



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

**NATIONAL
SENIOR CERTIFICATE**

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2019

MARKS: 150

TIME: 3 hours

This question paper consists of 21 pages and 2 data pages.

INSTRUCTIONS AND INFORMATION

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

13. The files that you need to complete this question paper have been given 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.

NOTE: Candidates must use the file **DataENGNov2019.exe**.

Do the following:

- Double click on the password-protected executable file:
DataENGNov2019.exe.
- Click on the 'Extract' button.
- Enter the following password: **SPoRt@ScHOOl%**

Once extracted, the following list of files will be available in the folder **DataENGNov2019**:

SUPPLIED FILES

Question 1:

Question1_P.dpr
 Question1_P.dproj
 Question1_P.res
 Question1_U.dfm
 Question1_U.pas

Question 2:

ConnectDB_U.dcu
 ConnectDB_U.pas
 HockeyDB.mdb
 Question2_P.dpr
 Question2_P.dproj
 Question2_P.res
 Question2_U.dfm
 Question2_U.pas

Question 3:

Player_U.pas
 Question3_P.dpr
 Question3_P.dproj
 Question3_P.res
 Question3_U.dfm
 Question3_U.pas

Question 4:

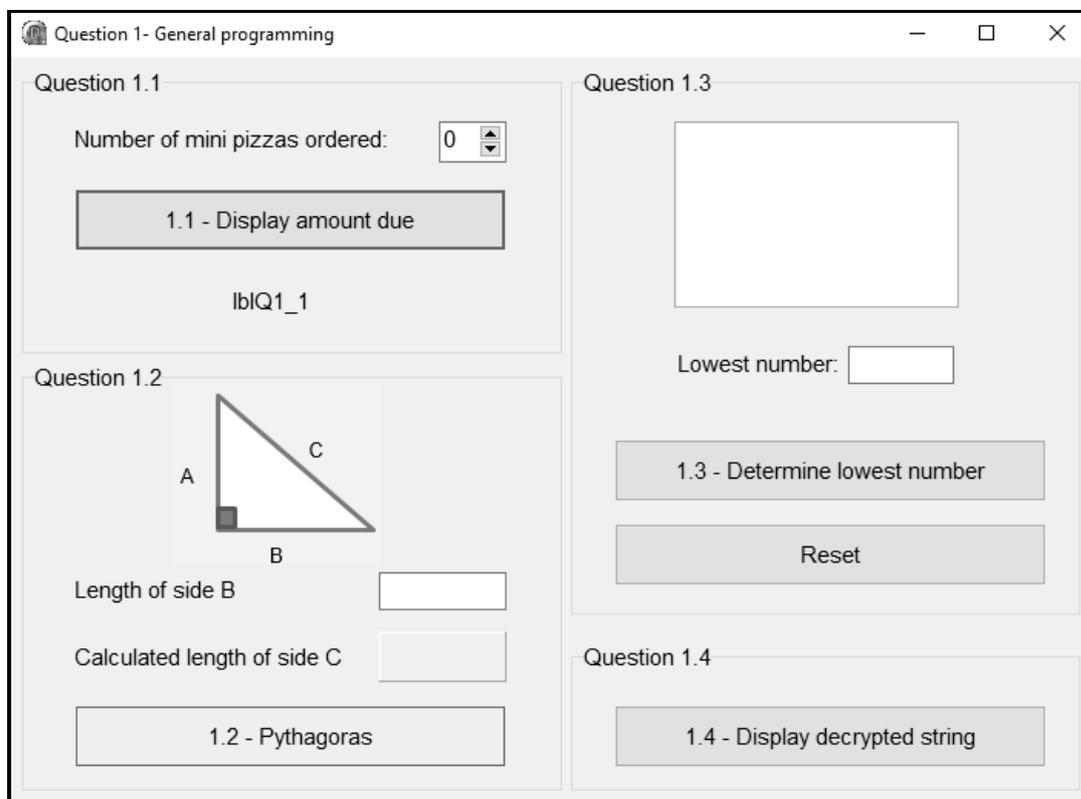
Maze 1.txt
 Maze 2.txt
 Maze 3.txt
 Question4_P.dpr
 Question4_P.dproj
 Question4_P.res
 Question4_U.dfm
 Question4_U.pas

SECTION A**QUESTION 1: GENERAL PROGRAMMING SKILLS**

Do the following:

- Open the incomplete project file called **Question1_P.dpr** 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 user interface displays FOUR different sections named Question 1.1 to Question 1.4. The program has no functionality currently.

