Downloaded from Stanmorephysics.com

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/
NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

MAY/JUNE 2026

MARKS: 150

TIME: 3 hours

**This question paper consists of 26 pages, 2 data pages,
2 pages for planning and a separate information sheet.**

Copyright reserved

Downloaded from Stanmorephysics.com

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into FOUR sections. Candidates must answer ALL the questions in ALL FOUR sections.

2. Two blank pages have been provided at the end of the question paper, which may be used for planning purposes

3. An information sheet has been provided for you to complete at the end of the examination session. Ensure that ALL the information that you provided is correct and submit the information sheet before you leave the examination room.

4. The duration of this examination is three hours. Because of the nature of this examination, it is important to note that you will not be permitted to leave the examination room before the end of the examination session.

5. This question paper is set with programming terms that are specific to Delphi programming language. The Delphi programming language must be used to answer the questions.

6. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements.

7. Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.

8. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper.

9. Routines, such as search, sort and selection, must be developed from first principles. You may NOT use the built-in features of the Delphi programming language for any of these routines.

10. All data structures must be defined by you, the programmer, unless the data structures are supplied.

11. You must save your work regularly on the disk/CD/DVD/flash disk you have been given, or on the disk space allocated to you for this examination session.

12. Make sure that your examination number appears as a comment in every program that you code, as well as on every event indicated.

13. If required, print the programming code of all the programs/classes that you completed. Your examination number must appear on all the printouts. You will be given half an hour printing time after the examination session.

Copyright reserved

Downloaded from Stanmorephysics.com

14. At the end of this examination session, you must hand in a disk/CD/DVD/flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read.

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

    Do the following:

    - Double click on the following password-protected executable file: **IT P1 MayJune 2026 DATA.exe**.
    - Click on the 'Extract' button.
    - Enter the following password: **J26@Sea#**

    Once extracted, the following list of files will be available in the folder **IT P1 MayJune 2026 DATA**:

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

Question 2:
DBConnect_U.pas
MarineDB - Copy.mdb
MarineDB.mdb
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question 3:
Populate_U.pas
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas
SeaTurtle_U.pas

Question 4:
Fishing.txt
Question4_P.dpr
Question4_P.dproj
Question4_P.res
Question4_U.dfm
Question4_U.pas

Downloaded from Stanmorephysics.com

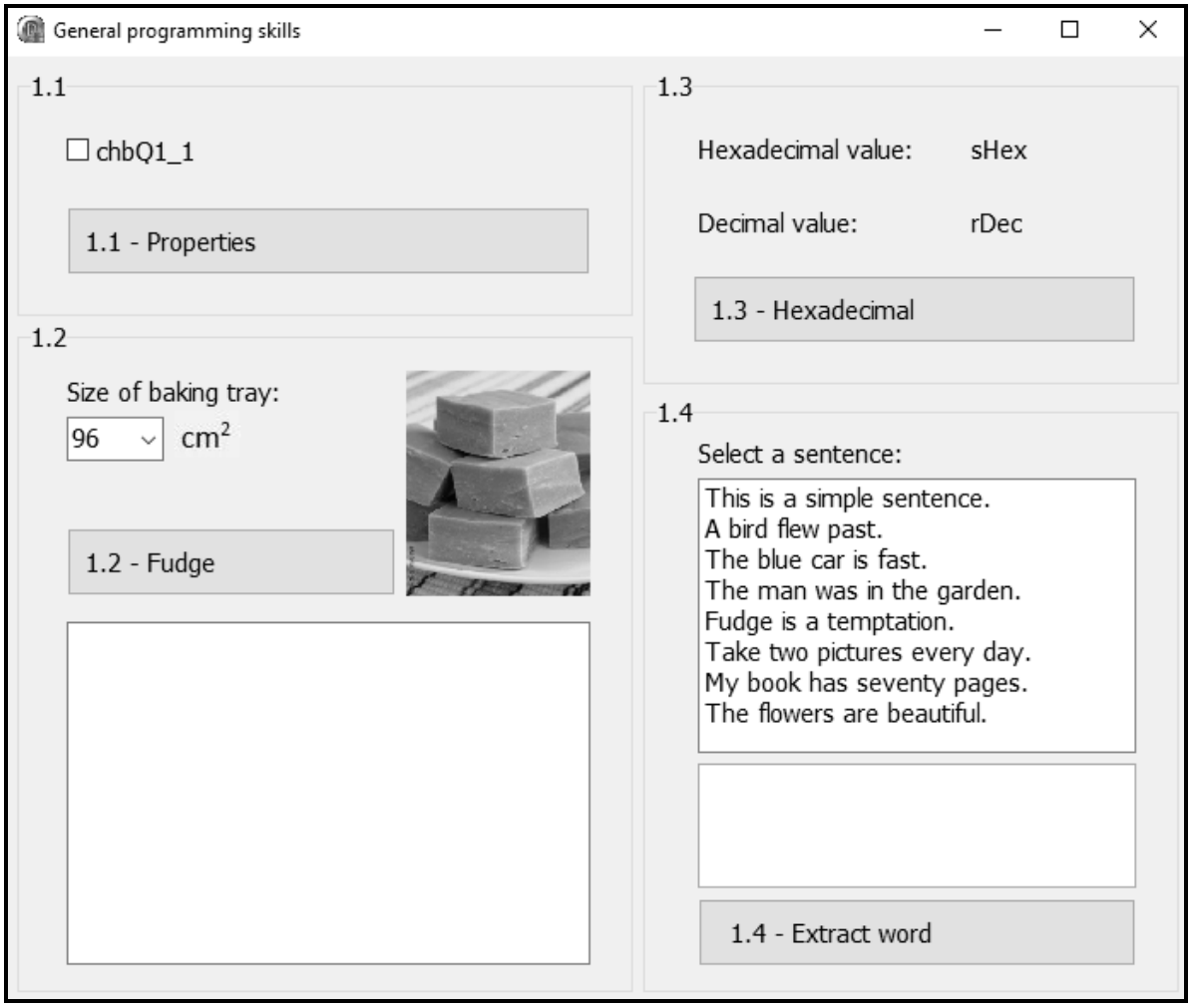
SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your examination number as a comment in the first line of the **Question1_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

Example of graphical user interface (GUI):



- Complete the code for each section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.4 on the next page.

Downloaded from Stanmorephysics.com

1.1 **Button [1.1 - Properties]**

Write code to apply the following changes to the check box component **chbQ1_1**:

- Set the caption to 'Marine World'.
- Ensure that the check box is ticked.
- Set the font to bold.
- Disable the check box.

Example of output:



(4)

1.2 **Button [1.2 - Fudge]**

Fudge, which is a type of sweet, needs to be prepared/made for a fundraising event. The following conditions apply during the preparation and packaging of the fudge:

- There are three different sizes of baking trays available, namely 96 cm^2, 144 cm^2 and 400 cm^2.
- Each piece of fudge should be 4 cm^2 in size.
- Each packet should contain six pieces of fudge.

The user must select the size of the baking tray to be used from combo box **cmbQ1_2**.

Code has been provided to clear the **memQ1_2** output area.

The following four integer variables have been provided and should be used in your solution:

```
iArea: Integer;
iNumPieces: Integer;
iPackets: Integer;
iExtra: Integer;
```

Write code to do the following:

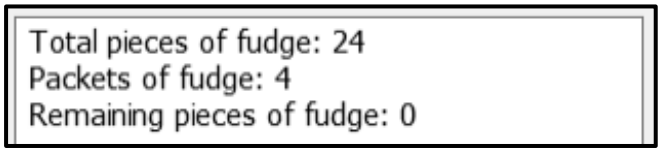
- Extract the size of the baking tray from combo box **cmbQ1_2**.
- Determine the total number of pieces of fudge that can be cut from the baking tray.
- Determine the number of packets of fudge that can be prepared from the total number of pieces of fudge.

Downloaded from Stanmorephysics.com

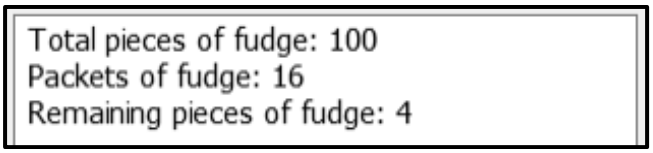
- Determine the number of remaining pieces of fudge that will not be enough to fill up a packet.
- Display the total number of pieces of fudge, the number of packets of fudge and the number of remaining pieces of fudge in the memo component **memQ1_2** in the format shown below.

```
Total pieces of fudge: <number of pieces of fudge>
Packets of fudge: <number of packets of fudge>
Remaining pieces of fudge: <number of remaining pieces of fudge>
```

Example of output if the size of the baking tray selected was 96 cm^2:



Example of output if the size of the baking tray selected was 400 cm^2:



(9)

1.3 **Button [1.3 - Hexadecimal]**

A hexadecimal value entered by the user will need to be converted to a decimal value. Validation of the input value will NOT be required.

The following variables have been provided:

```
sHex : String;
rDecimal : Real;
iExp : Integer;
```

Write code to do the following:

- Use an input box to enter a hexadecimal value and save the value in the variable **sHex**.
- Display the value saved in **sHex** in the label **lblQ1_3Hex**.
- Use the algorithm on the next page to convert the hexadecimal value saved in **sHex** into a decimal value:

Downloaded from Stanmorephysics.com

Initialisation:
- Set variable **rDecimal** to 0.
- Set variable **iExp** to 0.

Iteration:
For each character of the hexadecimal value, from the last character to the first character:
- Extract the character from the hexadecimal value.
- Use a CASE statement to determine the integer value of the character using the following conditions:
  - If the character is '0'-'9', type cast the character into an integer value and save the integer value.
  - If the character is 'A'-'F', save the specific integer value based on the following: A=10, B=11, C=12, D=13, E=14, F=15
- Calculate 16 raised to the power **iExp**, 16^{iExp}, using the power function.
- Multiply the character's integer value with the result of 16^{iExp}.
- Add the result of the previous step to **rDecimal**.
- Increase the **iExp** variable by the value of 1.

Output:
- Display the value of **rDecimal** in label **lblQ1_3Dec**.

Example of output if the hexadecimal value 1A was entered:

Hexadecimal value:	1A
Decimal value:	26

Example of output if the hexadecimal value A2B was entered:

Hexadecimal value:	A2B
Decimal value:	2603

(14)

1.4 **Button [1.4 - Extract word]**

List box **lstQ1_4** has been populated with several sentences. The user must select a sentence from the list box. To extract any word, apart from the first word, from the selected sentence, a random number must be generated repeatedly until the random number is the position/index of a space. The word following the identified space must be extracted and displayed.

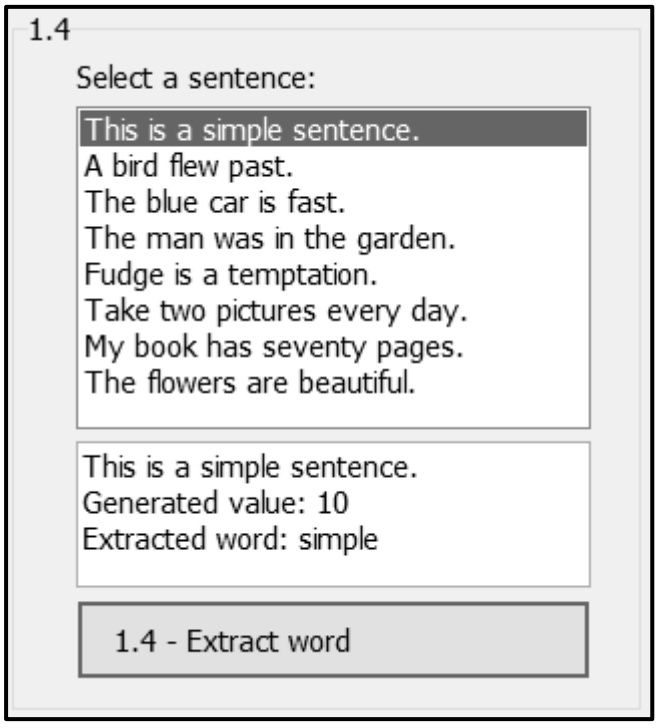
Downloaded from Stanmorephysics.com

Code has been provided to clear component **redQ1_4**.

