the marathon
fRecordHolder	String	The name of the person that currently holds the record
fRecordDate	String	The record date in the format yyyy/MM/dd
fRecordTime	String	The record time in the format hh:mm:ss
fDistance	Real	The distance of the marathon

The following methods have also been provided:

- A partially completed **constructor** method
- The mutator methods **setRecordHolder**, **setRecordDate** and **setRecordTime**
- An auxiliary method **toMinutes** that receives a runner's time as a string parameter, in the format hh:mm:ss, and returns the record time in minutes as a real data type

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

- 3.1.1 The incomplete **constructor** method receives five parameters.

Write code to assign the parameters to the **fMarathonName**, **fRecordHolder**, **fRecordDate**, **fRecordTime** and **fDistance** attributes respectively.

(3)

- 3.1.2 Write code for an accessor method called **getRecordTime** to return the **fRecordTime** attribute.

(2)

- 3.1.3 Write code for a method called **checkRecord** that receives a runner's time as a string parameter and returns a Boolean value.

The method should compare the current record time stored in the **fRecordTime** attribute, with the time received as a parameter. Return 'True' if the runner's time received as a parameter is less than the **fRecordTime** attribute value and 'False' if not.

(5)



3.1.4

The runner's pace is the speed at which a runner completes a marathon, measured in minutes per kilometre.

The following formula can be used to calculate a runner's pace:

$$pace = \frac{minutes}{distance}$$

Write code for a method called **calcPace** that uses the provided **toMinutes** method and the provided formula to calculate and return the runner's pace as a real value.

(4)

3.1.5

Write code for a method called **toString** to return a string in the following format:

```
<fMarathonName> - <fDistance> km: <fRecordHolder>
(<fRecordTime> on <fRecordDate>)
```

Example:

Karoo Marathon - 42.2 km: Sam Peterson (02:08:38 on 2024/06/13)

(5)

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 **objMRecord**.

Code has been provided in **btnQ3\_2\_1**, **btnQ3\_2\_2** and **btnQ3\_3\_3** to clear the output area.

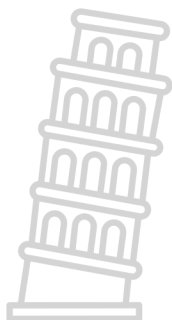
Write code to perform the tasks described in QUESTION 3.2.1 to QUESTION 3.2.3 that follow.

### 3.2.1 Button [3.2.1 - Instantiate object]

Sam Peterson is the current record holder of the Karoo Marathon. The current record information for the Karoo Marathon is recorded as follows:

Marathon name	Karoo Marathon
Record holder	Sam Peterson
Date record was set	2024/06/13
Marathon record time	02:08:38
Distance of marathon	42.2 km

Code has been provided to extract the name of the marathon, the name of the record holder, the date that the record was set and the record time that was recorded from the relevant edit boxes.



Write code to do the following:

- Extract the marathon distance from the radio group **rgpQ3\_2\_1** as a real value. The unit 'km' must not be saved as part of the distance value.
- Use the information extracted to instantiate the **objMRecord** object.
- Display the details of the object in the output area **redQ3**, using the **toString** method.

Example of input and output:

3.2.1

Marathon name:

Record holder:

Record date:

Record time:

Distance: ☐ 21.1 km ☒ 42.2 km

Karoo Marathon - 42.2 km: Sam Peterson (02:08:38 on 2024/06/13)

(8)

### 3.2.2 Button [3.2.2 - Pace]

The pace of the record holder needs to be calculated.

Write code to call the **calcPace** method and display the result, rounded to THREE decimal places, in the **redQ3** component in the following format:

Record holder's pace: <Pace of record holder> min/km



Example of input and output:

(4)

### 3.2.3 Button [3.2.3 - Check record]

The details of the current record holder object must be replaced if another runner completes the marathon in a shorter time.

The user must enter the name and the time it took the new runner to complete the marathon in the edit boxes **edtQ3\_2\_3\_Name** and **edtQ3\_2\_3\_Time** respectively.

Write code to do the following:

- Call the **checkRecord** method with the new runner's time as an argument to determine if the new runner's time entered is less than the current record time of the object.
- If the new runner's time is less than the current record time:
  - Update the current record object by using the following methods and arguments:

Methods	Arguments
setRecordHolder	New runner's name
setRecordTime	New runner's time
setRecordDate	System date

- Display the updated object information in the output area **redQ3** using the **toString** method.



- If the new runner's time is more than the object's current record time, display the current record time in the **redQ3** component in the following format:

The current record time remains: <Current record time>

Example of input and output if the new runner's time is **more** than the current record time:

The screenshot shows a form titled '3.2.3' with a background image of a potted plant. The form has two input fields: 'Name:' with the value 'Adam Smith' and 'Time:' with the value '02:10:15'. Below these fields is a button labeled '3.2.3 - Check record'.

The current record time remains: 02:08:38

Example of input and output if the new runner's time is **less** than the current record time:

The screenshot shows a form titled '3.2.3' with a background image of a potted plant. The form has two input fields: 'Name:' with the value 'Adam Smith' and 'Time:' with the value '02:07:15'. Below these fields is a button labeled '3.2.3 - Check record'.

Karoo Marathon - 42.2 km: Adam Smith (02:07:15 on 2024/10/05)

(9)