Example of the graphical user interface (GUI):



- Complete the code for EACH section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.4 that follow.

1.1 **Button [1.1 - Display amount due]**

Mini pizzas can be ordered from the tuck shop at a school at R14,95 each.

Write code to do the following:

- Declare a constant variable, **PRICE**, to contain the value 14.95.
- Declare TWO variables to store the number of mini pizzas ordered and the amount due respectively.
- Change the font size of label **lblQ1_1** to 20 pt.
- Retrieve the number of mini pizzas ordered from the **spnQ1_1** spin edit box.
- Calculate the amount due using the constant variable **PRICE** and the variable for the number of mini pizzas ordered.
- Display the amount due on the label **lblQ1_1**, formatted as currency.

Example of output if the number of mini pizzas ordered is 5:



(8)

1.2 **Button [1.2 - Pythagoras]**

In Mathematics, the Pythagorean theorem is a fundamental relation in Euclidean geometry with regard to the three sides of a right-angled triangle.

The formula $C^2 = A^2 + B^2$ is used to determine the length of the side opposite the right angle in the triangle below.

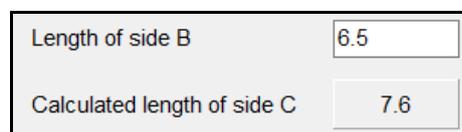
$$C^2 = A^2 + B^2$$

$$C = \sqrt{A^2 + B^2}$$

Write code to do the following:

- Declare variables for sides A, B and C.
- Assign the value of 4 to the variable for side A.
- Extract the length of side B from the **edtQ1_2** edit box and assign it to the variable for side B.
- Calculate the length of side C using the formula $C = \sqrt{A^2 + B^2}$.
- Display the length of side C on the **pnlQ1_2** component as a real value formatted to ONE decimal place.

Example of output if 6.5 was entered as the length of side B:



(10)

1.3 Button [1.3 - Determine lowest number]

A global variable, **iLowest**, is declared and initialised to the value 100.

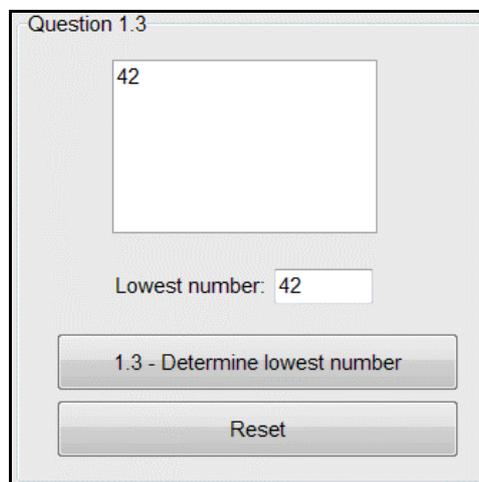
Write code to do the following when the **Determine lowest number** button is clicked:

- Generate and assign a random number in the range from 1 to 100 (inclusive) to a local variable **iNumber** provided.
- Display the number that was generated in the **redQ1_3** rich edit.
- Replace the current lowest number stored in the **iLowest** variable with the generated number if the generated number is lower than the number stored in the **iLowest** variable.
- Display the lowest number in the **edtQ1_3** edit box.

NOTE:

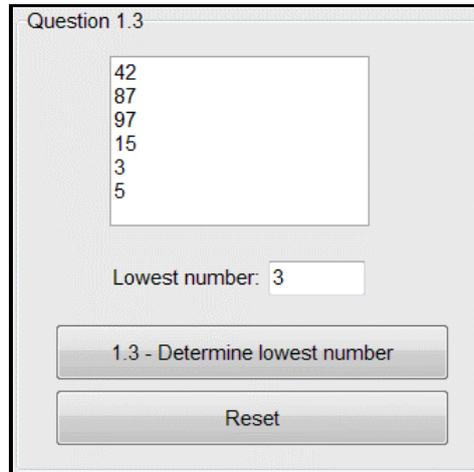
- The random number must be added to the rich edit with each click of the button.
- Code is provided in the **Reset** button to set the lowest number to the starting value of 100 and clear the content of the input and output components.