Write code to do the following:

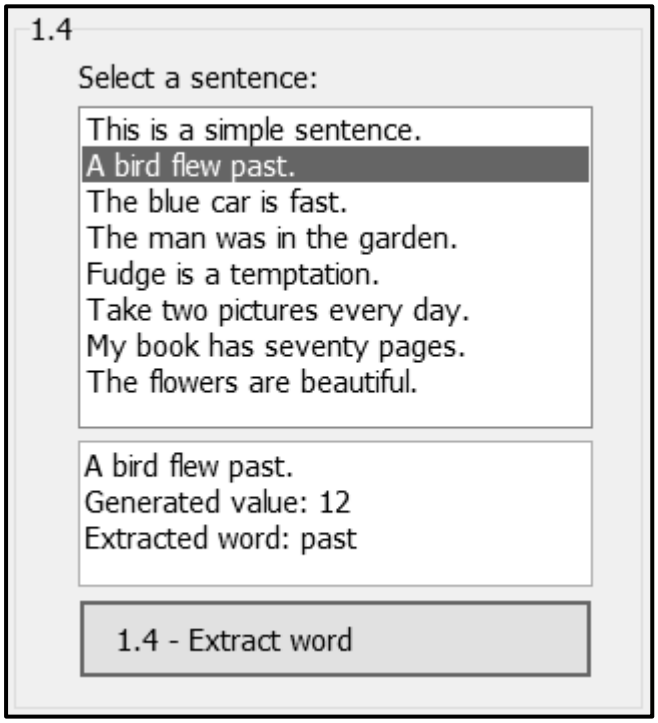
- Extract the selected sentence from list box **lstQ1_4**.
- Display the sentence in the rich edit component **redQ1_4**.
- Repeatedly generate a random number in the range from 1 to the length of the sentence until the character at the generated position/index in the sentence is a space.
- Display the random number in the rich edit component **redQ1_4**.
- Delete all the characters from the sentence up to the identified space, including the space.
- Find the position of the first space in the remaining part of the sentence.
  - Use this value to extract the first word in the remaining part of the sentence.
  - If the last word in the sentence is extracted, ensure that the full stop is not extracted as part of the word.
- Display the extracted word in the rich edit component **redQ1_4**.

Example of output if the first sentence was selected and 10 was the random value that was generated:



Downloaded from Stanmorephysics.com

Example of output if the second sentence was selected and 12 was the random value that was generated:



NOTE: Your output may be different due to the random values generated. (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 A: 40

Downloaded from Stanmorephysics.com

SECTION B

QUESTION 2: DATABASE PROGRAMMING

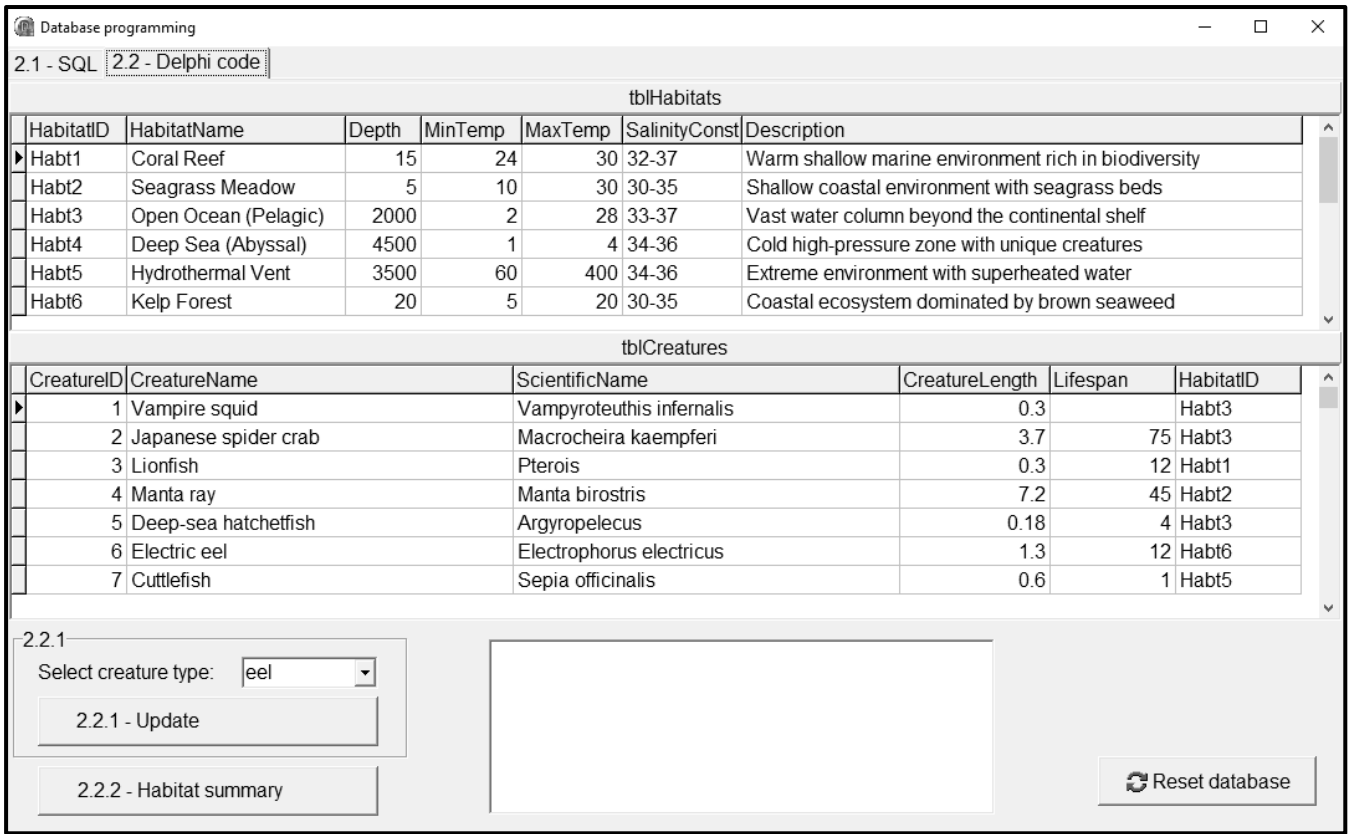
Marine World is creating a series of infographics for social media to highlight marine habitats and creatures. They need specific information from their database to feature facts about species, habitats and the conservation status of marine life.

A database called **MarineDB.mdb** has been developed, which contains two tables called **tblHabitats** and **tblCreatures**.

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

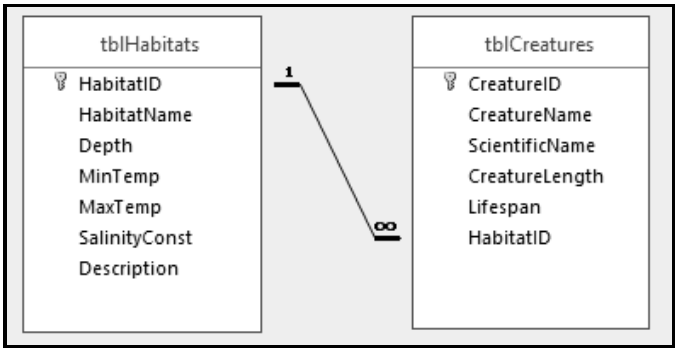
Do the following:

- Open the incomplete project file called **Question2_P.dpr** in the **Question 2** folder.
- Add your examination number as a comment in the first line of the **Question2_U.pas** file.
- Compile and execute the program. The program has limited functionality currently. The contents of the tables are displayed, as shown below on the selection of tab sheet **2.2 - Delphi code**.



Downloaded from Stanmorephysics.com

The relationship between the two tables **tblHabitats** and **tblCreatures** is shown below.



- 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.

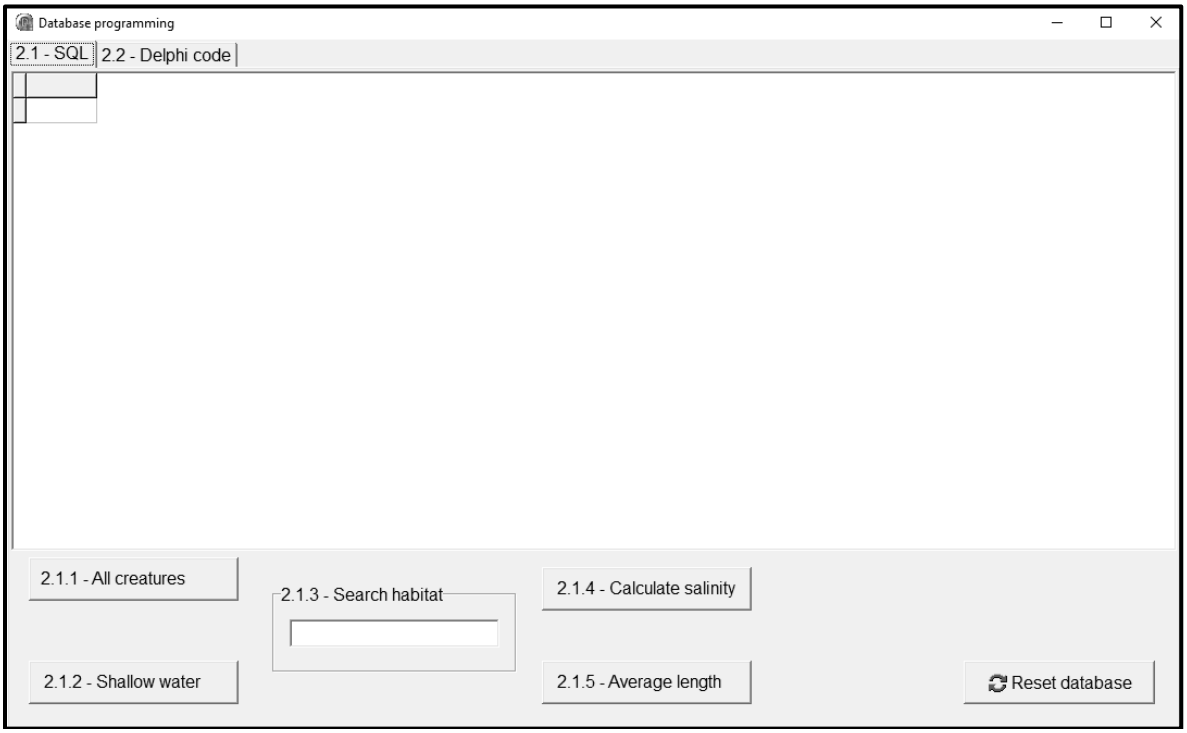
NOTE:

- The 'Reset database' button is provided to reset the data contained in the database to the original content.
- Code is provided to link the GUI components to the database. Do NOT change any of the code provided.
- TWO variables are declared as global variables, as described in the table below.