- 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**

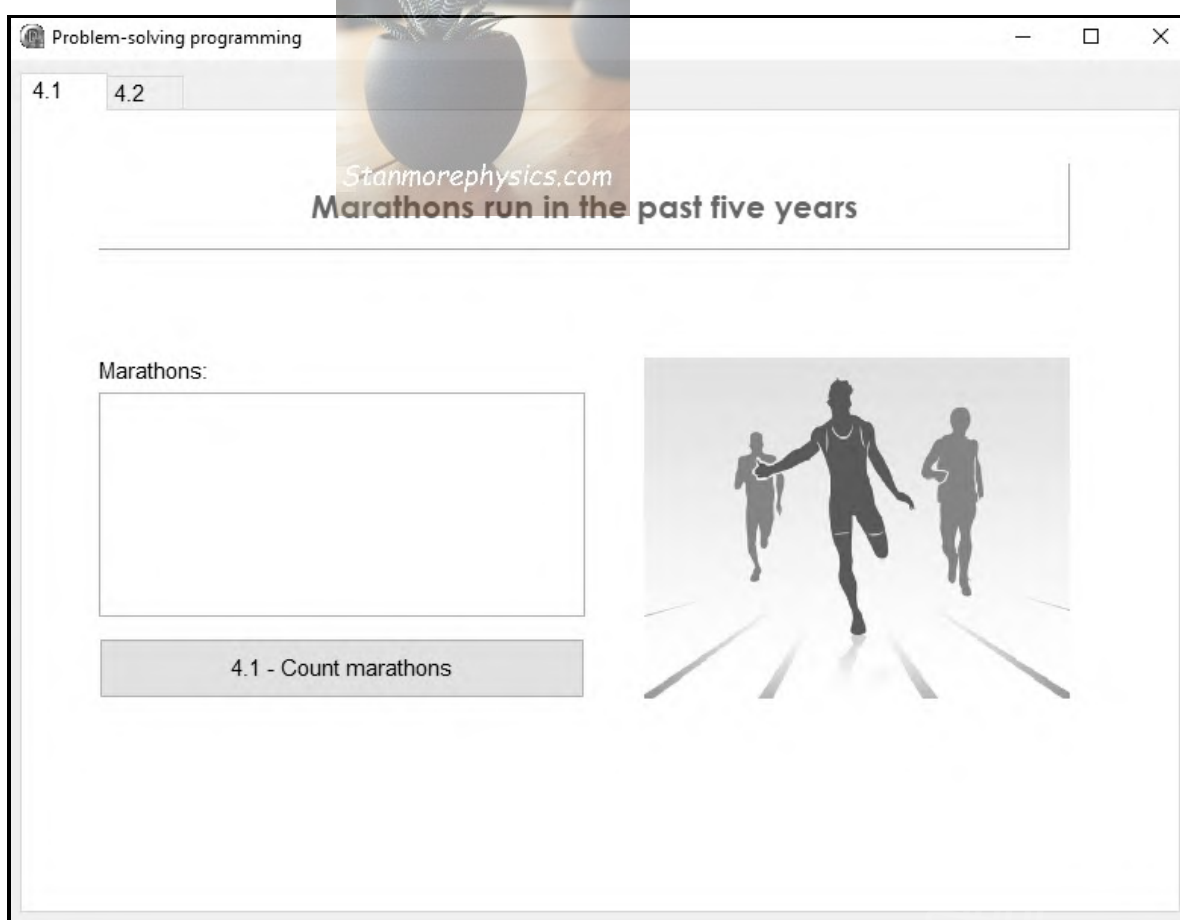
## SECTION D

### 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 **Quest4\_U.pas** file.
- Compile and execute the program. The program has limited functionality currently.

Example of the graphical user interface (GUI):



The following arrays have been provided in the program:

- An array, **arrMarathons**, which contains the names of ten marathons that have been completed by a participant in the past. It must be noted that a participant may have participated in a marathon more than once.

```
arrMarathons: array [1 .. 10] of String = ('Wally Hayward',  
'Sasol', 'Soweto', 'Jacaranda City', 'Sasol', 'Durban City', 'Soweto',  
'Soweto', 'Wally Hayward', 'Soweto');
```



- A two-dimensional array, **arrChar**, that stores alphabetical characters. The names of nine different marathons are hidden in the rows of this array, amongst the other characters.

```
arrChar: array [1 .. 14, 1 .. 14] of char = (
('u', 'x', 'v', 'm', 's', 'a', 's', 'o', 'l', 'f', 'k', 'j', 't', 'r'),
('u', 'm', 'g', 'e', 'n', 'i', 'w', 'a', 't', 'e', 'r', 'd', 's', 'e'),
('g', 'v', 'o', 'e', 't', 'v', 'a', 'n', 'a', 'f', 'r', 'i', 'k', 'a'),
('e', 'p', 'o', 'y', 'i', 'l', 'c', 'k', 'h', 'j', 's', 'd', 'f', 'd'),
('n', 'k', 'n', 'y', 's', 'n', 'a', 'f', 'o', 'r', 'e', 's', 't', 'u'),
('i', 's', 'y', 'd', 'b', 'c', 'r', 'g', 'h', 'k', 'c', 's', 'a', 'r'),
('w', 'a', 'l', 'l', 'y', 'h', 'a', 'y', 'w', 'a', 'r', 'd', 's', 'b'),
('a', 's', 'q', 'r', 't', 'n', 'n', 'j', 'h', 'e', 'r', 't', 'h', 'a'),
('t', 'o', 'e', 'r', 'y', 'b', 'd', 'r', 'h', 'k', 'l', 'g', 'd', 'n'),
('e', 'j', 'a', 'c', 'a', 'r', 'a', 'n', 'd', 'a', 'c', 'i', 't', 'y'),
('r', 'y', 'j', 'f', 'g', 'f', 'c', 'f', 'g', 'u', 'h', 'v', 'c', 'i'),
('k', 'h', 'h', 'l', 'p', 'h', 'i', 'l', 'l', 'c', 'r', 'e', 's', 't'),
('a', 'd', 'e', 'v', 'd', 's', 'o', 'w', 'e', 't', 'o', 'm', 'k', 'y'),
('p', 'd', 'u', 'r', 'b', 'a', 'n', 'c', 'i', 't', 'y', 'z', 'c', 'l'));
```

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

#### 4.1 Button [4.1 - Count marathons]

Use the provided array called **arrMarathons** that contains the names of ten marathons, which a runner has participated in over the past five years. The runner took part in some of the marathons more than once, hence the repetition of marathon names in the array.

Code has been provided for the alignment and to display the headings in the **redQ4\_1** rich edit.

Write code to do the following:

Display the marathon name and the number of times the runner participated in each marathon in the **redQ4\_1** component, as shown in the output below.

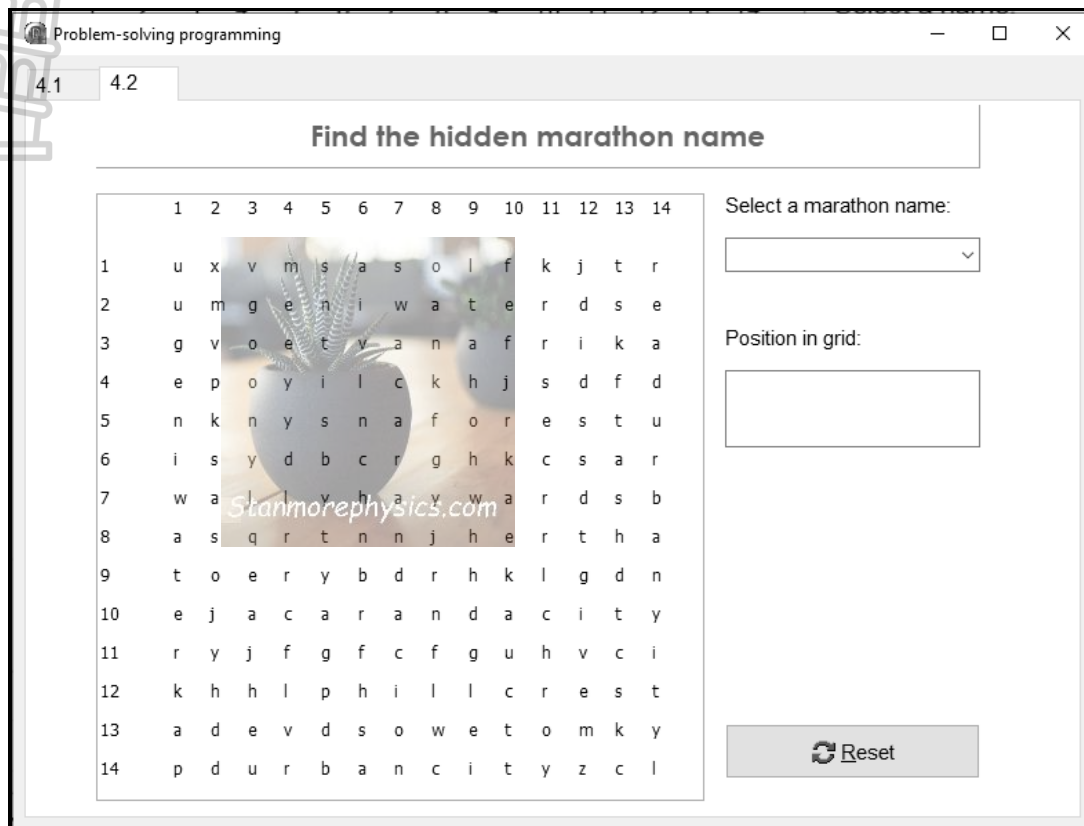
Example of output for the content in the array **arrMarathons**:

Marathon name	Number of times
Wally Hayward	2
Sasol	2
Soweto	4
Jacaranda City	1
Durban City	1

(11)

4.2 A mystery word puzzle game was developed to create awareness of the different marathons.

Example of the graphical user interface (GUI):



**NOTE:** Code has been provided for a method called **display2D** that displays the array **arrChar** in the rich edit **redQ4\_2**.