Example of output when the button is clicked once and the random number 42 is generated:



The screenshot shows a window titled "Question 1.3". Inside the window, there is a rich text editor (redQ1_3) containing the number "42". Below the editor is a text input field (edtQ1_3) labeled "Lowest number:" which also contains the number "42". At the bottom of the window, there are two buttons: "1.3 - Determine lowest number" and "Reset".

Example of output when the button is clicked six times:



NOTE: The output displayed by your program may differ from the example output, as random numbers are generated.

(9)

1.4 **Button [1.4 - Display decrypted string]**

An encrypted string entered by the user must be decrypted and displayed. The encrypted string contains digits, as some of the alphabetical characters in the string have been replaced with digits according to the table below.

ALPHABETICAL CHARACTER	REPLACEMENT DIGIT
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9

Example:

Encrypted version of the string: T8M4!
 Original string (decrypted): TIME!

Write code to do the following to decrypt an encrypted string:

- Use an input box to enter an encrypted string.
- Replace each digit contained in the encrypted string with the alphabetical character it represents (see table above).
- Display the decrypted string in a message box.

Example of input if T8M4! was entered as an encrypted string:



Output for the encrypted string T8M4!:



Test your code using the following test data:

90R represents JAR

03V4NTUR4 T8M4! represents ADVENTURE TIME!

039024NT represents ADJACENT

(13)

- Ensure that your examination number has been entered 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 hockey organiser at your school requires your assistance with the administration of the players, teams and coaches.

The database **HockeyDB** contains two tables called **tblPlayers** and **tblTeams**.

The data pages attached at the end of this question paper provide information on the design of the database and the content of the tables.

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 content of the tables is displayed as shown below on the selection of **TabSheet Question 2.2 - Delphi code**.

PlayerID	PlayerSurname	PlayerName	IDNumber	TeamName	SkillsLevel	GoalKeeper
1	Fivaz	Peter	0306170790504	u/16 A	6	False
2	Maphaphu	Bukelwa	0609220225852	u/14 B	3	False
3	Snyders	Con	0602200518279	u/14 B	3	False
4	Jagers	Ben	0403060227644	u/16 B	4	True

TeamName	Coach	NumberOfGamesPlayed	NumberOfGamesWon
u/14 A	Modikaze, P	8	4
u/14 B	Smith, GP	6	3
u/16 A	Smit, J	8	7
u/16 B	Mpofu, XB	6	3

2.2.1 - Junior players in u/18 A team

Number of junior players in u/18 A team:

2.2.1 - Junior players in u/18 A team

2.2.2 - Coach and goalkeeper information

2.2.2 - Coach and goalkeeper information

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

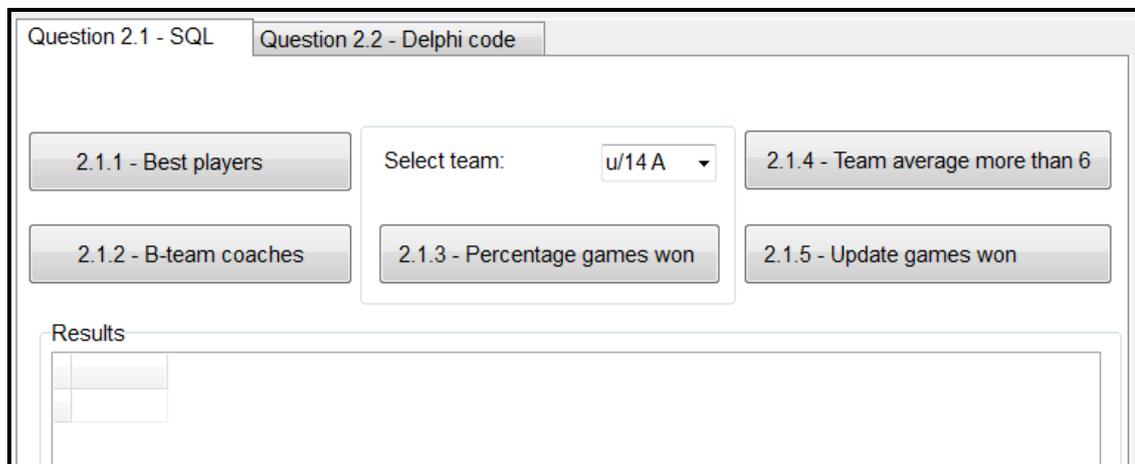
NOTE:

- 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, in other words 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 global variables, as described in the table below.