Variable	Data type	Description
`tblHabitats`	`TADOTable`	Refers to the table **tblHabitats**
`tblCreatures`	`TADOTable`	Refers to the table **tblCreatures**

2.1 Tab sheet [2.1 - SQL]

Example of graphical user interface (GUI) for QUESTION 2.1:



Downloaded from Stanmorephysics.com

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 has been provided. The SQL statements assigned to the global variable **sSQL** 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 - All creatures]**

Display all the details of each creature in the **tblCreatures** table sorted in alphabetical order according to **CreatureName**.

Example of output of the first eight records:

CreatureID	CreatureName	ScientificName	CreatureLength	Lifespan	HabitatID
30	Amerurus catfish	Nebulosus	0.25	12	Habt6
116	Amphipod	Gammarus pulex	0.9	3	Habt2
113	Anchor worm	Lernae cyprinacea	2	4	Habt1
112	Antlion	Myrmeleontidae	1.2	3	Habt2
111	Aphid	Aphidoidea Delta	0.8	1	Habt2
114	Apple fly	Rhagoletis pomonella	0.4	1	Habt6
115	Aquatic snail	Pysella acuta	1.5	5	Habt7
93	Axolotl	Ambystoma mexicanum	0.1	12	Habt6

(3)

2.1.2 **Button [2.1.2 - Shallow water]**

Display the **HabitatID**, **Depth** and **MinTemp** fields where the depth of the water is less than 25 m and the minimum temperature is 10 °C or more.

Example of output:

HabitatID	Depth	MinTemp
Habt1	15	24
Habt7	2.5	20
Habt2	5	10

(3)

2.1.3 **Edit box [2.1.3 - Search habitat]**

The user must enter a part of a **Description** in edit box **edtQ2_1_3**. Code has been provided to extract and store the description in variable **sFind**.

Display the **HabitatName** and **Description** of all the records where the description field contains the part of the description entered in the edit box.

Downloaded from Stanmorephysics.com

Example of output if 'WA' was entered:

HabitatName	Description
Coral Reef	Warm shallow marine environment rich in biodiversity
Open Ocean (Pelagic)	Vast water column beyond the continental shelf
Hydrothermal Vent	Extreme environment with superheated water
Arctic Ocean	Cold waters with seasonal ice cover

Example of output if 'Unique' was entered:

HabitatName	Description
Deep Sea (Abyssal)	Cold high-pressure zone with unique creatures

(4)

2.1.4 **Button [2.1.4 - Calculate salinity]**

SalinityConst is a field that stores a string value in the format XX-YY, where XX is the minimum salinity and YY is the maximum salinity. You may assume that XX and YY will always be a two-digit number.

Display the **HabitatID**, **SalinityConst** and a calculated field called **Salinity**. The salinity of the habitat is calculated by subtracting the minimum salinity from the maximum salinity in the **SalinityConst** field.

Example of output after the salinity was calculated:

HabitatID	SalinityConst	Salinity
Habt1	32-37	5
Habt3	33-37	4
Habt4	34-36	2
Habt5	34-36	2
Habt6	30-35	5
Habt7	10-35	25
Habt8	32-35	3
Habt9	33-35	2
Habt2	30-35	5

(4)

2.1.5 **Button [2.1.5 - Average length]**

Display the **HabitatName** and the average creature length, rounded to TWO decimal places only if the average creature length is greater than 1 metre.

Save the average creature length in a new field called **AvgLength**.

Example of output:

HabitatName	AvgLength
Hydrothermal Vent	3.12
Mangrove Swamp	1.5
Open Ocean (Pelagic)	1.07
Seagrass Meadow	6.45

(7)

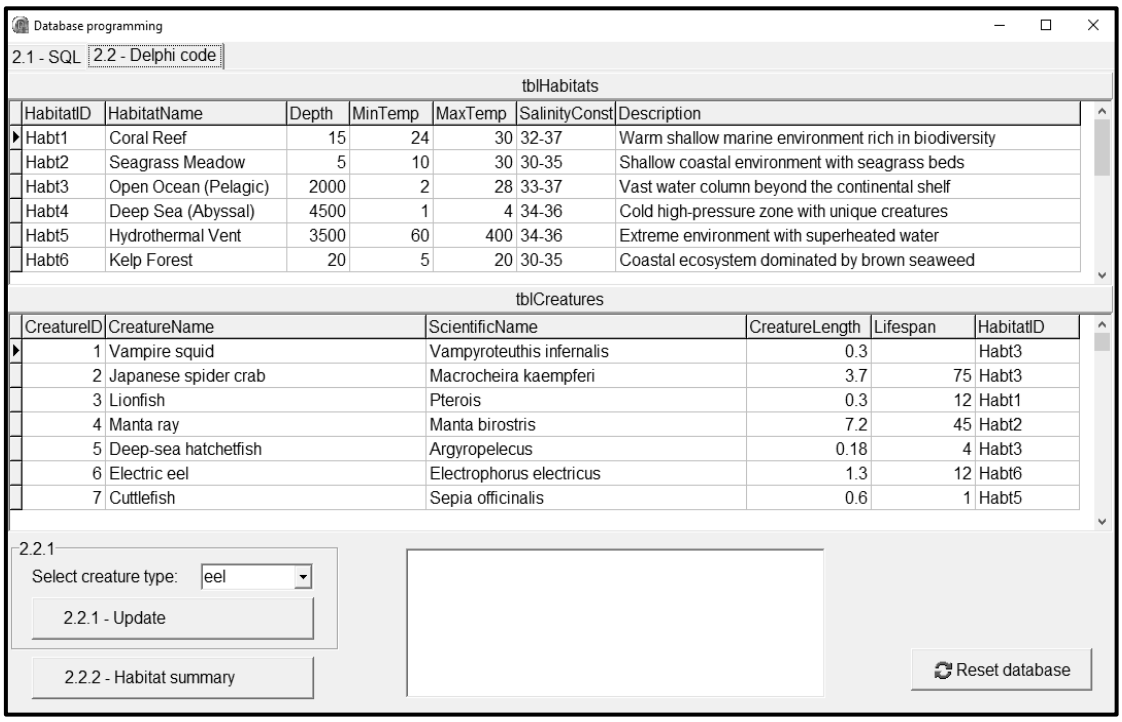
Downloaded from Stanmorephysics.com

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 graphical user interface (GUI) for QUESTION 2.2:



2.2.1 Button [2.2.1 - Update]

Creatures of a specific type will sometimes need to be moved to different habitats depending on various factors. The creature type is always included as a part of the **CreatureName** field in the table **tblCreatures**.

The user must select a creature type from combo box **cmbQ2_2_1**.

Code has been provided to:

- Extract and store the creature type in variable **sCreature**
- Clear and display a heading in the rich edit component **redQ2_2**

Write code to test if the creature name includes the creature type selected from the combo box, and update the creature name to include an asterisk character ('*') at the end. Display the **CreatureName** and the **HabitatID** fields in the rich edit component **redQ2_2**.

Downloaded from Stanmorephysics.com

Example of the first three lines of output if 'eel' was selected:

Creature name	Habitat ID
Electric eel*	Habt6
Gulper eel*	Habt3
Sphinxeel*	Habt3

(9)

2.2.2 **Button [2.2.2 - Habitat summary]**

A summary of the habitats and the number of creatures in each habitat must be compiled.

Code has been provided to clear and to display a heading in the rich edit component **redQ2_2**.

Write code to count the number of creatures in each habitat. Use the rich edit component **redQ2_2** to display each habitat name and the number of creatures in each habitat, only if there is at least one creature in the habitat.

Example of output of the first five habitat names and the number of creatures:

Habitat name	Number of creatures
Coral Reef	7
Seagrass Meadow	36
Open Ocean (Pelagic)	37
Hydrothermal Vent	8
Kelp Forest	15

(10)

- 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

Downloaded from Stanmorephysics.com

SECTION C

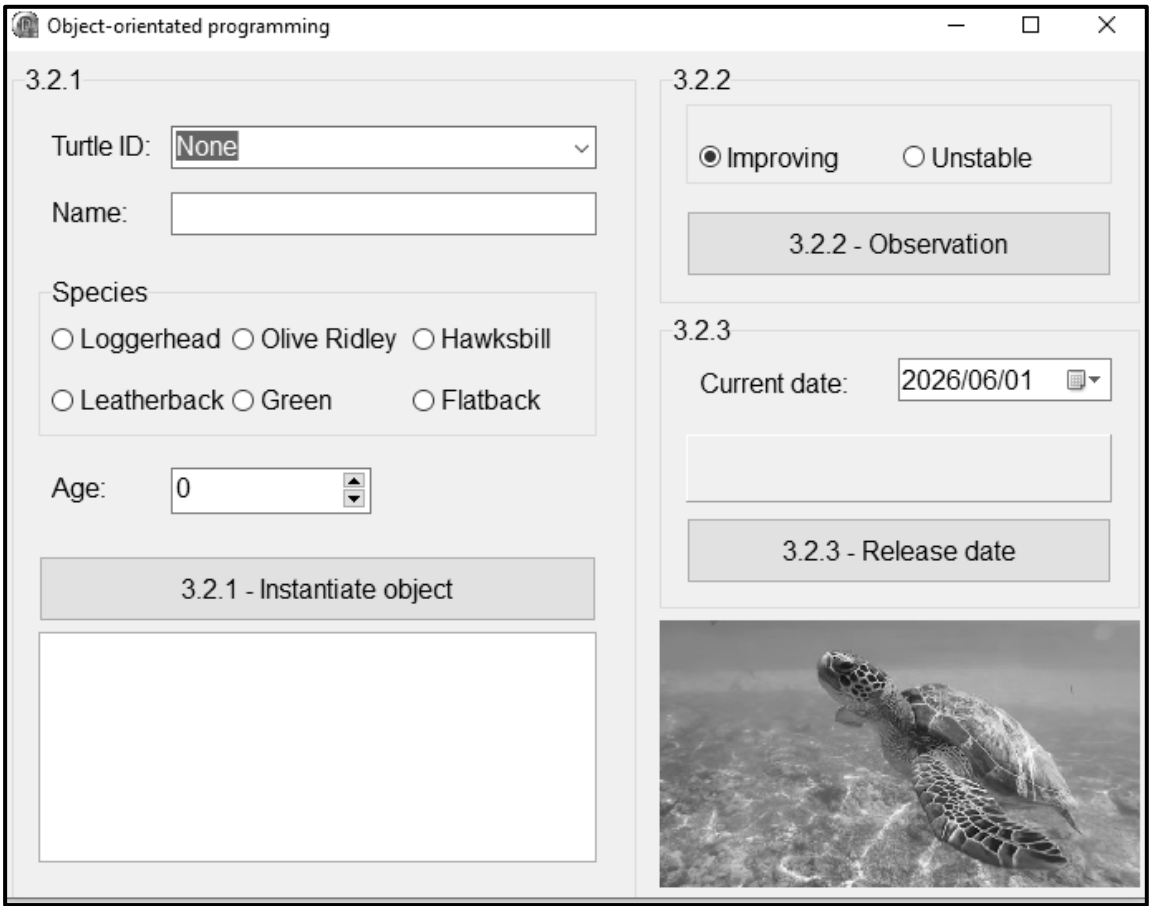
QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

A marine rescue centre requires assistance to record and update information about injured sea turtles during the turtles' rehabilitation process.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Open the incomplete object class **SeaTurtle_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question3_U.pas** file and the **SeaTurtle_U.pas** file.
- Compile and execute the program. The program has limited functionality currently.