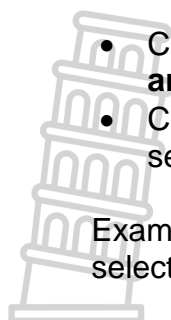
### Combo box [cmbQ4\_2]

The mystery word puzzle game uses characters placed in the provided two-dimensional array called **arrChar**, containing 14 rows and 14 columns. Nine marathon names are provided in the combo box **cmbQ4\_2** and these names are also hidden among the characters in the rows of the array **arrChar**. The hidden marathon names are placed horizontally in the rows of the array **arrChar** from left to right.

Write code to do the following:

- Extract the selected marathon name from the combo box **cmbQ4\_2**.
- Find the marathon name selected that is hidden in array **arrChar**.
- Use the **memQ4\_2** to display the row and column values of the selected word in the following format:

Row <Row index> @ column <Start col index> to <End col index>



- Change the characters of the selected marathon name in the array **arrChar** to upper case.
- Call the method **display2D** to display the array **arrChar** to show the selected word in **upper case**.

Example of input and output if the marathon name 'knysnaforest' was selected from the combo box **cmbQ4\_2**:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	u	x	v	m	s	a	s	o	l	f	k	j	t	r
2	u	m	g	e	n	i	w	a	t	e	r	d	s	e
3	g	v	o	e	t	v	a	n	a	f	r	i	k	a
4	e	p	o	y	i	l	c	k	h	j	s	d	f	d
5	n	K	N	Y	S	N	A	F	O	R	E	S	T	u
6	i	s	y	d	b	c	r	g	h	k	c	s	a	r

Select a marathon name:

knysnaforest ▼

Position in grid:

Row 5 @ column 2 to 13

(19)

**NOTE:** A **Reset** button was provided to reset the characters in the two-dimensional array back to lower case.

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

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

## INFORMATION TECHNOLOGY P1

### DATABASE INFORMATION FOR QUESTION 2:

The design of the database tables is as follows:

Table: **tblLocations**

This table contains the details of each location.

Field name	Data type	Description
LocationID (PK)	Number	A unique ID that identifies a location
City	Text (20)	The name of the city where the location is
Province	Text (15)	The name of the province where the city is located
Altitude	Number	The altitude of the location
Population	Number	The population of the city

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

LocationID ▾	City ▾	Province ▾	Altitude ▾	Population ▾
1	Cape Town	Western Cape	42	4618000
2	Durban	KwaZulu-Natal	8	3721000
3	Bloemfontein	Free State	1395	598000
4	Pretoria	Gauteng	1339	2473000
5	Gqeberha	Eastern Cape	85	967677
6	Johannesburg	Gauteng	1753	5635000
7	Rustenburg	North West	1170	574000
8	Polokwane	Limpopo	1310	479000
9	Mbombela	Mpumalanga	1653	110159
10	eMalahleni	Mpumalanga	1520	434238

Table: **tblMarathons**

This table contains information on the marathons available in each location.

Field name	Data type	Description
MarathonID (PK)	Number	A unique ID that identifies the marathon
MarathonName	Text (30)	The name of the marathon
MarathonDate	Date/Time	The date on which the marathon takes place
Distance	Number	The distance of each marathon
PrizeMoney	Currency	The prize money awarded to the winner of the marathon
Organiser	Text (25)	The name of the organiser of the marathon
LocationID (FK)	Number	The ID of the location where the marathon takes place

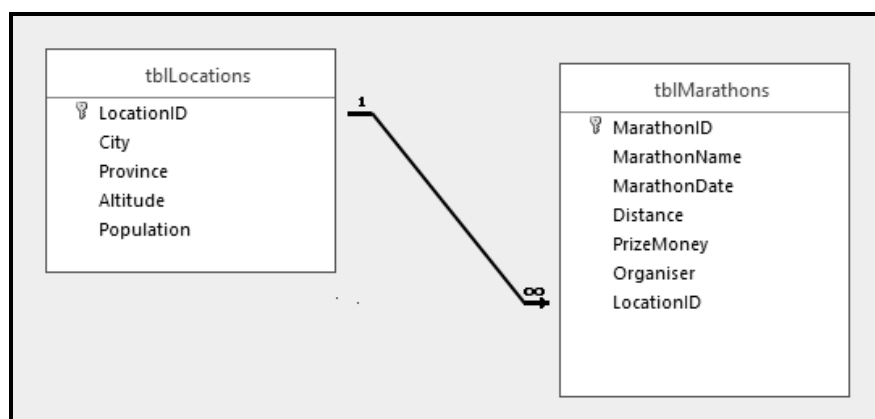
Example of the first ten records of the **tblMarathons** table:

MarathonID	MarathonName	MarathonDate	Distance	PrizeMoney	Organiser	LocationID
1	East London Marathon	2024/09/12	28.4	R50 000.00	Running South Africa	15
2	Joburg Run	2024/10/26	55.2	R12 000.00	Active Events	6
3	Cape Town Challenge	2025/01/18	40.6	R20 000.00	Runners United	1
4	Durban Ultra Marathon	2025/02/19	45.8	R10 000.00	Endurance Sports SA	2
5	Johannesburg Run Fest	2025/03/15	75.9	R13 000.00	Outdoor Adventures	6
6	Pretoria Classic	2025/04/28	32.5	R8 000.00	PE Running Club	4
7	Port Elizabeth Pursuit	2025/05/10	90.5	R5 000.00	Wild Runners SA	5
8	Kruger National Challenge	2025/06/23	58.3	R30 000.00	Mountain Marathoners	9
9	Drakensberg Dash	2025/07/07	37.6	R35 000.00	Winelands Athletics	14
10	Soweto Sprint	2025/08/19	42.1	R2 500.00	City Runners	6

**NOTE:**

- Connection code has been provided.
- The database is password-protected; therefore, you will not be able to access the database directly.

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





## PLANNING PAGE 1





## PLANNING PAGE 2







Examination sticker

150

**INFORMATION TECHNOLOGY P1 – NOVEMBER 2024**  
**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 NSC NOV 2024 Examination:**

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 enter the file name(s) used for each answer. Tick if saved and/or attempted.*

Question number	File name	Saved (✓)	Attempted (✓)	Maximum Mark	Mark Achieved	Marker Initial/ Code
1	Question1_P.dproj			40		
2	Question2_P.dproj			40		
3	MRecord_U.pas			19		
	Question3_P.dproj			21		
4	Question4_P.dproj			30		
<b>TOTAL</b>				<b>150</b>		

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

---



---



# basic education

Department:  
Basic Education  
**REPUBLIC OF SOUTH AFRICA**

## NATIONAL SENIOR CERTIFICATE

Stanmorephysics.com

**GRADE 12**

**INFORMATION TECHNOLOGY P1**

**NOVEMBER 2024**

**MARKING GUIDELINES**

**MARKS: 150**

**These marking guidelines consist of 33 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 15) include the marking grid for each question.
- **Annexures E, F, G and H** (pages 16 to 33) 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 15) 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
1.1.1	<b>Button [1.1.1 - Random]</b>  Generate random number ✓ from 5 to 10 (both values included in range) ✓ edtQ1_1_1.Text := ✓ convert number to String ✓	4	
1.1.2	<b>Button [1.1.2 - Round up]</b>  spnQ1_1_2.Value := ✓ Round up ✓ Ceil (NUMBER) Round (NUMBER + 0.5) Floor (NUMBER) + 1 Trunc (NUMBER) + 1 With constant as a parameter ✓	3	
1.2	<b>Button [1.2 - Surface area]</b>  Extract the height and radius ✓ from the edit boxes Converted to Float ✓ Calculate the Area: A := PI * rR * ✓ (rR + ✓ sqrt (✓ sqr(rR) + sqr(rH) ✓)) Display the Area in the label ✓ formatted to 2 decimal places ✓ FormatFloat ('0.00', A) FloatToStrF (A, ffFixed, 10, 2) Format ('%.2f', [A])  <b>Also ACCEPT</b> Power(rR, 2) instead of Sqr.  <b>NOTE:</b> Brackets must be added correctly in the calculation.	8	

1.3	<p><b>Button [1.3 - Read file]</b></p> <p>Declare variables for the address and bedrooms ✓                      AssignFile(tFile, 'Houses.txt') ✓                      Reset(tFile) ✓</p> <p>Loop through the text file with correct condition ✓                          Readln ✓ (tFile, address variable ✓)                          Readln (tFile, bedroom variable) ✓                          Concatenate the address and bedroom with a dash ( - )                              between the address and bedroom ✓                          Display the output in the rich edit ✓</p> <p><b>Also ACCEPT</b> alternative to read from file:                      If odd line number, then store in address variable (2)                      If even line number, then store in bedroom variable (1)</p>	9	
1.4	<p><b>Button [1.4 - Add name]</b></p> <p>Extract the name from the combo box ✓</p> <p>Test if ✓                          the check box is checked ✓                          Add name to the paid rich edit component ✓</p> <p>Else ✓                          Add name to the not paid rich edit component ✓</p> <p>Remove the selected name from the combo box ✓                          cmbQ1_4.DeleteSelected                          cmbQ1_4.Items.Delete (cmbQ1_4.ItemIndex)</p> <p><b>NOTE:</b>                      NO mark for changing the content of the combo box item to be an empty string and not removing it.</p>	7	

1.5	<p><b>Button [1.5 - Replace]</b></p> <p>Loop through the string ✓                  Test if character is NOT a space ✓                  Add character to password variable in reverse ✓</p> <p>Display password in memo ✓</p> <p>Loop Index from 1 to length of password ✓                  If Index MOD 3 is 0 ✓                  Generate random value from 1 to 6 ✓                  Replace password character at Index with random character from sCharacters ✓</p> <p>Display updated password in memo ✓</p> <p><b>Alternative for the first 3 marks:</b>                  Loop to remove spaces first (1), then loop (1) through the modified string in reverse (1)</p> <p><b>In the password part, also ACCEPT:</b></p> <ul style="list-style-type: none"> <li>• Loop Index from 3 to length of password</li> <li>• Randomly generate value from 1 to Length(sCharacters)</li> </ul>	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	<b>SQL statements</b>		
2.1.1	<b>Button [2.1.1 - Low population]</b> <pre>SELECT * ✓ FROM tblLocations ✓ WHERE Population &lt; 200000 ✓</pre>	3	
2.1.2	<b>Button [2.1.2 - Runners United September runs]</b> <pre>SELECT MarathonID, MarathonDate, Distance ✓ FROM tblMarathons ✓ WHERE Organiser = "Runners United" ✓ AND ✓ Month(MarathonDate) = 9 ✓</pre> <p><b>Also ACCEPT:</b></p> <ul style="list-style-type: none"> <li>• Organiser Like "%Runners United%"</li> <li>• MarathonDate Like "%/09/%"</li> <li>• Mid (Marathondate, 6, 2) = 9</li> </ul>	4	
2.1.3	<b>Button [2.1.3 - Marathon locations]</b> <pre>SELECT City &amp; " - " ✓ &amp; LEFT(Province, 3) ✓ AS Location ✓ FROM tblLocations ✓</pre> <p><b>Also ACCEPT:</b></p> <ul style="list-style-type: none"> <li>• + instead of &amp;</li> <li>• Mid(Province, 1, 3)</li> </ul>	4	
2.1.4	<b>Button [2.1.4 - Add city]</b> <pre>INSERT INTO tblLocations ✓ VALUES ✓ (19, "Welkom", "Free State", 1198, 423016) ✓✓ (correct order (1 mark), correct number of parameters (1 mark))</pre>	4	



2.1.5	<b>Button [2.1.5 - City details]</b>  SELECT City, ✓ COUNT(City) ✓ AS NumMarathons, SUM(Prizemoney) AS [Total Prize Money] ✓ FROM tblMarathons , tblLocations ✓ WHERE tblMarathons.LocationID = tblLocations.LocationID ✓ GROUP BY City ✓ HAVING SUM(Prizemoney) > 50000 ✓  <b>Also ACCEPT:</b> <ul style="list-style-type: none"> <li>Count(*)</li> <li>Count(A field name) / Count(tblLocations.LocationID)</li> </ul>	7	
	<div>Subtotal:</div>		

**QUESTION 2: MARKING GRID (CONT.)**

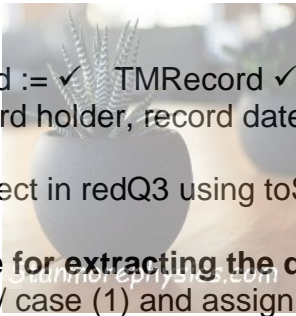
2.2	<b>Database Manipulation</b>		
2.2.1	<b>Button [2.2.1 - Remove marathons]</b>  Go to the first record in tblMarathons ✓ Loop through tblMarathons ✓ Test if tblMarathons['Organiser'] = sOrganiser ✓ tblMarathons.Delete ✓ else tblMarathons.Next ✓ End loop	5	
2.2.2	<b>Button [2.2.2 - Qualifying events]</b>  Initialise flag / counter ✓ tblLocations.First (mark with tblLocations.Next)** ✓ Loop through tblLocations ✓ Test if tblLocations['City'] = sCity ✓ Change flag / increment counter ✓ tblMarathons.First (mark with tblMarathons.Next)** ✓ Loop through tblMarathons ✓ Test if (tblMarathons['LocationID'] = tblLocations['LocationID']) ✓ AND ✓ (tblMarathons['Distance'] >= 40) ✓ Display the MarathonName and Distance converted to string ✓ tblMarathons.Next End loop (tblMarathons) tblLocations.Next End loop (tblLocations)  Test flag / counter ✓ Display message indicating that the city is not found ✓  <b>NOTE: **</b> <ul style="list-style-type: none"> <li>• The FIRST and NEXT statements for the outer loop (for tblLocations) must both be in the correct position for one mark.</li> <li>• The FIRST and NEXT statements for the inner loop (for tblMarathons) must both be in the correct position for one mark.</li> </ul>	13	
	<b>Subtotal:</b>	<b>18</b>	
	<b>TOTAL SECTION B:</b>	<b>40</b>	

## ANNEXURE C

### QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.1	<b>Constructor Create</b>  Set the attributes fMarathonName and fRecordHolder to correct parameters ✓ Set fRecordDate and fRecordTime to correct parameters ✓ Set fDistance to correct parameter ✓	3	
3.1.2	<b>Function getRecordTime</b>  Function heading with String value as return data type ✓ Return fRecordTime ✓	2	
3.1.3	<b>Function checkRecord</b>  Function with Boolean return datatype ✓ with string parameter ✓ Test if parameter value < fRecordTime ✓ Return true ✓ Else Return false ✓  <b>Also ACCEPT:</b> <ul style="list-style-type: none"> <li>• StrToTime and toMinutes to compare the times</li> <li>• Test if fRecordTime &gt; parameter value</li> </ul>	5	
3.1.4	<b>Function calcPace</b>  Function heading with real return data type ✓ Return ✓ toMinutes(fRecordTime) ✓ / fDistance ✓	4	
3.1.5	<b>Function toString</b>  Function heading with string value as return data type ✓ Build a string with labels: dash, 'km:', 'on' and brackets ✓ Distance converted to string ✓ Contains all attributes ✓ Return the string ✓	5	
	<b>Subtotal: Object class</b>	<b>19</b>	


**QUESTION 3: MARKING GRID (CONTINUED)**

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	<b>Button [3.2.1 - Instantiate Object]</b>  Distance extracted from radio group: Items at ItemIndex ✓  Extract the distance using String manipulation ✓ converted to real ✓   objMRecord := ✓ TMRecord ✓ .create ✓ (marathon name, record holder, record date, record time, distance) ✓  Display object in redQ3 using toString method ✓  <b>Alternative for extracting the distance:</b> Use if else / case (1) and assign distance (1)	8	
3.2.2	<b>Button [3.2.2 - Pace]</b>  Call the calcPace method ✓ Display the result in redQ3 converted to a string ✓ Formatted to 3 decimal places ✓ with correct label (min/km) ✓	4	
3.2.3	<b>Button [3.2.3 - Check record]</b>  Extract name and time of runner from edit boxes ✓  If objMRecord.checkRecord ✓ (Time of runner) ✓ Call setRecordHolder (Name of runner) ✓ Call setRecordTime (Time of runner) ✓ Call setRecordDate ✓ (Current date as string) ✓ Display toString method ✓ Else Display current record time in redQ3 by calling the getRecordTime method ✓	9	
	<b>Subtotal Form class:</b>	<b>21</b>	
	<b>TOTAL SECTION C:</b>	<b>40</b>	


## ANNEXURE D

### QUESTION 4: MARKING GRID – PROBLEM SOLVING

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX MARKS	LEARNER'S MARKS
4.1	<p><b>Button [4.1 - Count marathons]</b></p> <p>Outer loop from 1 to 9 ✓                      Initialise counter ✓                      Inner loop (out + 1 to 10) ✓                          Test array[out] ✓ = array[inner] ✓                          and array[inner] &lt;&gt; symbol ✓                          increment counter ✓                          replace array[inner] with a symbol                          such as ('*', ' ') ✓                      if array[out] is NOT symbol used ✓                          Display array[out] ✓ and counter converted to                          string ✓</p> <p><b>Concepts 1: (Replace duplicates when counting)</b>                      Loop outer through array (1)                          Initialise counter (1)                              Loop inner starting at outer loop counter + 1 (1)                              Test array[outer] (1) = array[inner] (1)                                  AND array[inner] &lt;&gt; Symbol used (1)                              Increase counter (1)                              Replace array[inner] with space or * or other                              symbol (1)                      if array[outer] is NOT symbol used (1)                          Display array[outer] (1) and counter converted to                          string (1)</p>	11	

	<p><b>Concepts 2: (Create list (array/string) without duplicates)</b></p> <p><b>Build string / populate array with unique values [4]</b>                  Initialisation of counter and bFound (1)                  Outer and inner loops (1)                  If statements used in testing (1)                  Assignment statements (1)</p> <p><b>Loop through unique values in Temp array and count from arrMarathons array [5]</b>                  Loop x through temporary array with unique values / string (1)                  Initialise iNumMarathons (1)                  Loop y through arrMarathons (1)                  Test if arrTemp[x] equal to arrMarathons[y] (1)                  Increment iNumMarathons (1)</p> <p><b>Display [2]</b>                  Display marathon name (1) and counter (1)</p> <p><b>Concepts 3: (Temp arrays Unique marathon names and counting values)</b>                  Use arrTempMarathons and arrCountMarathons                  Initialise counter ** (1) with found := false                  Loop outer through array (1)                  Set bFound to FALSE **                  Loop inside starting at 1 to counter (1)                  Test arrMarathons[outer] =                      arrTempMarathons[inside] (1)                  Change bFound to TRUE (1)                  Increase arrCountMarathons[inside] (1)                  If bFound is FALSE (1)                      Increase counter (1)                      Set arrCountMarathons[counter] to 1 (1) ##                      Set arrTempMarathons[counter] to                          arrMarathons[outside] ##                  Loop from 1 to counter (1)                  Display arrTempMarathons and arrCountMarathons (converted to integer) (1)</p>		
---	---	--	--

2  
C  
E  
R  
T  
I  
F  
I  
C  
A  
T  
E  
D  
P  
R  
O  
F  
E  
S  
S  
O  
R

	<p><b>Concept 1: (Row and Column loops – copy from row)</b></p> <p><i>Extract word from combo box [1]</i>                  Read word from combo box (1)</p> <p><i>Test if word is in the row [10]</i>                  Loop iR from 1 to 14 (1)                  Loop iC from 1 to 14 (1)                  Test if marathon name is in row iR                      starting at column index iC (4)                      Calculate/Set start column index of word (1)                      Calculate end column index of word (3)</p> <p><i>Display the row number, start and end cloumn [3]</i>                  Display row number of word (1)                      Start column (1)                      and End column (1)</p> <p><i>Change to uppercase [4]</i>                  Loop from start column (1) to end column (1)                      Change character (1) to upcase character (1)</p> <p><i>Display 2D array [1]</i>                  Call the display method (1)</p> <p><b>Concept 2: (Row – using pos directly)</b></p> <p><i>Extract word from combo box [1]</i>                  Read word from combo box (1)</p> <p><i>Test if word is in the row [10]</i>                  Row Loop from 1 to 14 (1)                  Test if marathon name is in the row (4)                      Calculate start column index of word (2)                      Calculate end column index of word (3)</p> <p><i>Display the row number, start and end column [3]</i>                  Display row number of word (1)                      Start column (1)                      End column (1)</p> <p><i>Change to uppercase [4]</i>                  Loop from start column (1) to end column (1)                      Change character (1) to upcase character (1)</p> <p><i>Display 2D array [1]</i>                  Call the display method (1)</p>		
	<p><b>TOTAL SECTION D:</b></p>	<p><b>30</b></p>	



**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, pngimage, ExtCtrls, Spin, math,
    jpeg;

type
    TfrmQuestion1 = class(TForm)
        grpQ1_1: TGroupBox;
        grpQ1_2: TGroupBox;
        grpQ1_3: TGroupBox;
        grpQ1_4: TGroupBox;
        grpQ1_5: TGroupBox;
        btnQ1_1_1: TButton;
        btnQ1_1_2: TButton;
        edtQ1_1_1: TEdit;
        spnQ1_1_2: TSpinEdit;
        Label1: TLabel;
        Label2: TLabel;
        edtQ1_2_r: TEdit;
        edtQ1_2_h: TEdit;
        btnQ1_2: TButton;
        btnQ1_3: TButton;
        redQ1_3: TRichEdit;
        Label3: TLabel;
        chbQ1_4: TCheckBox;
        btnQ1_4: TButton;
        Label4: TLabel;
        Label5: TLabel;
        redQ1_4_P: TRichEdit;
        redQ1_4_NP: TRichEdit;
        btnQ1_5: TButton;
        cmbQ1_4: TComboBox;
        edtQ1_5: TEdit;
        lblQ1_2: TLabel;
        Image1: TImage;
        memQ1_5: TMemo;
        procedure btnQ1_1_1Click(Sender: TObject);
        procedure btnQ1_1_2Click(Sender: TObject);
        procedure btnQ1_2Click(Sender: TObject);
        procedure btnQ1_3Click(Sender: TObject);
        procedure btnQ1_4Click(Sender: TObject);
        procedure btnQ1_5Click(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmQuestion1: TfrmQuestion1;
implementation
```

```
{ $R *.dfm}

// =====
// 1.1.1 Random 4 marks
// =====
procedure TfrmQuestion1.btnQ1_1_1Click(Sender: TObject);
begin
    edtQ1_1_1.Text := intToStr(randomRange(5, 11));
end;

// =====
// 1.1.2 Random 3 marks
// =====

procedure TfrmQuestion1.btnQ1_1_2Click(Sender: TObject);
const
    NUMBER = 5.63247;
begin
    spnQ1_1_2.Value := ceil(NUMBER);
end;

// =====
// 1.2 Surface area 8 marks
// =====
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
    rA, rH, rR: real;

begin
    rH := StrToFloat(edtQ1_2_h.Text);
    rR := StrToFloat(edtQ1_2_r.Text);
    rA := PI * rR * (rR + Sqrt(Sqr(rH) + Sqr(rR)));
    lblQ1_2.Caption := FloatToStrF(rA, ffFixed, 8, 2);
end;

// =====
// 1.3 Read file 9 marks
// =====
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
    tFile: textfile;
    sAdd, sRooms: String;
begin
    AssignFile(tFile, 'Houses.txt');
    Reset(tFile);

    while NOT Eof(tFile) do
    begin
        readln(tFile, sAdd);
        readln(tFile, sRooms);
        redQ1_3.Lines.Add(sAdd + ' - ' + sRooms);
    end;
    CloseFile(tFile);
end;
```

```
// =====
// 1.4 Add name 7 marks
// =====
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
    sName: String;
begin
    sName := cmbQ1_4.Text;
    if chbQ1_4.Checked then
    begin
        redQ1_4_P.Lines.Add(sName);
    end
    else
    begin
        redQ1_4_NP.Lines.Add(sName);
    end;

    cmbQ1_4.items.Delete(cmbQ1_4.ItemIndex);
end;
// =====
// 1.5 Replace 9 marks
// =====
procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
    sNameSurname, sCharacters, sPassword: String;
    iCnt, iLen, iRandom: integer;
begin
    // Provided code
    sNameSurname := edtQ1_5.Text;
    sCharacters := '@#$$%^&';
    // Add your code here
    sPassword := '';
    iLen := Length(sNameSurname);
    for iCnt := iLen downto 1 do
    begin
        if sNameSurname[iCnt] <> ' ' then
            sPassword := sPassword + sNameSurname[iCnt];
    end;
    memQ1_5.Lines.Add(sPassword);
    for iCnt := 1 to Length(sPassword) do
    begin
        if(iCnt mod 3 = 0) then
        begin
            iRandom := random(6) + 1;
            sPassword[iCnt] := sCharacters[iRandom];
        end;
    end;
    memQ1_5.Lines.Add(sPassword);
end;
end.
```

## ANNEXURE F: SOLUTION FOR QUESTION 2

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

```
// =====
// 2.1.1 Low population 3 marks
// =====
```

```
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
    sSQL1: String;
begin
    // Question 2.1.1

    sSQL1 := 'SELECT * ' + 'FROM tblLocations ' + 'WHERE Population <
    200000';

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

```
// =====
// 2.1.2 Runners United September runs 4 marks
// =====
```

```
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
    sSQL2: String;
begin
    // Question 2.1.2
    sSQL2 := 'SELECT MarathonID, MarathonDate, Distance ' +
        'FROM tblMarathons ' + 'WHERE Organiser = "Runners United" AND ' +
        'Month(MarathonDate) = 9';

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

```
// =====
// 2.1.3 Marathon locations 4 marks
// =====
```

```
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
    sSQL3: String;
begin
    // Question 2.1.3

    sSQL3 := 'SELECT City & " - " & left(Province,3) AS [Location] ' +
        'FROM tblLocations ';

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

```
// =====  
// 2.1.4 Add city 4 marks  
// =====
```

```
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);  
var  
    sSQL4: String;  
    bValid: boolean;  
begin  
    // Question 2.1.4  
  
    sSQL4 := 'INSERT INTO tblLocations VALUES  
              (19,"Welkom","Free State",1198,423016)';  
  
    // Provided code - do not change  
    dbCONN.ExecuteSQL(sSQL4);  
  
end;
```

```
// =====  
// 2.1.5 High prize money 7 marks  
// =====
```

```
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);  
var  
    sSQL5: String;  
    bChanged: boolean;  
begin  
    // Question 2.1.5  
  
    sSQL5 :=  
        'SELECT City, COUNT(City) AS [NumMarathons],  
          SUM(Prizemoney) AS [Total Prize Money] '  
        + 'FROM tblMarathons , tblLocations '  
        + 'WHERE tblMarathons.LocationID = tblLocations.LocationID ' +  
        'GROUP BY City HAVING SUM(Prizemoney) > 50000';  
  
    // Provided code - do not change  
    dbCONN.runSQL(sSQL5);  
end;
```

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

// =====
// 2.2.1 Remove marathons                                     5 marks
// =====

procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
    sOrganiser: String;
begin
    // Provided code
    sOrganiser := InputBox('Organiser',
        'Enter the name of the organiser to remove', 'Endurance Sports SA');

    // Question 2.2.1
    tblMarathons.First;
    while NOT tblMarathons.Eof do
    begin
        if tblMarathons['Organiser'] = sOrganiser then
            tblMarathons.Delete
        else
            tblMarathons.Next;
        end;
    end;
end;

// =====
// 2.2.2 Qualifying events                                     13 marks
// =====

procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
    sCity: String;
    bFound: boolean;
    iLocation: Integer;
begin
    // Provided code
    sCity := InputBox('City', 'Enter the name of the city', 'Paarl');
    redQ2_2_2.Clear();
    // Question 2.2.2
    bFound := False;
    tblLocations.First;
    while (NOT tblLocations.Eof) AND (bFound = False) do
    begin
        if tblLocations['City'] = sCity then
        begin
            bFound := True;
            iLocation := tblLocations['LocationID'];
        end;
        tblLocations.Next;
    end;

    if bFound then
    begin
        tblMarathons.First;
        while NOT tblMarathons.Eof do
```

```

begin
    if (tblMarathons['LocationID'] = iLocation) AND
        (tblMarathons['Distance'] >= 40) then
        begin
            redQ2_2_2.Lines.Add(tblMarathons['MarathonName'] + #9 +
                FloatToStr(tblMarathons['Distance']));
        end;
        tblMarathons.Next;
    end
end
else
    redQ2_2_2.Lines.Add(sCity + ' is not found.');
```

// Alternative:

```

{ bFound := False;
  tblLocations.First;
  while NOT tblLocations.Eof do
  begin
      if tblLocations['City'] = sCity then
      begin
          bFound := True;
          tblMarathons.First;
          while NOT tblMarathons.Eof do
          begin
              if (tblMarathons['LocationID'] = tblLocations['LocationID'])
                  AND (tblMarathons['Distance'] >= 40) then
              begin
                  redQ2_2_3.Lines.Add(tblMarathons['MarathonName'] + #9 +
                      FloatToStr(tblMarathons['Distance']));
              end;
              tblMarathons.Next;
          end;
      end;
      tblLocations.Next;
  end;

  if bFound = False then
      redQ2_2_3.Lines.Add(sCity + ' is not found.')}
end;
```

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

```

procedure TfrmQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
    // Restores the Database
    dbCONN.RestoreDatabase;
    redQ2_2_2.Clear;
    dbCONN.SetupGrids(dbgLocations, dbgMarathons, dbgrdSQL);
end;
```

```

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



```
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
    // Provided code
    redQ2_2_2.Paragraph.TabCount := 2;
    redQ2_2_2.Paragraph.Tab[0] := 150;
    redQ2_2_2.Paragraph.Tab[1] := 300;
end;

procedure TfrmQuestion2.FormShow(Sender: TObject);
begin
    // Sets up the connection to database and opens the tables.
    dbCONN := TConnection.Create;
    dbCONN.dbConnect;
    tblLocations := dbCONN.tblOne;
    tblMarathons := dbCONN.tblMany;
    dbCONN.SetupGrids(dbgLocations, dbgMarathons, dbgGrdSQL);
    pgcDBAdmin.ActivePageIndex := 0;
end;

// =====
// {$ENDREGION}
// =====

end.
```

## ANNEXURE G: SOLUTION FOR QUESTION 3

### Object class

```
unit MRecord_U;

interface

type
  TRecord = class(TObject)
  private
    var
      fMarathonName: String;
      fRecordHolder: String;
      fRecordDate: String;
      fRecordTime: String;
      fDistance: real;
  public
    // Provide code
    constructor create(sMarathonName, sRecordHolder, sRecordDate,
      sRecordTime: String; rDistance: real);
    procedure setRecordHolder(sName: String);
    procedure setRecordTime(sNewRecord: String);
    procedure setRecordDate(sNewDate: String);
    function toMinutes(sTime: String): real;

    function getRecordTime: String;
    function checkRecord(sRecordTime: String): boolean;
    function calcPace: real;
    function toString: String;
  end;

implementation

uses
  SysUtils, Math;

{ TRecord }

// =====
// 3.1.1 Constructor Create                                     3 marks
// =====

constructor TRecord.create(sMarathonName, sRecordHolder, sRecordDate,
  sRecordTime: String; rDistance: real);
begin
  fMarathonName := sMarathonName;
  fRecordHolder := sRecordHolder;
  fDistance := rDistance;
  fRecordDate := sRecordDate;
  fRecordTime := sRecordTime;
end;
```

```
// =====
// 3.1.2 Function getRecordTime                                2 marks
// =====
```

```
function TMRecord.getRecordTime: String;
begin
    Result := fRecordTime;
end;
```

```
// =====
// 3.1.3 Function checkRecord                                5 marks
// =====
```

```
function TMRecord.checkRecord(sRecordTime: String): boolean;
begin
    Result := sRecordTime < fRecordTime;
    { Alternative:
      if sRecordTime < fRecordTime then
        Result := True
      Else
        Result := False;
    }
end;
```

```
// =====
// 3.1.4 Function calcPace                                    4 marks
// =====
```

```
function TMRecord.calcPace: real;
begin
    Result := toMinutes(fRecordTime) / fDistance;
end;
```

```
// =====
// 3.1.5 Function toString                                    5 marks
// =====
```

```
function TMRecord.toString: String;
begin
    Result := fMarathonName + ' - ' + FloatToStr(fDistance)
              + ' km: ' + fRecordHolder + ' (' + fRecordTime + ' on ' +
              fRecordDate + ')';
end;
```

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

```
procedure TMRecord.setRecordHolder(sName: String);
begin
    fRecordHolder := sName;
end;
```

```
procedure TMRecord.setRecordTime(sNewRecord: String);
begin
    fRecordTime := sNewRecord;
end;
```

```
procedure TMRecord.setRecordDate(sNewDate: String);  
begin  
    fRecordDate := sNewDate;  
end;  
function TMRecord.toMinutes(sTime: String): real;  
var  
    rMin: real;  
begin  
    rMin := StrToFloat(copy(sTime, 4, 2));  
    rMin := rMin + StrToFloat(copy(sTime, 1, 2)) * 60;  
    rMin := rMin + StrToFloat(copy(sTime, 7, 2)) / 60;  
    Result := rMin;  
end;  
  
end.
```



## Main Form Unit

```
unit Question3_U;

interface

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

type
    TfrmQuestion3 = class(TForm)
        gbxD3_2_1: TGroupBox;
        gbxD3_2_2: TGroupBox;
        redQ3: TRichEdit;
        btnQ3_2_1: TButton;
        gbxD3_2_3: TGroupBox;
        btnQ3_2_3: TButton;
        Panel1: TPanel;
        Panel2: TPanel;
        btnQ3_2_2: TButton;
        Label6: TLabel;
        edtQ3_2_1_Marathon: TEdit;
        Label2: TLabel;
        Label1: TLabel;
        edtQ3_2_1_RecordHolder: TEdit;
        Label3: TLabel;
        Label4: TLabel;
        Label5: TLabel;
        edtQ3_2_1_RecordTime: TEdit;
        rgpQ3_2_1: TRadioGroup;
        Label7: TLabel;
        edtQ3_2_3_Name: TEdit;
        edtQ3_2_3_Time: TEdit;
        Image1: TImage;
        edtQ3_2_1_RecordDate: TEdit;
        procedure btnQ3_2_1Click(Sender: TObject);
        procedure btnQ3_2_3Click(Sender: TObject);
        procedure btnQ3_2_2Click(Sender: TObject);
    private
    public
    end;
var
    frmQuestion3: TfrmQuestion3;
implementation
{$R *.dfm}
uses
    MRecord_U;

var
    objMRecord: TMRecord;
```

// =====  
**// 3.2.1 Instantiate object** **8 marks**  
 // =====

```
procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
    sMarathonName, sRecordHolder, sRecordDate, sRecordTime: String;
    rDistance: real;
    sDistance: String;
begin
    // Provided code
    redQ3.Clear;
    sMarathonName := edtQ3_2_1_Marathon.Text;
    sRecordHolder := edtQ3_2_1_RecordHolder.Text;
    sRecordDate := edtQ3_2_1_RecordDate.Text;
    sRecordTime := edtQ3_2_1_RecordTime.Text;

    // Question 3.2.1

    sDistance := rgpQ3_2_1.Items[rgpQ3_2_1.ItemIndex];
    rDistance := StrToFloat(Copy(sDistance, 1, Pos(' ', sDistance) - 1));
    objMRecord := TMRecord.create(sMarathonName, sRecordHolder,
        sRecordDate, sRecordTime, rDistance);
    redQ3.lines.Add(objMRecord.toString);
end;
```

// =====  
**// 3.2.2 Pace** **4 marks**  
 // =====

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

    // Question 3.2.2
    redQ3.lines.Add('Record holder''s pace:'+
        FloatToStrF(objMRecord.calcPace, ffFixed, 8, 3) + '
        min/km');
end;
```

// =====  
**// 3.2.3 Check record** **9 marks**  
 // =====

```
procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
var
    sName, sTime: String;
begin
    // Provided code
    redQ3.Clear;

    // Question 3.2.3
    sName := edtQ3_2_3_Name.Text;
    sTime := edtQ3_2_3_Time.Text;
```

```
if objMRecord.checkRecord(sTime) then
begin
    objMRecord.setRecordHolderName(sName);
    objMRecord.setRecordTime(sTime);
    objMRecord.setRecordDate(DateToStr(Date()));
    redQ3.lines.Add(objMRecord.toString);
end
Else
begin
    redQ3.lines.Add
        ('The current record remains: ' + objMRecord.getRecordTime);
end;
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, Buttons, pngimage;

type
    TfrmQ4_1 = class(TForm)
        pgcQ4: TPageControl;
        tshQ4_1: TTabSheet;
        tshQ4_2: TTabSheet;
        redQ4_1: TRichEdit;
        btnQ4_1: TButton;
        pnlQ4_1Heading: TPanel;
        redQ4_2: TRichEdit;
        cmbQ4_2: TComboBox;
        pnlQ4_2Heading: TPanel;
        btnReset: TBitBtn;
        memQ4_2: TMemo;
        imgQ4_1: TImage;
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        procedure btnQ4_1Click(Sender: TObject);
        procedure display2D;
        procedure FormCreate(Sender: TObject);
        procedure cmbQ4_2Change(Sender: TObject);
        procedure btnResetClick(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmQ4_1: TfrmQ4_1;
    iCheck: Integer = 0;
    arrMarathons: array [1 .. 10] of String = (
        'Wally Hayward Marathon',
        'Sasol Marathon',
        'Soweto Marathon',
        'Jacaranda City Marathon',
        'Sasol Marathon',
        'Durban City Marathon',
        'Soweto Marathon',
        'Soweto Marathon',
        'Wally Hayward Marathon',
        'Soweto Marathon'
    );
```



```
arrChar: array [1 .. 14, 1 .. 14] of char =
    (('u', 'x', 'v', 'm', 's', 'a', 's', 'o', 'l', 'f', 'k', 'j', 't', 'r'),
    ('u', 'm', 'g', 'e', 'n', 'i', 'w', 'a', 't', 'e', 'r', 'd', 's', 'e'),
    ('g', 'v', 'o', 'e', 't', 'v', 'a', 'n', 'a', 'f', 'r', 'i', 'k', 'a'),
    ('e', 'p', 'o', 'y', 'i', 'l', 'c', 'k', 'h', 'j', 's', 'd', 'f', 'd'),
    ('n', 'k', 'n', 'y', 's', 'n', 'a', 'f', 'o', 'r', 'e', 's', 't', 'u'),
    ('i', 's', 'y', 'd', 'b', 'c', 'r', 'g', 'h', 'k', 'c', 's', 'a', 'r'),
    ('w', 'a', 'l', 'l', 'y', 'h', 'a', 'y', 'w', 'a', 'r', 'd', 's', 'b'),
    ('a', 's', 'q', 'r', 't', 'n', 'n', 'j', 'h', 'e', 'r', 't', 'h', 'a'),
    ('t', 'o', 'e', 'r', 'y', 'b', 'd', 'r', 'h', 'k', 'l', 'g', 'd', 'n'),
    ('e', 'j', 'a', 'c', 'a', 'r', 'a', 'n', 'd', 'a', 'c', 'i', 't', 'y'),
    ('r', 'y', 'j', 'f', 'g', 'f', 'c', 'f', 'g', 'u', 'h', 'v', 'c', 'i'),
    ('k', 'h', 'h', 'l', 'p', 'h', 'i', 'l', 'l', 'c', 'r', 'e', 's', 't'),
    ('a', 'd', 'e', 'v', 'd', 's', 'o', 'w', 'e', 't', 'o', 'm', 'k', 'y'),
    ('p', 'd', 'u', 'r', 'b', 'a', 'n', 'c', 'i', 't', 'y', 'z', 'c', 'l'));
```

implementation

```
{ $R *.dfm }
```

```
// =====
// 4.1 Count marathons 11 marks
// =====
```

```
procedure TfrmQ4_1.btnQ4_1Click(Sender: TObject);
var
    iOut, iIn, iNumMarathons: Integer;
begin
    for iOut := 1 to 10 do
        begin
            iNumMarathons := 1;
            for iIn := iOut + 1 to 10 do
                begin
                    if (arrMarathons[iOut] = arrMarathons[iIn]) and
                        (arrMarathons[iIn] <> '') then
                        begin
                            inc(iNumMarathons);
                            arrMarathons[iIn] := '';
                        end;
                end;
            end;
            if arrMarathons[iOut] <> '' then
                redQ4_1.Lines.Add(arrMarathons[iOut] + #9 +
                    IntToStr(iNumMarathons));
            end;
        end;
end;
```

// =====  
**// 4.2 Find hidden marathons** **19 marks**  
 // =====

```
procedure TfrmQ4_1.cmbQ4_2Change(Sender: TObject);
var
    iOut, iIn, iLen, iR, iC, iC2, iPos: Integer;
    sWord, sNewWord1, sNewWord2: String;
    bFound: boolean;
begin
    bFound := false;
    sWord := cmbQ4_2.Text;
    iLen := Length(sWord);
    for iOut := 1 to 14 do
    begin
        for iIn := 1 to 14 do
        begin
            iR := iOut;
            iC := iIn;
            if sWord[i] = arrChar[iOut, iIn] then
            begin

                sNewWord2 := '';
                while (iC <= 14) AND (bFound = false) do
                begin
                    sNewWord2 := sNewWord2 + arrChar[iR, iC];

                    if sNewWord2 = sWord then
                    begin
                        bFound := true;
                        memQ4_2.Text := 'Row ' + IntToStr(iR)
                            + ' @ column ' + IntToStr(iIn) + ' to ' + IntToStr
                                (iIn + iLen - 1); // iC - iLen + 1);
                        for iC2 := iC downto iC - iLen + 1 do
                            arrChar[iR, iC2] := upCase(arrChar[iR, iC2]);
                        display2D;
                        end;
                        inc(iC);
                    end;
                end;
            end;
        end;
    end;
end;
//=====
// Provided code - Do not change
//=====
=
procedure TfrmQ4_1.display2D;
var
    iOut, iIn: Integer;
    sOut: string;
begin
    redQ4_2.Clear;
    for iOut := 1 to 14 do
    begin
```

```

        sOut := sOut + #9 + IntToStr(iOut);
    end;
    redQ4_2.Lines.Add('' + #9 + sOut + #13);
    for iOut := 1 to 14 do
    begin
        sOut := #13 + IntToStr(iOut) + #9;
        for iIn := 1 to 14 do
        begin
            if iCheck = 1 then
            Begin
                arrChar[iOut, iIn] := lowercase(arrChar[iOut, iIn] + '')[1];
                sOut := sOut + #9 + arrChar[iOut, iIn];
            End
            else
                sOut := sOut + #9 + arrChar[iOut, iIn];
            end;
            redQ4_2.Lines.Add(sOut);
        end;
        iCheck := 0;
    end;
procedure TfrmQ4_1.FormCreate(Sender: TObject);
var
    iOut, iIn: Integer;
    sOut: String;
begin
    pgcQ4.ActivePageIndex := 0;
    // Q4.2
    redQ4_2.Clear;
    redQ4_2.Paragraph.TabCount := 15;
    redQ4_2.Paragraph.Tab[0] := 20;
    redQ4_2.Paragraph.Tab[1] := 40;
    redQ4_2.Paragraph.Tab[2] := 60;
    redQ4_2.Paragraph.Tab[3] := 80;
    redQ4_2.Paragraph.Tab[4] := 100;
    redQ4_2.Paragraph.Tab[5] := 120;
    redQ4_2.Paragraph.Tab[6] := 140;
    redQ4_2.Paragraph.Tab[7] := 160;
    redQ4_2.Paragraph.Tab[8] := 180;
    redQ4_2.Paragraph.Tab[9] := 200;
    redQ4_2.Paragraph.Tab[10] := 220;
    redQ4_2.Paragraph.Tab[11] := 240;
    redQ4_2.Paragraph.Tab[12] := 260;
    redQ4_2.Paragraph.Tab[13] := 280;
    redQ4_2.Paragraph.Tab[14] := 300;
    display2D;
end;

procedure TfrmQ4_1.btnResetClick(Sender: TObject);
Var
    i: Integer;
begin
    iCheck := 1;
    display2D;
end;

end.

```