Variable	Data type	Description
tblTeams	TADOTable	Refers to the table tblTeams in the database HockeyDB
tblPlayers	TADOTable	Refers to the table tblPlayers in the database HockeyDB

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

Example of the GUI for QUESTION 2.1:



NOTE:

- Use ONLY SQL statements to answer QUESTION 2.1.1 to QUESTION 2.1.5.
- Code is provided to execute the SQL statements and display the results of the queries. 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 that follow.

2.1.1 Button [2.1.1 - Best players]

Display the surnames and names of all players with a skills level of 10.

Example of output of the first four records:

PlayerSurname	PlayerName
Mjikelo	Karel
Goliath	Siphokazi
Ncamiso	Nozipo
Baxter	Nomhle

(3)

2.1.2 Button [2.1.2 - B-team coaches]

Display the names of the coaches and teams of all the B-teams.

Example of output:

Coach	TeamName
Smith, GP	u/14 B
Mpofu, XB	u/16 B
Mullan, NV	u/18 B

(4)

2.1.3 Button [2.1.3 - Percentage games won]

Code has been provided to extract a team name from the combo box, **cmbQ2_1_3**.

Display the name of the team, their coach and the **percentage** of games won by the team. Save the percentage of games won in a calculated field called **PercentageGamesWon**.

Example of output if the u/14 B team was selected:

TeamName	Coach	PercentageGamesWon
u/14 B	Smith, GP	50

NOTE: You do NOT have to format the calculated value.

(4)

2.1.4 Button [2.1.4 - Team average more than 6]

The average skills levels of teams are used to identify teams with the highest possibility of winning their games.

Display the names and average skills levels of all teams with an average skills level of more than 6. The average skills level per team must be saved in a calculated field called **AverageSkillsLevel**, formatted to ONE decimal place.

Example of output:

TeamName	AverageSkillsLevel
u/14 A	8.2
u/16 A	7.7
u/18 A	8.4

(5)

2.1.5 Button [2.1.5 - Update games won]

The results of the games won during the last sports day must be used to update the **NumberOfGamesWon** field. Only the u/14 B team lost their game.

Update the data in the **tblTeams** table by adding a value of 1 to the **NumberOfGamesWon** field for the teams who won their game.

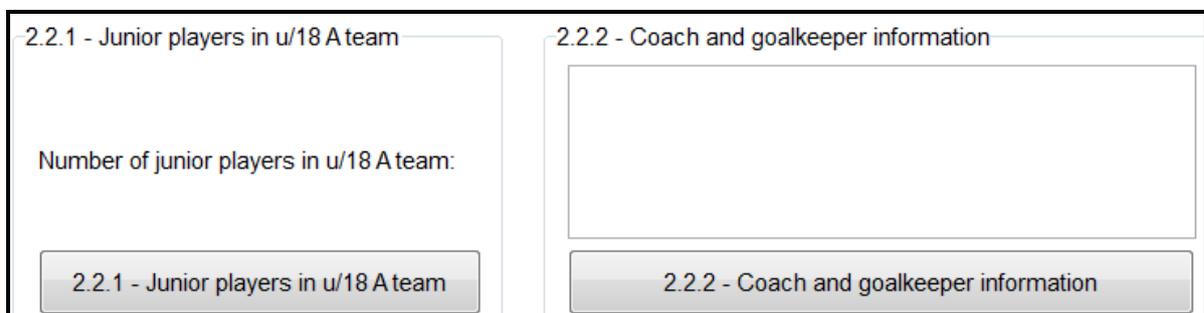
Example of output:



(3)

2.2 Tab sheet [Question 2.2 - Delphi code]

Example of GUI for QUESTION 2.2:



NOTE:

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

2.2.1 **Button [2.2.1 - Junior players in u/18 A team]**

The u/18 A team includes some of the junior players that are exceptionally talented. Junior players are players that were born after the year 2002.

NOTE: The first two digits of the **IDNumber** field indicate the year of birth of a player.

Write code to do the following:

- Save the surnames and names of all the junior players who are members of the u/18 A team to a new text file called **Junior18A.txt**.
- Determine the total number of junior players in the u/18 A team and display the result in the **lblQ2_2_1** label.

Example of content of the **Junior18A** text file:

Cannon Julian Goliath Siphokazi

Example of output to be displayed on the **lblQ2_2_1** label:

Number of junior players in u/18 A team: 2
--

(11)

2.2.2 **Button [2.2.2 - Coach and goalkeeper information]**

The coach and goalkeeper of all the teams are invited to a special training session.