Example of graphical user interface (GUI):



NOTE:

The date displayed on your interface may differ from the date displayed above, as the date will reflect the current system date.

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

Downloaded from Stanmorephysics.com

NOTE:

- You are NOT allowed to add any additional attributes or user-defined methods, unless explicitly stated in the question.
- The unit file **Populate_U** has been provided to populate the program with data. Do NOT change the content of the **Populate_U** unit.

3.1 The incomplete object class (**TSeaTurtle**) provided contains the declaration of six attributes which describe a **SeaTurtle** object.

Attribute	Data type	Description
fTurtleID	String	Unique identifier of the turtle
fName	String	A name given to the turtle
fSpecies	String	Turtle species
fAge	Integer	Estimated age in years
fInjurySeverity	Integer	Injury level from 1 (mild) to 10 (critical)
fRehabDays	Integer	Days spent in rehabilitation

The following methods have been provided:

- A partially completed **Constructor** method
- A completed **toString** method

Complete the code in the object class as described in QUESTION 3.1.1 to QUESTION 3.1.6 that follow.

3.1.1 Write code for an auxiliary method called **generateTurtleID** to generate and return a unique **TurtleID** for a sea turtle.

Generate a unique **TurtleID** in uppercase using the format provided below.

```
<first three letters of the sea turtle's species><last two letters of the sea turtle's name><fAge>
```

Example of a **TurtleID** if the name of the turtle is Sandy, the species is Loggerhead and the age is 48:

```
LOGDY48
```

(5)

3.1.2 The incomplete **Constructor** method receives four parameters and assigns the parameter values to the respective attributes. The value of 5 has been assigned to the **fInjurySeverity** attribute and the value of 60 has been assigned to **fRehabDays**.

Write code to complete the constructor method as follows:

If the **TurtleID** parameter is 'None', call the auxiliary method **generateTurtleID** to assign a unique TurtleID to the **fTurtleID** attribute. (4)

Downloaded from Stanmorephysics.com

3.1.3 Write code for a method called **injuryObservation** that receives the observation as a string parameter.

Observation parameter	**Do the following:**
Improving	Decrease the current injury severity of the object by the value of two. If the injury severity is a negative value, set the injury severity value to one.
Unstable	Change the injury severity to ten.

(6)

3.1.4 Write code for an accessor method called **getInjurySeverity** to return the **fInjurySeverity** attribute. (2)

3.1.5 Write code for a mutator method called **setRehabDays** that receives the number of days as an integer parameter and change the **fRehabDays** attribute to the value received. (3)

3.1.6 Write code for a method called **estimateReleaseDate** that receives the current date as a parameter.

Calculate the estimated release date by adding the **fRehabDays** to the current date and return the result. (3)

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 a global object variable called **objSeaTurtle**.

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

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

The combo box **cmbQ3_2_1** contains the **TurtleID**s of all previously tagged sea turtles. If a turtle has been tagged previously, the input components will be populated with the information of the sea turtle with the **TurtleID** selected from the combo box.

If the selected **TurtleID** is 'None', the user must enter information in the input components **edtQ3_2_1**, **rgpQ3_2_1** and **spnQ3_2_1**.

Code has been provided to:

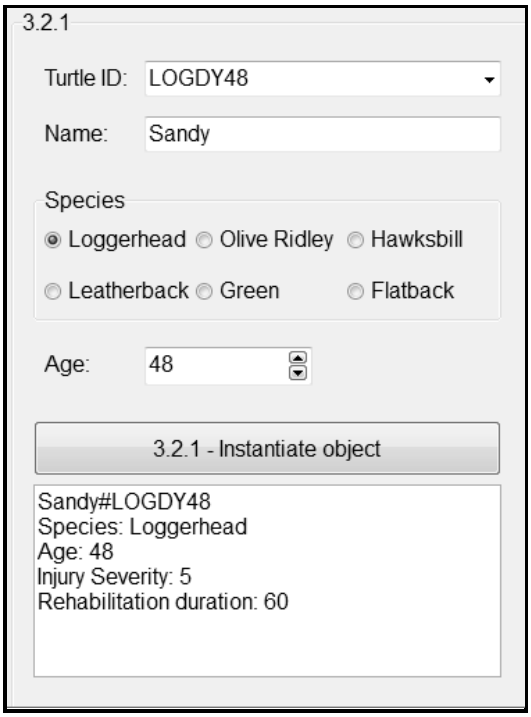
- Populate the input components when a **TurtleID** has been selected from combo box **cmbQ3_2_1**
- Extract the **TurtleID**, name, species and age from the relevant components, and assign to the respective variables provided
- Clear the rich edit component **redQ3_2**

Downloaded from Stanmorephysics.com

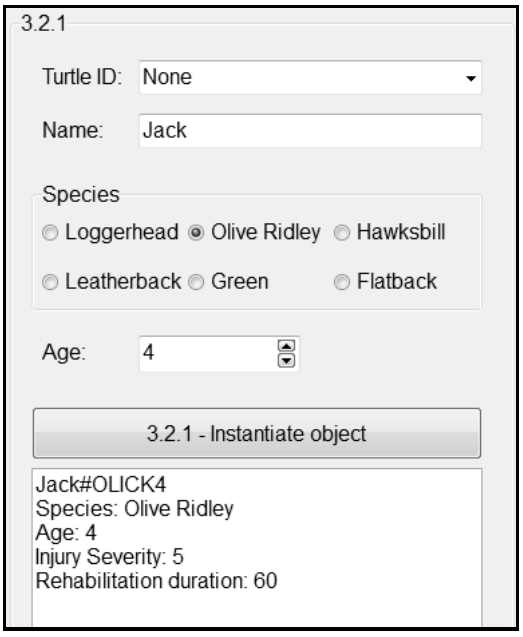
Write code to do the following:

- Use the information extracted to instantiate the **objSeaTurtle** object.
- Call the **toString** method of the object to display the sea turtle's information in the rich edit component **redQ3_2**.

Example of input and output if an existing **TurtleID** (LOGDY48) was selected:



Example of input and output if the selected **TurtleID** was NONE, the turtle name entered was Jack, the species selected was Olive Ridley and the selected age was 4:



(5)

Downloaded from Stanmorephysics.com

3.2.2 Button [3.2.2 - Observation]

Daily observations will be made to monitor the recovery of each sea turtle. Based on the observations, the injury severity is adjusted to reflect the healing progress or setbacks of the turtle.

The user must select an observation status from radio group **rgpQ3_2_2**.

Code has been provided to:

- Extract the observation, either 'Improving' or 'Unstable', from the radio group **rgpQ3_2_2** and store in the variable called **sObservation**
- Clear the rich edit component **redQ3_2**

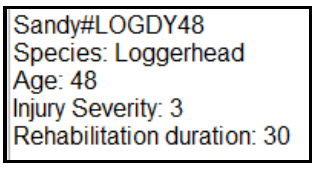
Write code to do the following:

- Call the **injuryObservation** method of the object, using the extracted observation as an argument to adjust the severity of the injury.
- Based on the adjusted injury severity, update the rehabilitation days attribute (**fRehabDays**) according to the following:

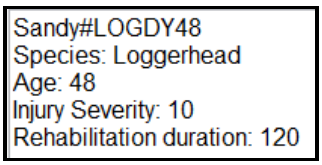
Injury severity	Rehabilitation days
1–4	30
5–7	60
8–10	120

- Call the **toString** method of the object to display the sea turtle's updated information in the rich edit component **redQ3_2**.

Example of output if the observation selected was 'Improving':



Example of output if the observation selected was 'Unstable':



(9)

Downloaded from Stanmorephysics.com

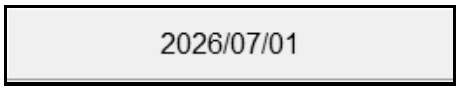
3.2.3 **Button [3.2.3 - Release date]**

The marine centre staff must calculate an estimated release date for the sea turtle to safely return to its natural ocean habitat.

Code has been provided to extract the date from the date-time-picker component **dtpQ3_2_3Current** and to store in the variable **dCurrentDate**.

Write code to call the relevant method to determine the release date and display the result in panel **pnlQ3_2_3**.

Example of input and output if the **TurtleID** was LOGDY48, the observation was 'Improving', the current date was 2026/06/01 and the rehabilitation days were 30:



(3)

- 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

Downloaded from Stanmorephysics.com

SECTION D

QUESTION 4: PROBLEM-SOLVING PROGRAMMING

QUESTION 4 consists of TWO parts:

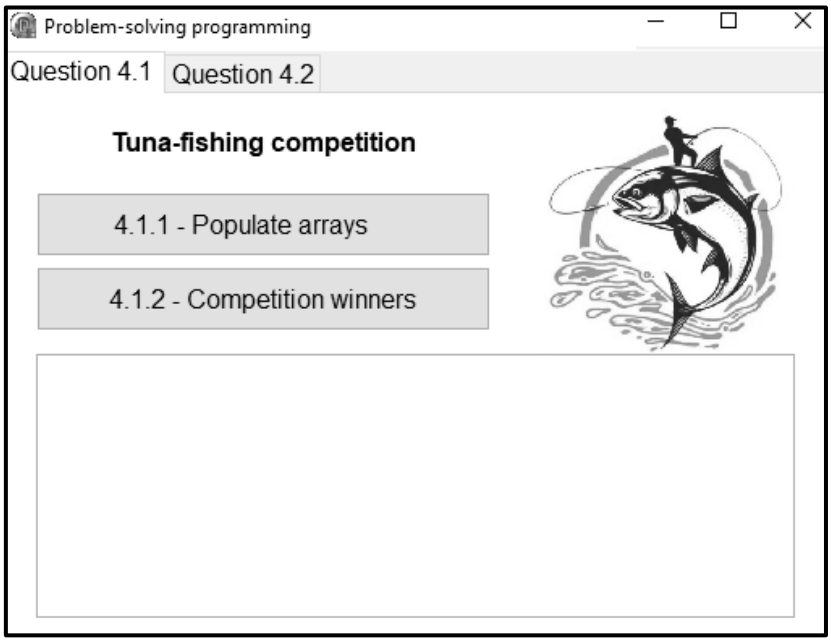
- QUESTION 4.1: Tuna-fishing competition
- QUESTION 4.2: Pascal's triangle

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 limited functionality currently.

4.1 A number of competitors recently took part in a tuna-fishing competition.

Example of graphical user interface (GUI):



The following have been provided:

- Three global parallel arrays each with a size of 30 have been declared:

```
arrSurnameName: array [1 .. 30] of String;
arrNumFish: array [1 .. 30] of Integer;
arrTotalWeight: array [1 .. 30] of Real;
```

- The declaration of a global integer variable, **iNumEntries**, to keep track of the actual number of entries in the arrays:

```
iNumEntries: Integer = 0;
```

Downloaded from Stanmorephysics.com

- A text file **Fishing.txt**, which contains the results of all the competitors who participated in the fishing competition, in the following format:

```
Entry number of the competitor
Surname, name of the competitor
Weight of each catch (in kg), each on a separate line
Blank line
```

Example of the results of the first four competitors stored in the text file:

```
#001
Vermaak, John
288.90
98.32
188.21

#002
Petersen, Lindiwe

#003
Davids, Peter
381.21
90.22

#004
Johnson, Gaby
510.12
```

Explanation of the information of the first two competitors saved in the text file:

- Vermaak John, with entry number #001, caught three fish. The weight of each fish (in kg) is listed on separate lines.

- Petersen Lindiwe, with entry number #002, did not catch any fish.

4.1.1 Button [4.1.1 - Populate arrays]

The content of the text file **Fishing.txt** must be used to populate the three parallel arrays **arrSurnameName**, **arrNumFish** and **arrTotalWeight**.

Write code to do the following:

- Open the provided text file, **Fishing.txt**, to read from the file. You may assume that the file exists.

Downloaded from Stanmorephysics.com

- Read through the text file and populate the three parallel arrays with the relevant data from the text file as explained below:
    - The array **arrSurnameName** should contain the surname and name of the competitor, separated by a comma, e.g. Vermaak, John.
    - The array **arrNumFish** should contain the number of fish caught by the competitor, e.g. John Vermaak caught three fish.
    - The array **arrTotalWeight** should contain the total weight of the fish caught by the competitor, e.g. the total weight of the fish caught by John Vermaak is 575,43 kg.
    - If a competitor did not catch any fish, a value of zero (0) must be saved in the **arrNumFish** and **arrTotalWeight** arrays for that competitor.

NOTE: A procedure called **displayInfo** has been provided and called at the end to display the content of the arrays.

Example of output of the results of the first seven competitors:

Surname and Name	Number of fish	Total weight
Vermaak, John	3	575.43
Petersen, Lindiwe	0	0
Davids, Peter	2	471.43
Johnson, Gaby	1	510.12
Smith, John	4	1596.23
Jones, Robert	3	1502.74
Williams, Richard	4	1546.09

(14)

4.1.2 Button [4.1.2 - Competition winners]

The winner is the competitor who caught fish with the highest average weight.

Code has been provided to:

- Clear the rich edit component **redQ4_1**
- Declare and initialise variable **rHighestAvg**

Write code to determine and display the name of the competitor who caught fish with the highest average weight. If more than one competitor qualifies to be a winner, display the names of all the winners on separate lines in the rich edit component **redQ4_1**.

NOTE: Code has been provided to display the highest average weight.

Example of output:

```
Taylor, Roger
Lewis, Ellen

Highest average: 591.87
```

(10)

Downloaded from Stanmorephysics.com

4.2 Button [4.2 - Pascal's triangle]

Pascal's triangle is a triangular arrangement of numbers that illustrates various mathematical patterns and relationships. Each number in the triangle is the sum of the two numbers diagonally above it, starting with 1 at the top and 1s along the sides.

If Pascal's triangle is represented in a two-dimensional (2D) array, the numbers will be arranged as shown below.

						1						
					1		1					
				1		2		1				
			1		3		3		1			
		1		4		6		4		1		
	1		5		10		10		5		1	
1		6		15		20		15		6		1

The following have been provided:

- A 2D array, **arrPattern[1..7,1..15]**, that has been declared and initialised:

```
arrPattern: array [1 .. 7, 1 .. 15] of Integer =
  ((0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0),
   (0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0),
   (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
   (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
   (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
   (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
   (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0));
```

NOTE:

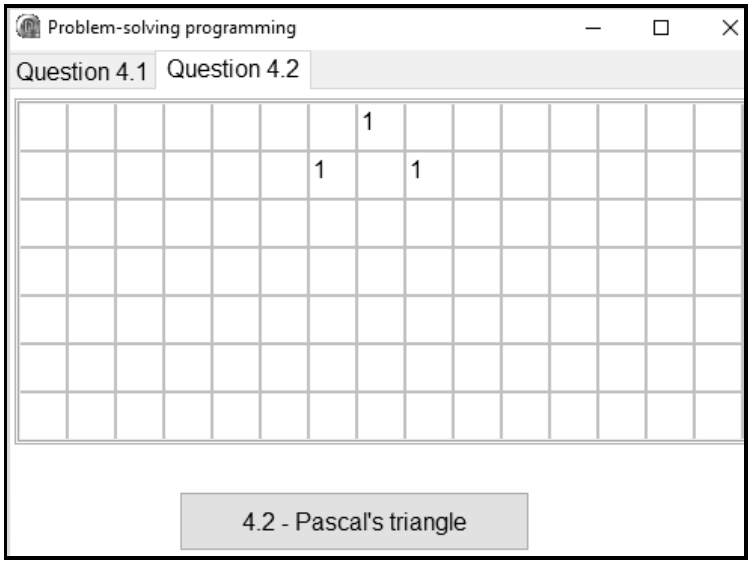
The only non-zero values currently in the provided 2D array **arrPattern** reflect the values in the first two rows of Pascal's triangle. These values are centred within the array at the following positions:

```
arrPattern[1, 8] := 1;
arrPattern[2, 7] := 1;
arrPattern[2, 9] := 1;
```

- A procedure named **display2D** to display the contents of the 2D array in StringGrid **sgdQ4_2**

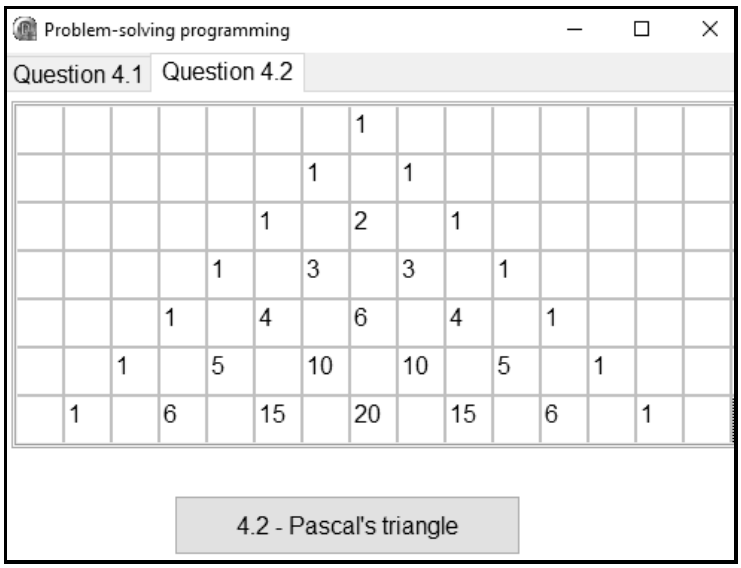
Downloaded from Stanmorephysics.com

Example of graphical user interface (GUI) with the first two rows in StringGrid **sgdQ4_2** populated with the values assigned to the **arrPattern** array, as explained on the previous page:



Write code to complete Pascal's triangle in the array **arrPattern** up to row 7 using iteration structures.

Example of output if the array is populated with values up to row 7 of Pascal's triangle:



NOTE: Code has been provided to call the **display2D** procedure. (6)

- 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

Copyright reserved

Downloaded from Stanmorephysics.com

INFORMATION TECHNOLOGY P1

DATABASE INFORMATION FOR QUESTION 2:

The design of the **MarineDB** database tables is as follows:

Table: **tblHabitats**

This table contains the details of each habitat.

Field Name	Data Type	Description
HabitatID	Short Text	Unique ID of the habitat, starts with 'Habt', followed by a number
HabitatName	Short Text	Name of the habitat
Depth	Number	Depth of the habitat in metres
MinTemp	Number	Common minimum temperature (°C) of the habitat
MaxTemp	Number	Common maximum temperature (°C) of the habitat
SalinityConst	Short Text	The salt concentration of the habitat in the format Minimum-Maximum
Description	Short Text	Short description of the habitat

Example of the records of the **tblHabitats** table:

HabitatID	HabitatName	Depth	MinTemp	MaxTemp	SalinityConst	Description
Habt1	Coral Reef	15	24	30	32-37	Warm shallow marine environment rich in biodiversity
Habt2	Seagrass Meadow	5	10	30	30-35	Shallow coastal environment with seagrass beds
Habt3	Open Ocean (Pelagic)	2000	2	28	33-37	Vast water column beyond the continental shelf
Habt4	Deep Sea (Abyssal)	4500	1	4	34-36	Cold high-pressure zone with unique creatures
Habt5	Hydrothermal Vent	3500	60	400	34-36	Extreme environment with superheated water
Habt6	Kelp Forest	20	5	20	30-35	Coastal ecosystem dominated by brown seaweed
Habt7	Mangrove Swamp	2.5	20	35	10-35	Coastal ecosystem with salt-tolerant trees
Habt8	Arctic Ocean	500	-2	5	32-35	Cold waters with seasonal ice cover
Habt9	Antarctic Ice Shelf	250	-2	2	33-35	Ice-covered marine ecosystem

Table: **tblCreatures**

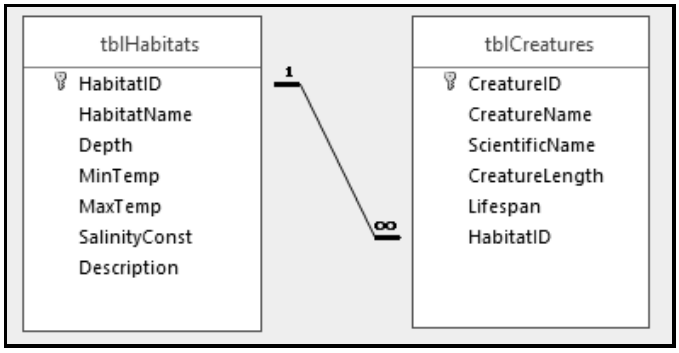
This table contains the details of each creature.

Field Name	Data Type	Description
CreatureID	Number	Unique ID of the creature
CreatureName	Short Text	Name of the creature
ScientificName	Short Text	Scientific name of the creature
CreatureLength	Number	The length of the creature in metres (m)
Lifespan	Number	The estimated lifespan of the creature in years. If the value is unknown, the field is left empty.
HabitatID	Short text	The ID of the habitat that identifies which habitat the creature belongs to

Downloaded from Stanmorephysics.com

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

CreatureID	CreatureName	ScientificName	CreatureLength	Lifespan	HabitatID
1	Vampire squid	Vampyroteuthis infernalis	0.3		Habt3
2	Japanese spider crab	Macrocheira kaempferi	3.7	75	Habt3
3	Lionfish	Pterois	0.3	12	Habt1
4	Manta ray	Manta birostris	7.2	45	Habt2
5	Deep-sea hatchetfish	Argyropelecus	0.18	4	Habt3
6	Electric eel	Electrophorus electricus	1.3	12	Habt6
7	Cuttlefish	Sepia officinalis	0.6	1	Habt5
8	Dumbo octopus	Grimpoteuthis	0.33	4	Habt3
9	Giant isopod	Bathynomus giganteus	0.51		Habt3
10	Flying fish	Exocoetidae	0.35	7	Habt2



Copyright reserved

Downloaded from Stanmorephysics.com

PLANNING PAGE 1

Copyright reserved

Downloaded from Stanmorephysics.com

PLANNING PAGE 2

Copyright reserved

Downloaded from Stanmorephysics.com

Examination sticker	150

INFORMATION TECHNOLOGY P1 – MAY/JUNE 2026
INFORMATION SHEET *(to be completed by the candidate AFTER the 3-hour examination session)*

CENTRE NUMBER: _____

EXAMINATION NUMBER: _____

WORK STATION NUMBER: _____

Version of Delphi used during the INFORMATION TECHNOLOGY SC/NSC MAY/JUNE 2026 Examination:

Mark appropriate box with a cross (**X**)	Delphi 2010	Delphi XE	Delphi 10.3	Delphi Community	Delphi 11	Delphi 12	Other: (*Specify*) _____

FOLDER NAME: _____

Candidate must tick if the file name(s) used for each answer has been saved and/or attempted.

Question number	File name	Saved (✓)	Attempted (✓)	Maximum Mark	Mark Achieved	Marker/ Code
1	Question1_P.dproj			**40**		
2	Question2_P.dproj			**40**		
3	SeaTurtle_U.pas			**23**		
	Question3_P.dproj			**17**		
4	Question4_P.dproj			**30**		
TOTAL				**150**		

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

_____

_____

Copyright reserved

Downloaded from Stanmorephysics.com

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/
NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

MAY/JUNE 2026

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 27 pages.

Copyright reserved

Downloaded from Stanmorephysics.com

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 10) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 11 to 27) contain examples of solutions for QUESTIONS 1 to 4 in programming code.

- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 10) should be made for each learner and completed during the marking session.

Downloaded from Stanmorephysics.com

ANNEXURE A

SUMMARY OF LEARNER'S MARKS:

CENTRE 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					

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
1.1	**Button [1.1 - Properties]** For chbQ1_1 do: • Change caption to 'Marine World' ✓ • Checked to TRUE ✓ • Font style to [fsBold] ✓ • Enabled to FALSE ✓	**4**	
1.2	**Button [1.2 - Fudge]** Extract area from cmbQ1_2 ✓ Converted to integer ✓ iNumPieces = iArea ✓ DIV 4 ✓ iPackets = iNumPieces DIV 6 ✓ iExtra = iNumPieces MOD 6 ✓ Display the output in memQ1_2 on three separate lines ✓ With correct labels ✓ With correct variables converted to string ✓	**9**	

Downloaded from Stanmorephysics.com

1.3	**Button [1.3 - Hexadecimal]** Extract sHex using InputBox ✓ Display sHex on lblQ1_3Hex ✓ Initialise rDecimal and iExp to 0 ✓ Determine Length(sHex) ✓ Loop from length of sHex downto 1 ✓ Extract character ✓ Case character of ✓ '0'..'9' convert character to integer and assign to iValue variable ✓ 'A'..'F' ✓ assign corresponding integer value to iValue ✓ rDecimal = rDecimal ✓ + iValue * Power(16, iExp) ✓ increment (iExp) ✓ Display rDecimal converted to string in lblQ1_3Dec ✓	**14**	
1.4	**Button [1.4 - Extract word]** Extract sentence selected by user from lstQ1_4 ✓ Display sentence in redQ1_4 ✓ Conditional loop to: ✓ Generate random index value ✓ in the range 1 to length of sentence ✓ Until character at position of random value is space ✓ Display generated value in redQ1_4 converted to string ✓ Delete text in sentence from 1 to position of space ✓ Find the first space in the remaining part of the sentence ✓ Test if not last word in sentence ✓ Extract word from 1 to position of space - 1 ✓ Else Extract word from 1 to length of sentence - 1 ✓ Display the word in redQ1_4 ✓ **Alternative for finding next space:** Loop through remaining of sentence (1) Test for space (1)	**13**	
	TOTAL SECTION A:	**40**	

Downloaded from Stanmorephysics.com

ANNEXURE B

QUESTION 2: MARKING GRID – DATABASE PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**		
2.1.1	**Button [2.1.1 - All creatures]** SELECT * ✓ FROM tblCreatures ✓ ORDER BY CreatureName ✓	**3**	
2.1.2	**Button [2.1.2 - Shallow water]** SELECT HabitatID, Depth, MinTemp FROM tblHabitats ✓ WHERE Depth < 25 ✓ AND MinTemp >= 10 ✓	**3**	
2.1.3	**Edit box [2.1.3 - Search habitat]** SELECT HabitatName, Description ✓ FROM tblHabitats ✓ WHERE Description LIKE ✓ "%' + sFind + '%" ✓ **Also ACCEPT:** LIKE ' + quotedstr('%' + sFind + '%')	**4**	
2.1.4	**Button [2.1.4 - Calculate salinity]** SELECT HabitatID, SalinityConst, RIGHT(SalinityConst,2) ✓-LEFT(SalinityConst,2) ✓ AS Salinity ✓ FROM tblHabitats ✓	**4**	
2.1.5	**Button [2.1.5 - Average length]** SELECT HabitatName, ROUND(AVG(CreatureLength) ✓,2 ✓) AS AvgLength FROM tblHabitats H, tblCreatures C ✓ WHERE H.HabitatID = C.HabitatID ✓ GROUP BY HabitatName ✓ HAVING ✓ AVG(CreatureLength) > 1 ✓	**7**	
	Subtotal:	**21**	

Downloaded from Stanmorephysics.com

QUESTION 2: MARKING GRID (CONT.)

2.2	**Database Manipulation**		
2.2.1	**Button [2.2.1 - Update]** Go to the first record in tblCreatures ✓ Loop through tblCreatures ✓ Test if sCreature ✓ is contained in tblCreatures['CreatureName'] ✓ tblCreatures.Edit✓ tblCreatures['CreatureName'] = tblCreatures['CreatureName'] + '*' ✓ tblCreatures.Post Display the CreatureName and HabitatID ✓ in redQ2_2 inside IF-statement ✓ tblCreatures.Next ✓ End loop (tblCreatures)	**9**	
2.2.2	**Button [2.2.2 - Habitat summary]** Go the first record in tblHabitats Loop through tblHabitats ✓ Set Counter = 0 ✓ Go to the first record in tblCreatures Loop through tblCreatures ✓ If tblCreatures['HabitatID'] = tblHabitats['HabitatID'] ✓ Increment Counter ✓ tblCreatures.Next ✓ End loop (tblCreatures) If Counter > 0 ✓ Display HabitatName + #9✓ + Counter converted to string ✓ tblHabitats.Next ✓ End loop (tblHabitats) **NOTE:** One mark allocated for the correct use of Tablename.First and Tablename.Next for each table.	**10**	
	Subtotal:	**19**	
	TOTAL SECTION B:	**40**	

Downloaded from Stanmorephysics.com

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING

CENTRE NUMBER:		**EXAMINATION NUMBER:**	
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**Function generateTurtleID** Function heading with string return type and result ✓ Result = UpperCase ✓ (Copy first 3 characters of fSpecies ✓ + Copy last 2 characters of fName) ✓ + fAge converted to string ✓	**5**	
3.1.2	**Constructor Create** The following must be tested <u>after</u> the provided code ✓ Test if TurtleID ✓ = 'None' ✓ fTurtleID = generateTurtleID ✓	**4**	
3.1.3	**Procedure injuryObservation** Procedure heading with string parameter ✓ Test if parameter value = 'Improving' ✓ Decrease fInjurySeverity by 2 ✓ Test If fInjurySeverity < 0 change to 1 ✓ Test if parameter value = 'Unstable' ✓ fInjurySeverity = 10 ✓	**6**	
3.1.4	**Function getInjurySeverity** Function heading with integer return type ✓ Result = fInjurySeverity ✓	**2**	
3.1.5	**Procedure setRehabDays** Procedure heading ✓ with integer parameter ✓ fRehabDays = parameter value ✓	**3**	
3.1.6	**Function estimateReleaseDate** Function heading with date parameter ✓ and date return type ✓ Result = fRehabDays + parameter value ✓	**3**	
	Subtotal: Object class	**23**	

Downloaded from Stanmorephysics.com

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 - Instantiate object]** objSeaTurtle := ✓ TSeaTurtle.Create ✓ (sTurtleID, sName, sSpecies, iAge) ✓ Display objSeaTurtle in redQ3_2 ✓ using toString method ✓	**5**	
3.2.2	**Button [3.2.2 - Observation]** Call injuryObservation ✓ (sObservation) ✓ Test if ✓ getInjurySeverity ✓ 1 .. 4: Days = 30 ✓ 5 .. 7: Days = 60 ✓ 8 ..10:Days = 120 ✓ Call setRehabDays(Days) ✓ Call toString to display in redQ3_2 ✓	**9**	
3.2.3	**Button [3.2.3 - Release date]** Display on pnlQ3_2_3: Converted to string ✓ Call estimatedReleaseDate ✓ with dCurrentDate as argument ✓	**3**	
	Subtotal Form class:	**17**	
	TOTAL SECTION C:	**40**	

Downloaded from Stanmorephysics.com

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1.1	**Button [4.1.1 - Populate arrays]** AssignFile(tFile, 'Fishing.txt') ✓ Reset(tFile) ✓ Loop through the text file with correct condition ✓ Increment iNumEntries ✓ Readln(tFile, sTemp) ✓ Readln(tFile, sLine) ✓ arrSurnameName[iNumEntries] = sLine ✓ Readln(tFile, sLine) Initialise NumberFish and TotalWeight to 0 ✓ Loop through lines while sLine <> empty ✓ Increment NumberFish ✓ TotalWeight = TotalWeight + sLine as Real ✓ Readln(tFile, sLine) ✓ (initialise and change sLine) arrNumFish[iNumEntries] = NumberFish ✓ arrTotalWeight[iNumEntries] = TotalWeight ✓ CloseFile(tFile) **Concepts:** File Open and Reset (2) Loop through text file (1) Read a line of text file (1) Number of entries (participants) (1) Obtain (1) and save name and surname in array (1) Initialise number of fish and total weight (1) Obtain (1), calculate (1) and save the number of fish in array (1) Obtain (1), calculate (1) and save the total weight in array (1)	**14**	

Downloaded from Stanmorephysics.com

4.1.2	**Button [4.1.2 - Competition winners]** Loop from 1 to iNumEntries ✓ Test each entry if number of fish is greater than 0 ✓ Calculate the average weight: arrTotalWeight[Loop] ✓ /arrNumFish[Loop] ✓ Test if average weight is equal to rHighestAvg ✓ Add arrSurnameName[Loop] to winner string ✓ Test if average is greater than rHighestAvg ✓ Replace winner string with arrSurnameName[Loop] ✓ Replace rHighestAvg with average weight ✓ End loop Display winner string in redQ4_1 ✓	**10**	
4.2	**Button [4.2 - Pascal's triangle]** Loop iRow from 3 to 7 ✓ Nested loop ✓ iCol from 2 to 14 ✓ arrPattern[iRow,iCol] = ✓ arrPattern[iRow -1,iCol -1] ✓ + arrPattern[iRow -1,iCol +1] ✓	**6**	
	TOTAL SECTION D: **GRAND TOTAL:**	**30** **150**	

Downloaded from Stanmorephysics.com

ANNEXURE E: SOLUTION FOR QUESTION 1

```
unit Question1_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, ExtCtrls, StdCtrls, ComCtrls, Math, jpeg, DateUtils;

type
  TfrmQuestion1 = class(TForm)
    grpQ1_1: TGroupBox;
    grpQ1_4: TGroupBox;
    btnQ1_1: TButton;
    lstQ1_4: TListBox;
    redQ1_4: TRichEdit;
    grpQ1_3: TGroupBox;
    lblQ1_3Hex: TLabel;
    btnQ1_3: TButton;
    lblQ1_3Dec: TLabel;
    lblHex: TLabel;
    lblDec: TLabel;
    grpQ1_2: TGroupBox;
    imgFudge: TImage;
    lblArea: TLabel;
    memQ1_2: TMemo;
    btnQ1_2: TButton;
    lblSelectSentence: TLabel;
    btnQ1_4: TButton;
    cmbQ1_2: TComboBox;
    chbQ1_1: TCheckBox;
    procedure btnQ1_1Click(Sender: TObject);
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure FormShow(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion1: TfrmQuestion1;

implementation

{$R *.dfm}
```

Downloaded from Stanmorephysics.com

```
//=============================================================================
// Question 1.1                                                        4 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  // Question 1.1
  chbQ1_1.Caption := 'Marine World';
  chbQ1_1.Checked := True;
  chbQ1_1.Font.Style := [fsBold];
  chbQ1_1.Enabled := False;
end;
//=============================================================================
// Question 1.2                                                        9 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  iArea: Integer;
  iNumPieces: Integer;
  iPackets : Integer;
  iExtra : Integer;
begin
  // Provided code
  memQ1_2.Clear;
  // Question 1.2
  iArea := StrToInt(cmbQ1_2.Text);
  iNumPieces := iArea DIV 4;
  iPackets := iNumPieces DIV 6;
  iExtra := iNumPieces MOD 6;
  memQ1_2.Lines.Add('Total pieces of fudge: ' + IntToStr(iNumPieces));
  memQ1_2.Lines.Add('Packets of fudge: ' + IntToStr(iPackets));
  memQ1_2.Lines.Add('Remaining pieces of fudge: ' + IntToStr(iExtra));
end;
//=============================================================================
// Question 1.3                                                       14 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  sHex: String;
  rDecimal: Real;
  iExp, iValue, iCnt: Integer;
  cChar: Char;
begin
  // Question 1.3
  sHex := InputBox('Convert Hexadecimal to Decimal',
          'Enter hexadecimal value:', 'A2B');
  lblQ1_3Hex.Caption := sHex;
  rDecimal := 0;
  iExp := 0;
  for iCnt := Length(sHex) downTo 1 do
  begin
    cChar := sHex[iCnt];
```

Downloaded from Stanmorephysics.com

```
    case cChar of
      '0' .. '9': iValue := StrToInt(cChar);
      'A': iValue := 10;
      'B': iValue := 11;
      'C': iValue := 12;
      'D': iValue := 13;
      'E': iValue := 14;
      'F': iValue := 15;
    //'A'..'F' : iValue := ORD(cChar) - ORD('A') + 10; // Alternative
    end;
    rDecimal := rDecimal + iValue * Power(16, iExp);
    Inc(iExp);
  end;
  lblQ1_3Dec.Caption := FloatToStrF(rDecimal, ffFixed, 8, 0);
end;
//============================================================================
// Question 1.4                                                    13 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sSentence, sWord: String;
  iPos, iLen, iRandom, iIndex: Integer;
begin
  // Provided code
  redQ1_4.Clear;
  // Question 1.4
  sSentence := lstQ1_4.Items[lstQ1_4.ItemIndex];
  redQ1_4.Lines.Add(sSentence);

  repeat
    iRandom := Random(length(sSentence) + 1) + 1;
  until sSentence[iRandom] = ' ';

  redQ1_4.Lines.Add('Generated value: ' + IntToStr(iRandom));

  Delete(sSentence, 1, iRandom);
  iPos := Pos(' ', sSentence);
  if iPos > 0 then
    sWord := Copy(sSentence, 1, iPos - 1)
  else
    sWord := Copy(sSentence, 1, length(sSentence) - 1);

  redQ1_4.Lines.Add('Extracted word: ' + sWord);
end;

// ============================================================================
// Provided code
// ============================================================================
{$REGION 'Provided code - DO NOT CHANGE'}
procedure TfrmQuestion1.FormShow(Sender: TObject);
begin
  lstQ1_4.ItemIndex := 0;
end;
{$ENDREGION}
end.
```

Downloaded from Stanmorephysics.com

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;

interface

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

type
  TfrmQuestion2 = class(TForm)
    dbgOne: TDBGrid;
    dbgMany: TDBGrid;
    dbgSQL: TDBGrid;
    pgcQuestion2: TPageControl;
    tsQuestion2_1: TTabSheet;
    tsQuestion2_2: TTabSheet;
    pnlHabitats: TPanel;
    pnlCreatures: TPanel;
    redQ2_2: TRichEdit;
    btnQ2_2_1: TButton;
    btnQ2_2_2: TButton;
    btnReset: TBitBtn;
    btnQ2_1_1: TButton;
    btnQ2_1_3: TButton;
    edtQ2_1_3: TEdit;
    btnQ2_1_2: TButton;
    gpb2_1_3: TGroupBox;
    btnQ2_1_4: TButton;
    btnQ2_1_5: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure edtQ2_1_3Change(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
  private
    procedure runSQL(sSQL: string);
  public
    { Public declarations }
  end;

var
  frmQuestion2: TfrmQuestion2;
  objDB: TDBConnect;
  tblCreatures, tblHabitats: TADOTable;
  sSQL : String;

implementation
{$R *.dfm}
```

Downloaded from Stanmorephysics.com

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

//============================================================================
//  Question 2.1.1                                                    3 marks
//============================================================================
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
begin
  //Question 2.1.1

  sSQL := 'SELECT * ' +
          'FROM tblCreatures ' +
          'ORDER BY CreatureName';

  //Provided code
  runSQL(sSQL);
end;
//============================================================================
//  Question 2.1.2                                                    3 marks
//============================================================================
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
begin
  //Question 2.1.2

  sSQL := 'SELECT HabitatID, Depth, MinTemp ' +
          'FROM tblHabitats '  +
          'WHERE Depth < 25 AND MinTemp >= 10';

  //Provided code
  runSQL(sSQL);
end;
//============================================================================
//  Question 2.1.3                                                    4 marks
//============================================================================
procedure TfrmQuestion2.edtQ2_1_3Change(Sender: TObject);
 // Provided code
 //DO NOT change
var
  sFind: string;
begin
  //Question 2.1.3
  sFind := edtQ2_1_3.Text;
  //Provided code

  sSQL := 'SELECT HabitatName, Description ' +
          'FROM tblHabitats ' +
          'WHERE Description LIKE "%' + sFind + '%"';

  //Provided code
  runSQL(sSQL);
end;
```

Downloaded from Stanmorephysics.com

```
//============================================================================
//  Question 2.1.4                                                  4 marks
//============================================================================
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
begin
  //Question 2.1.4

  sSQL := 'SELECT HabitatID, SalinityConst, RIGHT(SalinityConst,2) -
               LEFT(SalinityConst,2) AS Salinity FROM tblHabitats ';

  //Provided code
  runSQL(sSQL);
end;
//============================================================================
//  Question 2.1.5                                                  7 marks
//============================================================================
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
begin
  //Question 2.1.5

  sSQL := 'SELECT H.HabitatName, ROUND(AVG(CreatureLength),2)
     AS AvgLength ' +
          'FROM tblHabitats H, tblCreatures C ' +
          'WHERE H.HabitatID = C.HabitatID ' +
          'GROUP BY HabitatName ' +
          'HAVING AVG(CreatureLength) > 1';

  //Provided code
  runSQL(sSQL);
end;
```

Downloaded from Stanmorephysics.com

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

//============================================================================
//  Question 2.2.1                                                  9 marks
//============================================================================
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  sCreature: String;
begin
  // Provided code
    sCreature := cmbQ2_2_1.Text;
    redQ2_2.Clear;
    redQ2_2.Lines.Add('Creature name' + #9 + 'Habitat ID');
    redQ2_2.Lines.Add('');

  // Question 2.2.1
  tblCreatures.First;
  while not tblCreatures.EOF do
  begin
    if Pos( sCreature, tblCreatures['CreatureName'] ) > 0 then
    begin
      tblCreatures.Edit;
      tblCreatures['CreatureName'] := tblCreatures['CreatureName'] +'*';
      tblCreatures.Post;
      redQ2_2.Lines.Add(tblCreatures['CreatureName'] + #9 +
      tblCreatures['HabitatID']);
    end;
     tblCreatures.Next;
  end;
end;
//============================================================================
//  Question 2.2.2                                                 10 marks
//============================================================================
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  iCount: Integer;
begin
  // Provided code
  redQ2_2.Clear;
  redQ2_2.Lines.Add('Habitat name' + #9 + 'Number of creatures');
  redQ2_2.Lines.Add('');

  // Question 2.2.2
  tblHabitats.First;
  while not tblHabitats.EOF do
  begin
    iCount := 0;
    tblCreatures.First;
    while not tblCreatures.EOF do
    begin
      if tblHabitats['HabitatID'] = tblCreatures['HabitatID'] then
        Inc(iCount);
      tblCreatures.Next;
    end;
```

Downloaded from Stanmorephysics.com

```
    if iCount > 0 then
      redQ2_2.Lines.Add(tblHabitats['HabitatName'] + #9 +
                IntToStr(iCount));

    tblHabitats.Next;
  end;

end;

// ============================================================================
// Provided code
// ============================================================================
procedure TfrmQuestion2.btnResetClick(Sender: TObject);
begin
  objDB.RestoreDatabase;
  objDB.setupGrids(dbgOne, dbgMany, dbgSQL);

  redQ2_2.Clear;
end;

procedure TfrmQuestion2.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  objDB.DisconnectFromDB;
  objDB.destroy;
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  objDB := TDBConnect.create;
  tblCreatures := objDB.getManyTbl;
  tblHabitats := objDB.getOneTbl;

  dbgOne.DataSource := objDB.getOneDS;
  dbgMany.DataSource := objDB.getManyDS;

  objDB.setupGrids(dbgOne, dbgMany, dbgSQL);

  redQ2_2.Clear;
  redQ2_2.Paragraph.Tabcount := 2;
  redQ2_2.Paragraph.Tab[0] := 150;
  redQ2_2.Paragraph.Tab[1] := 200;
end;

procedure TfrmQuestion2.runSQL(sSQL: string);
begin
  dbgSQL.DataSource := objDB.ExecuteSQL(sSQL);
  objDB.setupGrids(dbgOne, dbgMany, dbgSQL);
end;
{$ENDREGION}

end.
```

Downloaded from Stanmorephysics.com

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class:

```
unit SeaTurtle_U;

interface

uses
  SysUtils, Math;

type
  TSeaTurtle = class
  private
    fTurtleID: String;
    fName: String;
    fSpecies: String;
    fAge: Integer;
    fInjurySeverity: Integer;
    fRehabDays: Integer;

  public
    Constructor Create(sTurtleID, sName, sSpecies: String; iAge: Integer);
    function generateTurtleID: String;
    function toString: String;
    procedure injuryObservation(sObservation: String);
    function getInjurySeverity: Integer;
    procedure setRehabDays(iDays: Integer);
    function estimateReleaseDate(dDate: TDate): TDate;
  end;
implementation

{ TSeaTurtle }
// ============================================================================
// Question 3.1.1                                                    5 marks
// ============================================================================
function TSeaTurtle.generateTurtleID: String;
begin
  Result := UpperCase(Copy(fSpecies, 1, 3) + copy(fName, length(fName)-1))
                + IntToStr(fAge);
end;
// ============================================================================
// Question 3.1.2                                                    4 marks
// ============================================================================
Constructor TSeaTurtle.Create(sTurtleID, sName, sSpecies: String;
                iAge: Integer);
begin
  fTurtleID := sTurtleID;
  fName := sName;
  fSpecies := sSpecies;
  fAge := iAge;
  fInjurySeverity := 5;
  fRehabDays := 60;
  if fTurtleID = 'None' then
    fTurtleID := generateTurtleID;
end;
```

Downloaded from Stanmorephysics.com

```
// ========================================================================
// Question 3.1.3                                                 6 marks
// ========================================================================
procedure TSeaTurtle.injuryObservation(sObservation: String);
begin
  if sObservation = 'Improving' then
  begin
      fInjurySeverity := fInjurySeverity - 2;
      if fInjurySeverity < 0 then
           fInjurySeverity := 1;
  end;
  if sObservation = 'Unstable' then
    fInjurySeverity := 10;
end;
// ========================================================================
// Question 3.1.4                                                 2 marks
// ========================================================================
function TSeaTurtle.getInjurySeverity: integer;
begin
  Result := fInjurySeverity;
end;
// ========================================================================
// Question 3.1.5                                                 3 marks
// ========================================================================
procedure TSeaTurtle.setRehabDays(iDays: Integer);
begin
  fRehabDays := iDays;
end;
// ========================================================================
// Question 3.1.6                                                 3 marks
// ========================================================================
function TSeaTurtle.estimateReleaseDate(dDate: TDate): TDate;
begin
  Result := dDate + fRehabDays;
end;
// ========================================================================
// Provided code
// ========================================================================
function TSeaTurtle.toString: String;
begin
  Result := fName + '#' + fTurtleID + #13 +
            'Species: ' + fSpecies + #13 +
            'Age: ' + IntToStr(fAge) + #13 +
            'Injury Severity: ' + IntToStr(fInjurySeverity) + #13 +
            'Rehabilitation duration: ' + IntToStr(fRehabDays);
end;

end.
```

Downloaded from Stanmorephysics.com

Main Form Unit:

```
unit Question3_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, ExtCtrls, StdCtrls, ComCtrls, jpeg, Spin, Populate_U,
SeaTurtle_U, Math;

type
  TfrmQuestion3 = class(TForm)
    gpbQ3_2_1: TGroupBox;
    lblTagID: TLabel;
    lblName: TLabel;
    cmbQ3_2_1: TComboBox;
    edtQ3_2_1: TEdit;
    rgpQ3_2_1: TRadioGroup;
    lblAge: TLabel;
    spnQ3_2_1: TSpinEdit;
    btnQ3_2_1: TButton;
    imgTurtle: TImage;
    gpb3_2_2: TGroupBox;
    btnQ3_2_2: TButton;
    gpbQ3_2_3: TGroupBox;
    btnQ3_2_3: TButton;
    redQ3_2: TRichEdit;
    rgpQ3_2_2: TRadioGroup;
    pnlQ3_2_3: TPanel;
    dtpQ3_2_3Current: TDateTimePicker;
    lblCurrentDate: TLabel;
    procedure cmbQ3_2_1Change(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure btnQ3_2_1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;
  objSeaTurtle: TSeaTurtle;

implementation

{$R *.dfm}
```

Downloaded from Stanmorephysics.com

```
// ============================================================================
// Question 3.2.1                                                    5 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
  sOneline, sName, sSpecies, sTurtleID: String;
  iPlace, iAge: Integer;
begin
  // Provided code
  sTurtleID := cmbQ3_2_1.Text;
  sName := edtQ3_2_1.Text;
  sSpecies := rgpQ3_2_1.Items[rgpQ3_2_1.ItemIndex];
  iAge := spnQ3_2_1.Value;
  redQ3_2.Clear;
  // Question 3.2.1
  objSeaTurtle := TSeaTurtle.Create(sTurtleID, sName, sSpecies, iAge);
  redQ3_2.Lines.Add(objSeaTurtle.toString);
end;
// ============================================================================
// Question 3.2.2                                                    9 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  sObservation: String;
begin
  // Provided code
  sObservation := rgpQ3_2_2.Items[rgpQ3_2_2.ItemIndex];
  // Question 3.2.2
  objSeaTurtle.injuryObservation(sObservation);
  case objSeaTurtle.getInjurySeverity of
    1 .. 4: objSeaTurtle.setRehabDays(30);
    5 .. 7: objSeaTurtle.setRehabDays(60);
    8 .. 10: objSeaTurtle.setRehabDays(120);
  end;

  redQ3_2.Lines.Clear;
  redQ3_2.Lines.Add(objSeaTurtle.toString)
end;
// ============================================================================
// Question 3.2.3                                                    3 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
var
  dCurrentDate: TDate;
begin
  // Provided code
  dCurrentDate := dtpQ3_2_3Current.Date;
  // Question 3.2.3
  pnlQ3_2_3.caption :=
DateToStr(objSeaTurtle.estimateReleaseDate(dCurrentDate));
end;
```

Downloaded from Stanmorephysics.com

```
// ============================================================================
// Provided code
// ============================================================================
{$REGION 'Provide code'}
procedure TfrmQuestion3.cmbQ3_2_1Change(Sender: TObject);
begin
  if cmbQ3_2_1.ItemIndex <> 0 then
  begin
    edtQ3_2_1.Text := Populate_U.arrNames[cmbQ3_2_1.ItemIndex];
    rgpQ3_2_1.ItemIndex := Populate_U.arrSpecies[cmbQ3_2_1.ItemIndex];
    spnQ3_2_1.Value := Populate_U.arrAges[cmbQ3_2_1.ItemIndex];
  end
  else
  begin
    edtQ3_2_1.Text := '';
    rgpQ3_2_1.ItemIndex := -1;
    spnQ3_2_1.Value := 0;
  end
end;

procedure TfrmQuestion3.FormShow(Sender: TObject);
begin
  dtpQ3_2_3Current.Date := DATE;
end;
{$ENDREGION}
end.
```

Downloaded from Stanmorephysics.com

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, ComCtrls, jpeg, StdCtrls, Math, Grids;
type
  TfrmQuestion4 = class(TForm)
    pgcQ4: TPageControl;
    tsQ4_1: TTabSheet;
    tsQ4_2: TTabSheet;
    imgFishing: TImage;
    pnlQ4_1: TPanel;
    redQ4_1: TRichEdit;
    btnQ4_1_1: TButton;
    btnQ4_1_2: TButton;
    btnQ4_2: TButton;
    sgdQ4_2: TStringGrid;
    procedure btnQ4_2Click(Sender: TObject);
    procedure btnQ4_1_1Click(Sender: TObject);
    procedure btnQ4_1_2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure displayInfo;
  private
    { Private declarations }
    procedure display2D;
  public
    { Public declarations }
  end;
var
  frmQuestion4: TfrmQuestion4;
  arrSurnameName: array [1 .. 30] of String;
  arrNumFish: array [1 .. 30] of Integer;
  arrTotalWeight: array [1 .. 30] of Real;
  iNumEntries: Integer = 0;

  arrPattern: array [1 .. 7, 1 .. 15] of Integer =
   ((0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 1, 0, 1,
      0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
    (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
    (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0));
implementation
{$R *.dfm}
{ TfrmQuestion4 }

// ============================================================================
// Question 4.1.1                                                    14 marks
// ============================================================================
procedure TfrmQuestion4.btnQ4_1_1Click(Sender: TObject);
var
  tFile: Textfile;
  sLine, sTemp: String;
  iNumFish: Integer;
  rTotalWeight: Real;
begin
  //Question 4.1.1
```

Downloaded from Stanmorephysics.com

```
  AssignFile(tFile, 'Fishing.txt');
  Reset(tFile);
  while NOT Eof(tFile) do
  begin
    Inc(iNumEntries);
    Readln(tFile, sTemp);
    Readln(tFile, arrSurnameName[iNumEntries]);
    Readln(tFile, sLine);
    iNumFish := 0;
    rTotalWeight := 0;
    while sLine <> '' do
    begin
      inc(iNumFish);
      rTotalWeight := rTotalWeight + StrToFloat(sLine);
      Readln(tFile, sLine);
    end;
   arrNumFish[iNumEntries] := iNumFish;
   arrTotalWeight[iNumEntries] := rTotalWeight;
  end;
  CloseFile(tFile);

  //Provided code
  displayInfo;
end;
// ============================================================================
// Question 4.1.2                                                    10 marks
// ============================================================================
procedure TfrmQuestion4.btnQ4_1_2Click(Sender: TObject);
var
  rHighestAvg, rAvg: Real;
  iCnt, iNumFish: Integer;
  sOutput: String;
begin
  //Provided code
  redQ4_1.Clear;
  rHighestAvg := 0;

  //Question 4.1.2
  for iCnt := 1 to iNumEntries do
  begin
    if arrNumFish[iCnt] > 0 then
    begin
      rAvg := arrTotalWeight[iCnt] / arrNumFish[iCnt];

      if rAvg = rHighestAvg then
        sOutput := sOutput + arrSurnameName[iCnt] + #13;

      if rAvg > rHighestAvg then
      begin
        sOutput := arrSurnameName[iCnt] + #13;
        rHighestAvg := rAvg;
      end;
    end;
  end;
  redQ4_1.Lines.Add(sOutput);

 //Provided code
  redQ4_1.Lines.Add('Highest average: ' +
                      FloatToStrF(rHighestAvg,ffFixed,8,2));
end;
```

Downloaded from Stanmorephysics.com

```
// ============================================================================
// Question 4.2                                                      6 marks
// ============================================================================
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  iRow, iCol: Integer;
begin
  //Question 4.2

  for iRow := 3 to 7 do
    for iCol := 2 to 14 do
      arrPattern[iRow, iCol] := arrPattern[iRow - 1, iCol - 1] +
                                arrPattern[iRow - 1, iCol + 1];
// Provided code
  display2D;
end;

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

{$REGION 'Provided code - Do NOT change'}

procedure TfrmQuestion4.display2D;
var
  iRow, iCol: Integer;
  iLeftPad, iRightPad: Integer;
begin
  for iRow := 0 to 6 do
    for iCol := 0 to 14 do
      if arrPattern[iRow + 1, iCol + 1] = 0 then
        sgdQ4_2.cells[iCol, iRow] := ''
      else
        if Length(IntToStr(arrPattern[iRow+1, iCol+1])) = 1 then
          sgdQ4_2.Cells[iCol,iRow] := '  ' +
                          IntToStr(arrPattern[iRow+1, iCol+1])
        else
          sgdQ4_2.Cells[iCol,iRow]:=' '+ IntToStr(arrPattern[iRow+1, iCol+1]);
end;

procedure TfrmQuestion4.displayInfo;
var
  iCnt : integer;
begin
   redQ4_1.Clear;
   redQ4_1.Paragraph.TabCount := 3;
   redQ4_1.Paragraph.Tab[0] := 50;
   redQ4_1.Paragraph.Tab[1] := 150;
       redQ4_1.Lines.Add('Surname and Name' + #9 + 'Number of fish' + #9 +
                          'Total weight');
   for iCnt := 1 to 26 do
      redQ4_1.Lines.Add(arrSurnameName[iCnt] + #9 + IntToStr(arrNumFish[iCnt])
                          + #9#9#9 + FloatToStr(arrTotalWeight[iCnt]));
end;
```

Downloaded from Stanmorephysics.com

```
procedure TfrmQuestion4.FormCreate(Sender: TObject);
var
  iRow, iCol: Integer;
  iLeftPad, iRightPad: Integer;
begin
    for iRow := 0 to 1 do
    for iCol := 0 to 14 do
      if arrPattern[iRow + 1, iCol + 1] = 0 then
        sgdQ4_2.cells[iCol, iRow] := ''
      else
        if Length(IntToStr(arrPattern[iRow+1, iCol+1])) = 1 then
          sgdQ4_2.Cells[iCol,iRow] := '  ' +
                          IntToStr(arrPattern[iRow+1, iCol+1])
        else
          sgdQ4_2.Cells[iCol,iRow] := ' ' +
                          IntToStr(arrPattern[iRow+1, iCol+1]);
end;

{$ENDREGION}

end.
```

Copyright reserved