Code has been provided to set the various column widths and to display the headings, as shown in the example of output.

Write code to display a list containing the following information on each team:

- Name of the team
- Surname and initials of the coach
- Surname and name of the goalkeeper in the format:

<Surname>, <Name>

Example of output of the first five records:

TeamName	Coach	Goalkeeper
u/14 A	Modikaze, P	Nel, Koos
u/14 B	Smith, GP	Scheepers, Kurtley
u/16 A	Smit, J	Phillips, Moses
u/16 B	Mpofu, XB	Jagers, Ben
u/18 A	Decan, H	David, Ivan

(10)

- | |
|---|
| <ul style="list-style-type: none"> • Ensure that your examination number has been entered as a comment in the first line of the program file. • Save your program. • Print the code if required. |
|---|

TOTAL SECTION B: 40

SECTION C

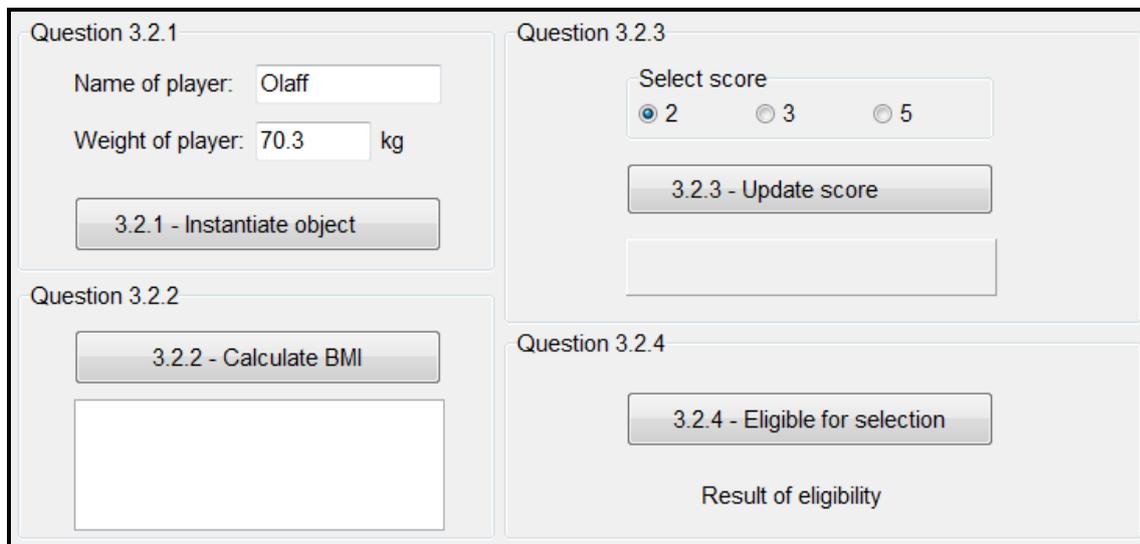
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

The school is designing software to calculate the body mass index (BMI) of the rugby players and to determine the eligibility of the rugby players for selection.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Open the incomplete object class **Player_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question3_U.pas** file and the **Player_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

Example of the GUI:



- Complete the code as specified in QUESTION 3.1 for the **Player_U** object class and QUESTION 3.2 for the **Question3_U** form class.

3.1 The incomplete object class (**TPlayer**) provided contains the declarations of three attributes that define a **Player** object.

The attributes for the **Player** object have been declared as follows:

Names of attributes	Description
fPlayerName	The first name of the rugby player
fWeightOfPlayer	The weight of the player
fScore	The score achieved at a specific rugby game

3.1.1 Write code for a **constructor** method that will receive the player's name and weight as parameter values. Assign these values to the respective attributes. Set the score attribute to the value of zero. (4)

3.1.2 Write code for an accessor method called **getScore** for the **fScore** attribute. (2)

3.1.3 Write code for a method called **updateScore** that will receive an integer value as a parameter and add the received value to the **fScore** attribute. (3)

3.1.4 Write code for a method called **calculateBMI** that must receive the height of the player as a parameter and calculate and return the player's BMI based on the following formula:

$$\text{BMI} = \text{weight of player} / (\text{height of player})^2 \quad (3)$$

3.1.5 Write code for a method called **eligibleForSelection** that can be used to determine the possibility for selection to play at the Provincial Trials Tournament. Possible selection is determined by evaluating the content of **fScore** attribute according to the following categories:

Score	Message
0 to 7 points	Low possibility
8 to 14 points	Medium possibility
More than 14 points	High possibility

The method must return the relevant message. (4)

3.1.6 Write code for a **toString** method to display the attributes of the player object in the following format:

```
Name: <fPlayerName>
Weight: <fWeightOfPlayer>
Current score: <fScore>
```

Example:

```
Name: Olaff
Weight: 70.3
Current score: 0
```

(4)

3.2 An incomplete unit **Question3_U** has been provided and contains code for the object class to be accessible.

The following global variable has been declared:

The object **objPlayer**

Do the following to complete the code for QUESTION 3.2.1 to QUESTION 3.2.4 in the main form unit:

3.2.1 Button [3.2.1 - Instantiate object]

Write code to do the following:

- Use the name and weight of the player from the edit boxes provided to instantiate a Player object.
- Display a message, using a dialog box, to indicate that the object has been instantiated.

(5)

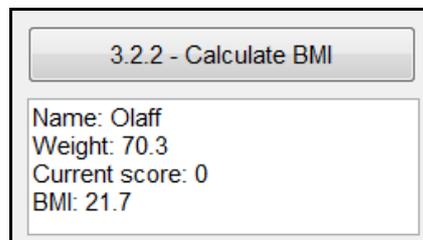
3.2.2 Button [3.2.2 - Calculate BMI]

The **redQ3_2_2** component must be used as the display area.

Write code to do the following:

- Use an input dialog box to enter the height of the player.
- Call the relevant method using the height as an argument to calculate the BMI of the player.
- Call the **toString** method to display the information of the player object.
- Display the BMI of the player, rounded off to ONE decimal place.

Example of output if the weight of the player is 70,3 kg and the height of the player is 1,80 m:



(7)

3.2.3 Button [3.2.3 - Update score]

The score of a player is updated as the game progresses.

The user must select a score in the radio group called **rgpQ3_2_3** and click the **Update score** button each time the player scores points during the game.

Write code to do the following:

- Extract the score that was selected from the component **rgpQ3_2_3**.
- Call the correct method to update the score attribute.
- Call the correct method to return the score.
- Display the updated score of the player in the **pnlQ3_2_3** component.

Example of output if the value of 2 was selected as the first score and the **Update score** button was clicked:

Example of output if the value of 5 was selected as the next score and the **Update score** button was clicked:

(6)

3.2.4 Button [3.2.4 - Eligible for selection]

Write code to call the method that returns a message indicating the player's eligibility for selection. Display the message in the label **lblQ3_2_4**.

Example of output if the current score of the player is 8:

Example of output if the current score of the player is 15:

(2)

- Ensure that your examination number has been entered as a comment in the first line of the object class and the form class.
- Save all files.
- Print the code if required.

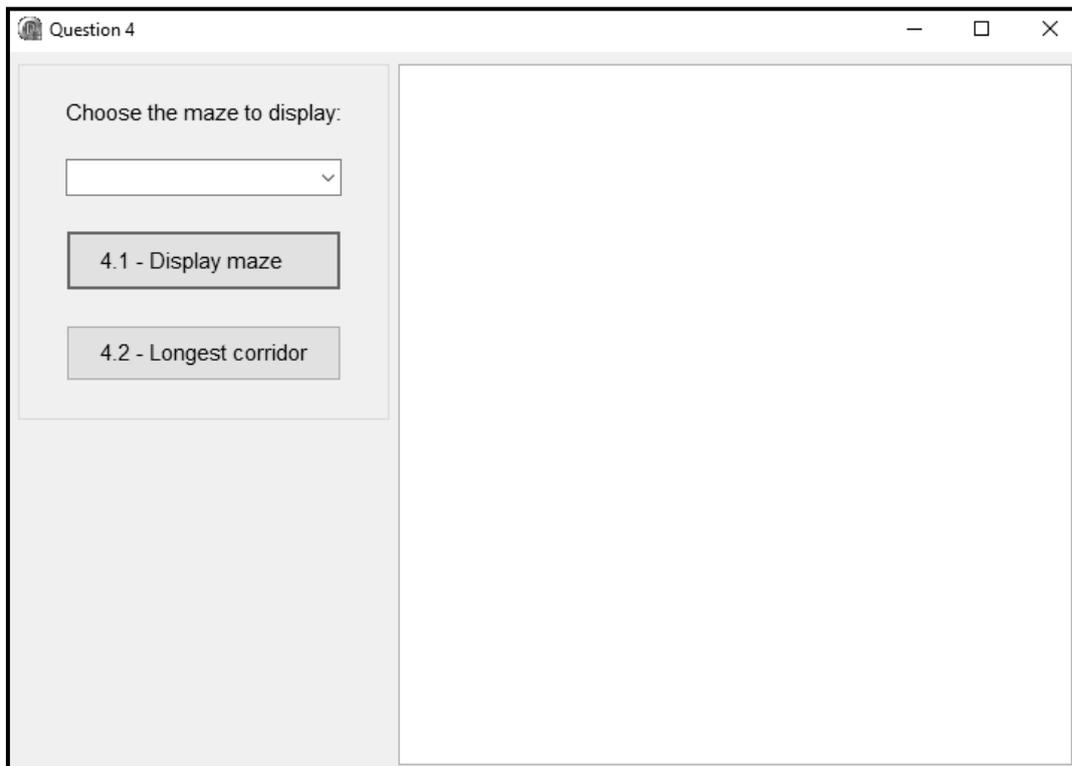
SECTION D**QUESTION 4: PROBLEM-SOLVING PROGRAMMING****SCENARIO**

A school uses a maze as a team-building activity for the rugby and netball first teams.

Do the following:

- Open the incomplete program in the **Question 4** folder.
- Enter your examination number as a comment in the first line of the **Question4_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

Example of the GUI:



The following code has been provided:

```
const
    iRowCount = 16;
var
    arrMaze : array[1..iRowCount] of String;
```

The maze is represented using various characters as follows:

- The '@' character indicates the outer border of the maze.
- The '#' character indicates a barrier (wall).
- The dash '-' character indicates an open space (corridor).

Complete the code for EACH section of QUESTION 4, as described in QUESTION 4.1 and QUESTION 4.2 that follow.

4.1 Button [4.1 - Display maze]

A combo box called **cmbQ4_1** has been populated with the names of the three mazes, namely **Maze 1**, **Maze 2** and **Maze 3**. Three text files called **Maze 1.txt**, **Maze 2.txt** and **Maze 3.txt** are provided. Each text file contains lines of characters that represent the layout of each maze respectively.

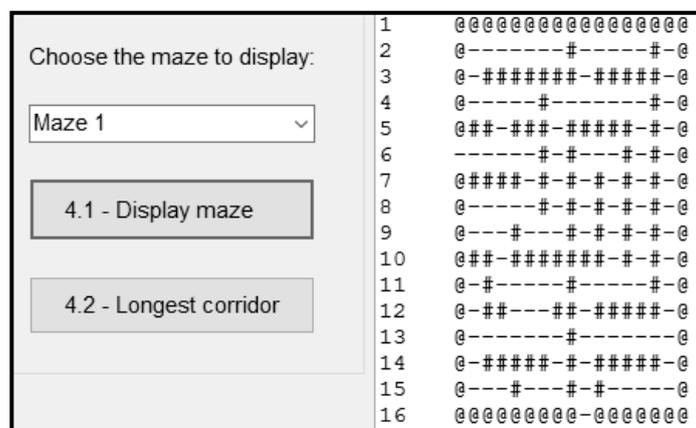
The user must select a maze from combo box **cmbQ4_1**.

Code has been provided to clear the component called **redQ4**.

Write code to do the following:

- Extract the name of the selected maze from combo box **cmbQ4_1**.
- Display a suitable message if a text file does NOT exist for the selected maze.
- Do the following if a text file for the selected maze does exist:
 - Read ONE line at a time from the text file.
 - Save EACH line at the correct index in the array **arrMaze** provided.
 - Display EACH line of the maze with its corresponding line number in the output component **redQ4**.

Example of output if **Maze 1** was selected from the combo box:



Example of output if **Maze 2** was selected from the combo box:

Choose the maze to display:	1	@@@@@@@@@@@@@@@@@@@@
	2	@-----#-----#--@
	3	@-#####-#####-@
	4	@-----#-----#--@
	5	@##-##-#####-#-@
	6	@####-#-#---#-#-@
	7	@####-#-#-#-#-#-@
	8	@-----#-#-#-#-#-@
	9	@##-#-#-#-#-#-#-@
	10	@##-#####-#-#-@
	11	--#-----#-----#--@
	12	@-##---##-#####-@
	13	@-----#-#-----@
	14	@-#####-#-#####-@
	15	@---#---#-----@
	16	@@@@@@@@@@@@@@@@@@@@

(11)

4.2 **Button [4.2 - Longest corridor]**

The longest corridor refers to the maximum number of consecutive horizontal open spaces in a row in the maze. An open space is represented by the dash ('-') character.

Write code to do the following:

- Determine the maximum number of consecutive horizontal open spaces in a row in the array **arrMaze**.
- Display the maximum number of consecutive horizontal open spaces as part of an output statement.
- Display a list of row(s) that contains a corridor with the maximum number of consecutive horizontal open spaces.

Example of output for **Maze 1**:

Choose the maze to display:	1	@@@@@@@@@@@@@@@@@@@@
	2	@-----#-----#--@
	3	@-#####-#####-@
	4	@-----#-----#--@
	5	@##-##-#####-#-@
	6	-----#-#---#-#-@
	7	@####-#-#-#-#-#-@
	8	@-----#-#-#-#-#-@
	9	@---#---#-#-#-#-@
	10	@##-#####-#-#-@
	11	@-#-----#-----#--@
	12	@-##---##-#####-@
	13	@-----#-----@
	14	@-#####-#-#####-@
	15	@---#---#-----@
	16	@@@@@@@@@@@@@@@@@@@@

Longest corridor(s) with 7 spaces in row(s):
2
4
13

Example of the output for **Maze 3**:

Choose the maze to display:	1 @@@@
Maze 3	2 @-----#-----#-@
4.1 - Display maze	3 @-#####-#####-@
4.2 - Longest corridor	4 @-----#-----#-@
	5 @##-###-#####-#-@
	6 @-----#-#-----#-@
	7 @###-#-#-#-#-#-#-@
	8 @-----#-#-#-#-#-@
	9 @---#--##-#-#-#-@
	10 @##-#####-#-#-@
	11 @-#-----#-----#-@
	12 @-##---##-#####-@
	13 @-----#-----@
	14 @-#####-#####-@
	15 @---#-----@
	16 @@@@@@@@@-@@@@@@@
	Longest corridor(s) with 11 spaces in row(s):
	15

(19)

- Ensure that your examination number has been entered 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 OF HockeyDB FOR QUESTION 2:**

The design of the database tables is as follows:

Table: **tblTeams**

This table contains the data of all the hockey coaches.

Field name	Data type	Description
TeamName (PK)	Text (10)	A unique team name. A team name is saved in the format 'u/18 A', where A indicates an A-team, B indicates a B-team and so on. The number 18 is the age group of the team players.
Coach	Text (25)	The surname and initials of the team's coach
NumberOfGamesPlayed	Integer	The total number of games the team played this season
NumberOfGamesWon	Integer	The total number of games the team won this season

Example of the first four records of the **tblTeams** table:

TeamName	Coach	NumberOfGamesPlayed	NumberOfGamesWon
u/14 A	Modikaze, P	8	4
u/14 B	Smith, GP	6	3
u/16 A	Smit, J	8	7
u/16 B	Mpofu, XB	6	3

Table: **tblPlayers**

This table contains the data of the hockey players of three different age groups:

Field name	Data type	Description
PlayerID (PK)	Autonumber	A unique number assigned to the player
PlayerSurname	Text (25)	The surname of the player
PlayerName	Text (25)	The name of the player
IDNumber	Text (20)	The South African ID number – first 6 digits is the person's date of birth in the format yymmdd.
TeamName	Text (10)	The team name that the player is a member of, e.g. 'u/18 A'
SkillsLevel	Integer	A value in the range 1 to 10 indicating the player's skills level. A value of 1 indicates a low skills level. A value of 10 indicates a high skills level.
GoalKeeper	Boolean	A value indicating whether the player is a goalkeeper (true) or not (false)

Example of the first four records of the **tbIPlayers** table:

PlayerID	PlayerSurname	PlayerName	IDNumber	TeamName	SkillsLevel	GoalKeeper
1	Fivaz	Peter	0306170790504	u/16 A	6	False
2	Maphaphu	Bukelwa	0609220225852	u/14 B	3	False
3	Snyders	Con	0602200518279	u/14 B	3	False
4	Jagers	Ben	0403060227644	u/16 B	4	True

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:

