basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2010

MARKS: 120

TIME: 3 hours

This question paper consists of 34 pages, 3 annexures and an information sheet.

INSTRUCTIONS AND INFORMATION

1. This is a three-hour examination. 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.

2. Answer SECTION A (for Delphi programmers) OR SECTION B (for Java programmers).

3. You require the files listed below in order to answer the questions. They are EITHER on a stiffy disk OR CD issued to you, OR the invigilator/teacher will tell you where to find them on the hard drive of the workstation you are using OR in a network folder:

QUESTION 1

Delphi:	**Java:**
Question1_P.dpr	Sightings.class
Question1_P.res	SightingsDB.mdb
Question1_U.dfm	tblRangers.txt
Question1_U.pas	tblSightings.txt
SightingsDB.mdb	TestSightings.java
tblRangers.txt	
tblSightings.txt	

QUESTION 2

Delphi:	**Java:**
Question2_P.dpr	Competitor.java
Question2_P.res	Sightings.txt
Question2_U.dfm	TestCompetitor.java
Question2_U.pas	
uCompetitor.pas	
Sightings.txt	

QUESTION 3:

Delphi:	**Java:**
Question3_P.dpr	TestDistances.java
Question3_P.res	
Question3_U.dfm	
Question3_U.pas	

If a disk (CD or stiffy) containing the above files was issued to you, write your examination number on the label.

4. Save your work at regular intervals as a precaution against power failures.

5. Save ALL your solutions in folders with the number of the question and your examination number as the name of the folder, for example Quest2_3020160012.

6. Type in your examination number as a comment in the first line of each program.

7. Read ALL the questions carefully. Do not do more than the questions require.

8. During the examination you may use the manuals originally supplied with the hardware and software. You may also use the HELP functions of the software. **Java candidates may make use of the Java API files. You may NOT use any other resource material.**

9. At the end of this examination session you will be required to hand in the disk with your work saved on it OR you must make sure that all your work has been saved on the hard drive/network as explained to you by the invigilator/teacher. Ensure that all files can be read.

10. You also have to hand in printouts of the programming code for all the questions that you did.

11. All printing of programming questions will take place within an hour of the completion of the examination.

12. Complete the information sheet attached to this question paper and hand it in at the end of this examination session.

SECTION A

Answer ALL the questions in this section only if you studied **Delphi**.

SCENARIO:

The Big Five Game Park is a new South African game park which protects a variety of South African wildlife. The game park's priorities are research, nature conservation, animal monitoring and public awareness. They require custom-developed computer software that will assist in performing some everyday tasks.

QUESTION 1: DELPHI PROGRAMMING AND DATABASE

The chief park ranger requires a program that will enable him/her to store personal information on the rangers (staff), as well as details of the sightings of animals being monitored. This information will assist visitors to the park to view details on animals on a daily basis. The program will also assist in monitoring the work carried out by the different rangers on different days. A database called **SightingsDB** has been developed. An incomplete program has been developed to process queries on the information in the given database. Your task will be to complete this program.

NOTE: The design of the tables in the **SightingsDB** database and sample data for this question can be found in **ANNEXURE A: Table Description Sheet**.

NOTE: If you cannot use the database in the provided format, follow the instructions in **ANNEXURE B** to create the database before you answer any of QUESTIONS 1.1 to 1.7.

NOTE: Make a copy of the **SightingsDB** database BEFORE you start with the solution. You will need a copy of the original database to be able to test your program thoroughly.

You have also been supplied with an incomplete Delphi project named **Question1_P.dpr** in the folder named **Question 1 Delphi**.

Do the following:

- Rename the folder **Question 1 Delphi** to **Quest1_X**, where X should be replaced with your examination number.
- Open Delphi and then open the file **Question1_P.dpr** in the folder **Quest1_X**. The program displays eight buttons as well as a DBGrid that will be used as an output component (see example on next page).
- Add your examination number to the right of 'Question 1 –' in the caption of the form.
- Go to File/Save As … and save the unit as **Question1_UXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Go to File/Save Project As … and save the project as **Question1_PXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).

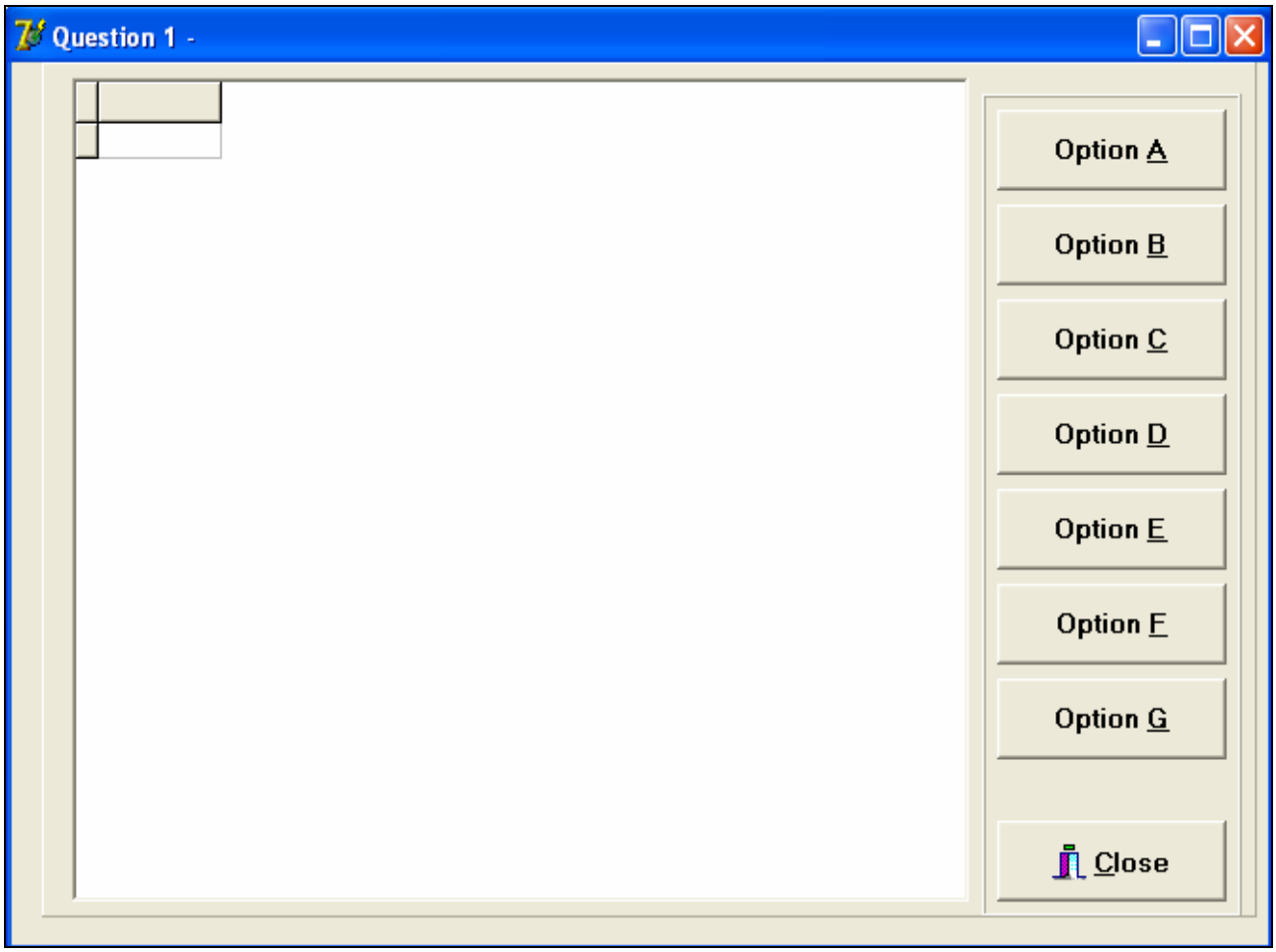
- The program should be able to connect to the database named **SightingsDB**. When you do QUESTION 1.1 (which follows below) and you find that the connectivity is not in place, use the steps supplied in **ANNEXURE C** to establish connection with the database.

HINT: If your program cannot connect to the database, ensure that the database file **SightingsDB** is in the same folder as your program. Your program will not work if the database file is in a folder other than the folder containing your Delphi program. If this is the case, copy the database file **SightingsDB** into the same folder as your program.

NOTE: If you cannot establish connectivity with the database at all when you execute the program, you must still do the SQL code and submit it for marking.

Marks will only be awarded for the code that contain the SQL statements in the unit named Question1_UXXXX.

When you execute the program, the interface below will be displayed. When the buttons are clicked, an error will be displayed due to the incomplete SQL statements.



Do the following:

Complete the SQL statements in **Question1_UXXXX.pas** for each menu option as indicated in QUESTIONS 1.1 to 1.7 below. The code to execute the SQL statements and display the results on the DBGrid has been given to you.

1.1 The chief park ranger would like to view all details of animals sighted on a daily basis. Complete the code for the **Option A** button by formulating an SQL statement to display **all details** of the sightings stored in the **tblSightings** table. Display the output in descending order of **SightingID**.

Example of output for the first five sightings:

SightingID	SightingDate	Animal	NumAnimals	Young	RangerID
200	2010/05/07	Kudu	16	False	9
199	2010/05/31	Kudu	9	True	11
198	2010/04/04	Impala	21	True	4
197	2010/07/29	Cheetah	2	True	4
196	2010/01/19	Kudu	4	True	12

:

NOTE: The date on your output may be in a different format, depending on the settings on your computer. Any format of the date will be acceptable. (4)

1.2 A visiting international student is carrying out a survey and requires information, particularly about the different types of young animals that have been sighted at the park. Complete the code for the **Option B** button by formulating an SQL statement to display **only the names** of the different types of young animals that have been sighted.

NOTE: The name of each type of young animal should be displayed once only.

Example of output:

Animal
Aardvark
Cheetah
Elephant
Giraffe
Impala
Kudu
Lion
Rhino

(4)

1.3 A record needs to be kept to determine the number of years each ranger has been employed at the park. Complete the code for the **Option C** button by formulating an SQL statement to display the **RangerID**, **Name**, **Surname** and the number of years the ranger has been employed. Store the calculated field in **TotalYears**.

Example of output for the first five rangers:

RangerID	Name	Surname	TotalYears
1	Jada	Harrison	8
2	Kenyon	Carney	3
3	Dylan	Pollard	5
4	Sylvester	Walls	7
5	Urielle	Wynn	3

:

(6)

1.4 At the end of every month the chief ranger is required to print a report to show the average number of each type of animal sighted, using all listings in the **tblSightings** table. Complete the code for the **Option D** button by formulating an SQL statement that will display the **animal** name and the average number sighted, rounded off to TWO decimal points (name this field **AvgSighted**). The output must be grouped according to the animal field.

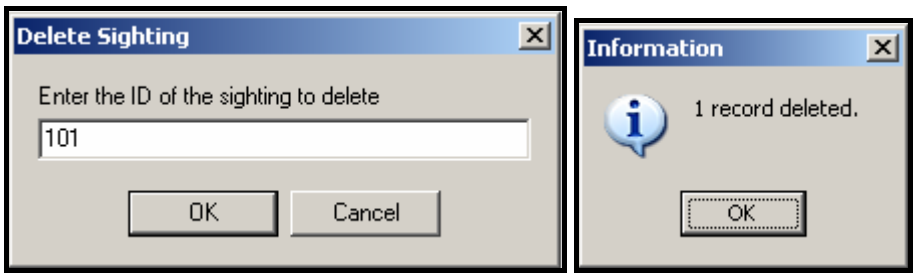
Example of output:

Animal	AvgSighted
Aardvark	13.00
Cheetah	17.46
Elephant	18.15
Giraffe	15.25
Impala	16.92
Kudu	16.18
Lion	15.54
Rhino	16.25

(6)

1.5 The results of sightings during poor weather conditions can sometimes be inaccurate. The park administrator is required to delete inaccurate sightings as they come up. Complete the code for the **Option E** button by allowing the user to enter the **SightingID** of a sighting and then formulate an SQL statement that will **delete** the record of the corresponding sighting.

Example of output:



HINT: Run **Option A** to verify that the record has been deleted. (4)

1.6 All rhinos sighted in the park happen to be white rhinos. Complete the code for the **Option F** button by formulating an SQL statement that will update the **animal** field to 'White Rhino' if the animal field contains the word 'Rhino'.

Example of output:

HINT: Run **Option A** to verify that the records have been updated. (5)

1.7 Information is needed regarding the details of rangers who sighted elephants after a given date. Complete the code for the **Option G** button by formulating an SQL statement to display the **SightingDate**, **Name** and **Surname** of the rangers that sighted elephants after **30/04/2010.**

Example of output:

SightingDate	Name	Surname
2010/05/29	Ivory	Frost
2010/05/31	Karleigh	Jones
2010/07/06	Jada	Harrison
2010/06/08	Odessa	Head

NOTE: The date on your output may be in a different format, depending on the settings on your computer. Any format of the date will be acceptable. (6)

- Enter your examination number as a comment in the first line of the file named **Question1_UXXXX.pas** containing the SQL statements.
- Save the unit **Question1_UXXXX** and the project **Question1_PXXXX** (File/Save All).
- A printout for the code of the **Question1_UXXXX.pas** file will be required. **[35]**

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

The Big Five Game Park wants to launch a competition for the public. The competition will require each person to write down the type of animals that he/she sees (regardless of duplicates), during a day at the park. For the competition, the park divides animals into three categories, **large game**, **small game** and **birds**.

Each competitor will record his/her name and then his/her list of animals together with each animal's category.

Each competitor submits a text file with his/her sightings.

Each competitor's animal sightings will be processed to obtain the total number of animals sighted in each of the three categories. The park's management will award points to each competitor based on the animals they have sighted.

You are required to write a program (as indicated in QUESTION 2.1 and QUESTION 2.2) to process ONE such competitor text file.

The data stored in the text file named **Sightings.txt** in the folder **Question 2 Delphi** contains the information on the **sightings for a single competitor**. The format of the data in the file is as follows:

Name of competitor
Animal(Letter)
Animal(Letter)
etc.

where **Letter** represents one of the three categories of animals as defined above (**L** for large game, **S** for small game and **B** for birds).

Only these three categories (**L, S, B**) are valid. Any other category referred to in the text file, such as (**M**) for medium game, will be regarded as an invalid entry.

An example of the data in the text file:

Jane
Elephant(L)
Rhino(L)
Rhino(L)
Crocodile(M)
Eagle(B)
Owl(B)
Warthog(S)
Meerkat(S)
etc.

Do the following:

- Rename the folder **Question 2 Delphi** as **Quest2_X** (where X must be replaced by your examination number).
- Open Delphi and then open the file **Question2_P.dpr** in the folder **Quest2_X**.
- Go to File/Save As … and save the unit as **Question2_UXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Open the unit **uCompetitor.pas**.
- Go to File/Save As … and save the unit as **uCompetitorXXXX.pas** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Go to File/Save Project As … and save the project as **Question2_PXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).

2.1 The object class named **uCompetitorXXXX.pas** will represent the results for a single competitor, **TCompetitor**, including his/her name and how many animals in each category he/she sighted. Note the following:

- All fields in this class should be private and all methods public.
- Parts of this class have been inserted as comments in order to get the class to compile.
- In addition to modifying the given methods, you will be required to add code for new methods as described below.

Do the following in the **uCompetitorXXXX.pas** file:

2.1.1 Create private fields **with the following names** to hold the data. You should choose appropriate data types for these fields:

- **name** – name of competitor
- **largeGameCount** – total number of large game animals sighted
- **smallGameCount** – total number of small game animals sighted
- **birdCount** – total number of birds sighted

It is important that you use the field names given in bold above in order for the given code to operate correctly. (3)

2.1.2 (a) You have been provided with a **default constructor method**. Write an **additional constructor method** that has one parameter for the competitor's name. Initialise the name field using the parameter value and initialise the other fields to zero. (2)

(b) You have been provided with three methods named **spotLarge**, **spotSmall** and **spotBird**. Remove the comment symbols from the code provided in these methods. (1)

2.1.3 You have been provided with an incomplete method (function method) with a defined return data type called **calculatePoints** that should return the total number of points for a competitor. The points are awarded as follows:

- Five points for each large game animal sighted
- Three points for each small game animal sighted
- Two points for each bird sighted

The method contains an incomplete statement that has been blocked out as a comment. The total points are supposed to be calculated by calculating the sum of the number of large game sighted multiplied by five, the number of small game sighted multiplied by three and the number of birds sighted multiplied by two.

Remove the comment symbols from the given statement and complete the code so that the method returns the correct result. (3)

2.1.4 Write a method (function method) called **totalAnimals** which returns the total number of animals sighted as an integer. The total is calculated by calculating the sum of the number of large game, small game and birds sighted. (2)

2.1.5 Write a 'get' method called **getName** to return the name of the competitor. (2)

2.1.6 Write a method (function method) called **mostSpotted** that will determine and return the category of animal ("Large Game", "Small Game" or "Bird") that the competitor spotted the most. (4)

2.1.7 You have been provided with a method (function method) called **toString** that constructs and returns a string with the name and sighting results of a competitor. However, the code provided in the method is incomplete and has been commented out.

Remove the comment symbols so that the given statements will execute and complete the code so that it returns the information in the following format:

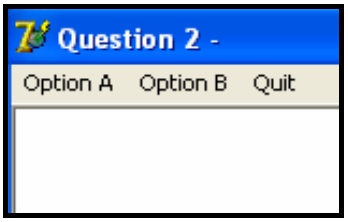
```
Competitor : name
Large : largeGameCount, Small : smallGameCount, Bird : birdCount
Total Animals : <tab>totalAnimals
```

Example of the output when the returned string is displayed:

```
Competitor : Jane
Large : 9  Small : 6  Bird : 7
Total Animals :      22
```

(5)

2.2 In the **Question2_UXXXX.pas** file (the main unit) you have been given code to display the following menu when the program is executed:



Open the **Question2_UXXXX.pas** file (the main unit) and do the following:

- Add your examination number to the caption of the form to the right of 'Question 2 –'.
- Write code to do the following in the **Question2_UXXXX.pas** file (the main unit) in the given program:

2.2.1 Write code in the **OnActivate** event handler of the form to read information from the text file **Sightings.txt** according to the following steps:

(a) Test if the text file exists. Display a suitable message if the file does not exist and terminate the program. Continue with the remainder of the steps if the file exists.

(b) Read the first line from the text file and store this as the competitor's name.

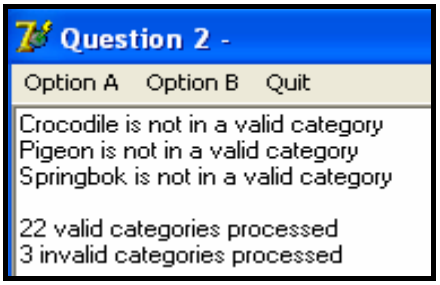
(c) Using the competitor's name, create a **single object** of type **TCompetitor**. You do not need to create an array of these objects as **only one competitor will be processed** each time the program is run.

(d) Use a loop to do the following:

- Read a line of text (one animal) from the text file.
- Test if the category of the animal that appears in brackets after the name of the animal is valid or not. Only the letters L (for large game), S (for small game) and B (for birds) are valid. Display an appropriate message which includes the name of the animal for an invalid category.
- If the category is valid, then call one of the following methods from your **TCompetitor** class: **spotLarge**, **spotSmall**, **spotBird**. These methods increase the number of large game, small game and bird sightings respectively each time the method is called.

(e) Use two counter variables to keep track of how many valid and invalid categories were recorded.

(f) Display the total number of invalid and valid categories recorded as shown below:

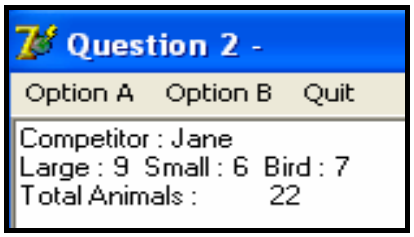


(20)

2.2.2 Menu Option A

When the user selects this menu option, the program must display the name and results for the competitor, by using the **toString** method from the **TCompetitor** class.

Example of output:



(2)

2.2.3 Menu Option B

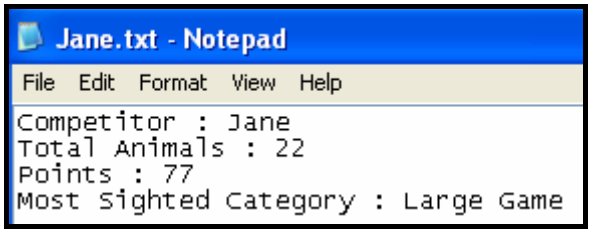
When the user selects this menu option the program must do the following:

- Create a new text file to save the name and results for the competitor. Construct a name for the text file that will contain the name of the competitor.

NOTE: Do not hardcode 'Jane.txt' as the file name since the name of the file should vary according to the name of the competitor.

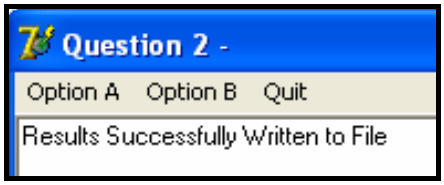
- Call the appropriate methods from the **TCompetitor** class and save the information to the created file.

Example of the contents of the text file:



- Display a message indicating that the information was successfully written to the file.

Example of output:



(5)

- Make sure that your examination number is a comment in the first line of the main class **Question2_UXXXX.pas**, as well as the object class **uCompetitorXXXX.pas**.
- Save all the files (File/Save All).
- Printouts of the code for the classes **Question2_UXXXX.pas** and **uCompetitorXXXX.pas** will be required.

[49]

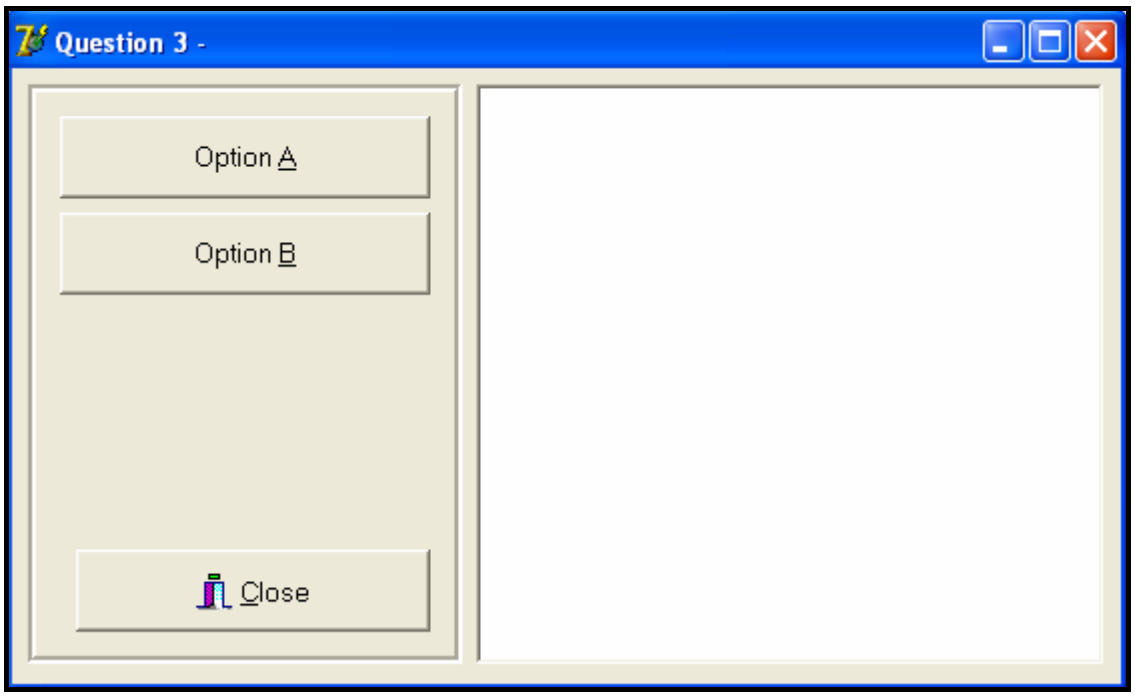
QUESTION 3: DELPHI – PROGRAMMING

The park rangers are doing some research on the movement of a **specific group of animals** relative to a **watering hole** situated at a specific location. They have monitored and recorded the locations of this group of animals over a specific period of time.

You have been given an incomplete program in the folder named **Question 3 Delphi**.

Do the following:

- Rename the folder named **Question 3 Delphi** to **Quest3_X** (where X should be replaced with your examination number).
- Open the Delphi program in this folder.
- Save the unit as ('File/Save As') **Question3_UXXXX** and the project as ('File/Save Project As') **Question3_PXXXX** inside the folder (XXXX should be replaced by the last FOUR digits of your examination number).
- Add your examination number to the caption of the form to the right of 'Question 3 –'.
- Execute the program. A menu with the following options will be displayed:



3.1 Do the following:

- Declare an array called **arrEntries** that must contain a maximum of 12 strings.
- Remove the comment symbols from the code supplied to assign strings to the **arrEntries** array.
- Use the array appropriately in your program to answer the questions that follow.

Each array entry consists of a string indicating the coordinates of the location of the group of animals relative to the watering hole at a specific time. Each string has the following format:

X-Coordinate,Y-Coordinate:Time recorded

Example of the first five strings assigned to the array:

```
arrEntries[1]  := '12,15:02h00';
arrEntries[2]  := '13,10:05h00';
arrEntries[3]  := '9,20:06h00';
arrEntries[4]  := '10,15:09h00';
arrEntries[5]  := '7,8:10h00';
```

NOTE:
In the first entry:
- The x-coordinate is 12
- The y-coordinate is 15
- The time is 02h00

(3)

3.2 The coordinates of the watering hole are as follows:

X-coordinate = 10
Y-coordinate = 10

Complete the code for the **Option A** button as follows:

- Declare TWO variables to store the coordinates of the watering hole and assign values to the variables.
- For each recording in the **arrEntries** array, extract the x- and y-coordinates and the time recorded. Use the x- and y- coordinates to calculate the distance between the location of the animals and the location of the watering hole. The distance should be an integer which has been rounded off. Use the following formula to calculate the distance:

$$\text{Distance} = \sqrt{(\text{AnimalXpos} - \text{WaterXpos})^2 + (\text{AnimalYpos} - \text{WaterYpos})^2}$$

where

AnimalXpos represents the x-coordinate of where the animal was spotted
AnimalYpos represents the y-coordinate of where the animal was spotted
WaterXpos represents the x-coordinate of where the watering hole is situated
WaterYpos represents the y-coordinate of where the watering hole is situated

- For each entry in the **arrEntries** array display the time, the distance from the watering hole and the x- and y-coordinates. Use a suitable heading and subheadings.

Example of output (on the next page):

Distances from the watering hole			
Time	Distance(km)	X-pos	Y-pos
02h00	5	12	15
05h00	3	13	10
06h00	10	9	20
09h00	5	10	15
10h00	4	7	8
11h00	0	10	10
14h00	8	12	18
17h00	9	7	18
19h00	3	11	7
20h00	0	10	10
23h00	12	2	1
24h00	7	12	17

(17)

3.3 The rangers had difficulty in tracking the animals. To facilitate the tracking of animals, it has been decided to create a tag for each animal in order to locate them more easily.

You are required to write code for the **Option B** button that will allow you to do the following:

- Enter the number of different types of animals in a mixed group of animals.
- Enter each type of animal in the group followed by the number of that type of animal in the group.

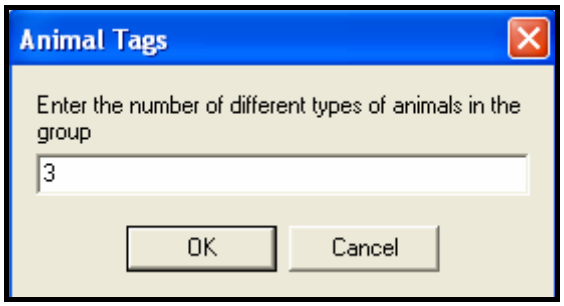
Your code must generate a tag for each of the animals in the group. The tag is generated as follows:

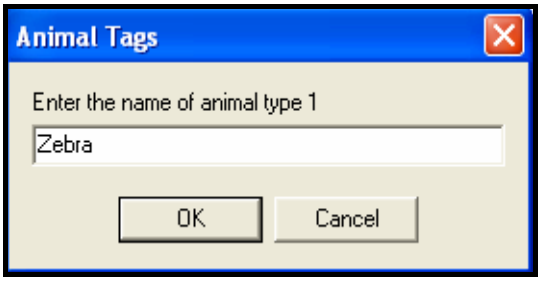
- Extract the first two letters from the name of the type of animal, for example "Rh" will be extracted from Rhino.
- Extract the last letter from the name of the type of animal, for example "o" will be extracted from Rhino.
- Randomly generate a three-digit even number per animal type.
- Combine the extracted letters and the number generated to form the first part of the tag.
- For each animal of a specific type, add a hyphen and a unique number beginning at 1 for the first animal of a specific type to the tag as indicated in the sample output.

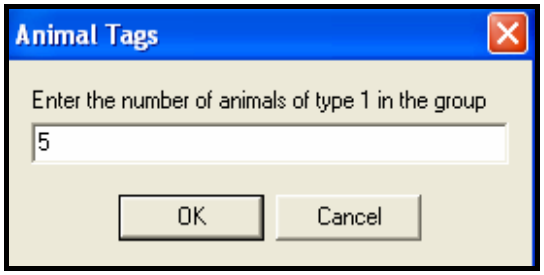
You are required to display the name of the type of animal as part of a heading and a numbered list with a tag number for each animal of the specific type.

Example of input and output for a group of animals consisting of three different types of animals, for example five zebra, four black wildebeest and ten impala (on the next page):

Input:



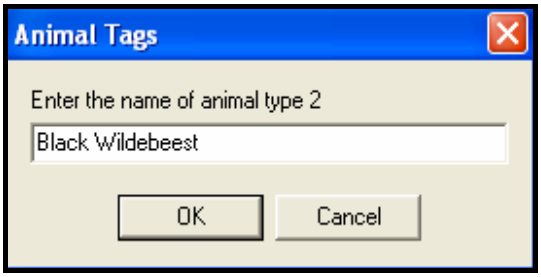


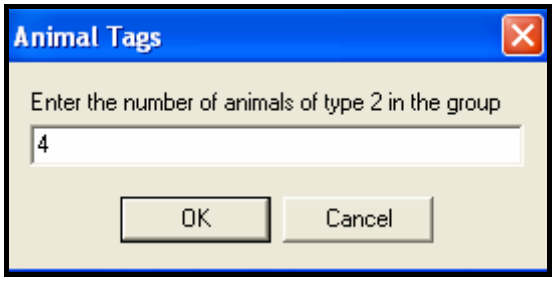


Output:

Zebra	Tag number
1.	Zea782-1
2.	Zea782-2
3.	Zea782-3
4.	Zea782-4
5.	Zea782-5

Input:

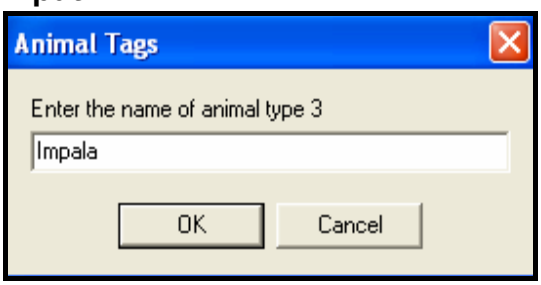


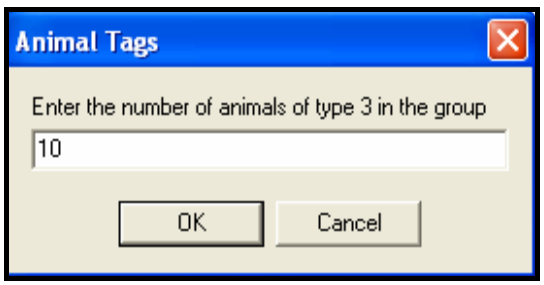


Output:

Black Wildebeest	Tag number
1.	Blt548-1
2.	Blt548-2
3.	Blt548-3
4.	Blt548-4

Input:





Output:

Impala	Tag number
1.	Ima658-1
2.	Ima658-2
3.	Ima658-3
4.	Ima658-4
5.	Ima658-5
6.	Ima658-6
7.	Ima658-7
8.	Ima658-8
9.	Ima658-9
10.	Ima658-10

NOTE: Different random numbers will be generated for each execution of the program. (16)

- Enter your examination number as a comment in the first line of the unit **Question3_UXXXX**, as well as any other unit(s) you may have created.
- Save the unit(s) and the project ('File/Save All').
- A printout of the code for the unit **Question3_UXXXX**, as well as any other unit(s) you may have created, will be required. **[36]**

TOTAL SECTION A: 120

SECTION B

Answer ALL the questions in this section only if you studied **Java**.

SCENARIO:

The Big Five Game Park is a new South African game park which protects a variety of South African wildlife. The game park's priorities are research, nature conservation, animal monitoring and public awareness. They require custom-developed computer software that will assist in performing some everyday tasks.

QUESTION 1: JAVA PROGRAMMING AND DATABASE

The chief park ranger requires a program that will enable him/her to store personal information on the rangers (staff), as well as details of the sightings of animals being monitored. This information will assist visitors to the park to view details on animals on a daily basis. The program will also assist in monitoring the work carried out by the different rangers on different days. A database called **SightingsDB** has been developed. An incomplete program has been developed to process queries on the information in the given database. Your task will be to complete this program.

NOTE: The design of the tables in the **SightingsDB** database and sample data for this question can be found in **ANNEXURE A: Table Description Sheet**.

NOTE: If you cannot use the database in the provided format, follow the instructions in **ANNEXURE B** to create the database before you answer any of QUESTIONS 1.1 to 1.7.

NOTE: Make a copy of the **SightingsDB** database BEFORE you start with the solution. You will need a copy of the original database to be able to test your program thoroughly.

You have also been supplied with an incomplete Java program, in the folder named **Question 1 Java**, with a test class named **TestSightings.java** and an object class named **Sightings.class** which will display the results of the queries.

Do the following:

- Rename the folder **Question 1 Java** as **Quest1_X**, where X should be replaced with your examination number.
- Rename the **TestSightings.java** file in the folder **Quest1_X** to **TestSightingsXXXX.java** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Open the **TestSightingsXXXX.java** file.
- Change the name of the class to **TestSightingsXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).

NOTE: If you compile and run the **TestSightingsXXXX.java** file, the following menu will be displayed (see on the next page). However, if you enter any of the options A to G, the program will not work because of the incomplete SQL statements.

```
     MENU

  Option A
  Option B
  Option C
  Option D
  Option E
  Option F
  Option G

  Q - QUIT
```

The connectivity code, as well as the code to display the results, are contained in the file named **Sightings.class**.

HINT: If your program cannot connect to the database, ensure that the database file **SightingsDB** is in the same folder as the files **TestSightingsXXXX.java** and **Sightings.class**. Your program will not work if the database file is in a folder other than the folder containing your Java program. If this is the case, copy the database file **SightingsDB** into the same folder as your program.

NOTE: If you cannot establish connectivity with the database at all when you execute the program, you must still do the SQL code and submit it for marking.

Marks will only be awarded for the programming code which contains the SQL statements in the program named TestSightingsXXXX.java.

Do the following:

Complete the SQL statements in **TestSightingsXXXX.java** for each menu option as indicated in QUESTIONS 1.1 to 1.7 below. The code to pass the SQL statements to the relevant methods in the **Sightings.class** file has already been programmed for you.

1.1 The chief park ranger would like to view all details of animals sighted on a daily basis. Complete the code for menu **Option A** by formulating an SQL statement to display **all details** of the sightings stored in the **tblSightings** table. Display the output in descending order according to **SightingID**.

Example of output of the first five sightings:

SightingID	SightingDate	Animal	NumAnimals	Young	RangerID
200	2010-05-07	Kudu	16	False	9
199	2010-05-31	Kudu	9	True	11
198	2010-04-04	Impala	21	True	4
197	2010-07-29	Cheetah	2	True	4
196	2010-01-19	Kudu	4	True	12
:					

NOTE: The date on your output may be in a different format, depending on the settings on your computer. Any format of the date will be acceptable. (4)

1.2 A visiting international student is carrying out a survey and requires information particularly about the different types of young animals that have been sighted at the park. Complete the code for menu **Option B** by formulating an SQL statement to display **only the names** of the different types of young animals that have been sighted.

NOTE: The name of each type of young animal should be displayed once only.

Example of output:

```
Animal
==========
Aardvark
Cheetah
Elephant
Giraffe
Impala
Kudu
Lion
Rhino
```

(4)

1.3 A record needs to be kept to determine the number of years each ranger has been employed at the park. Complete the code for menu **Option C** by formulating an SQL statement to display the **RangerID**, **Name**, **Surname** and the number of years the ranger has been employed. Store the calculated field in **TotalYears**.

Example of output for the first five rangers:

```
RangerID        Name                Surname             TotalYears
=================================================================
1               Jada                Harrison            8
2               Kenyon              Carney              3
3               Dylan               Pollard             5
4               Sylvester           Walls               7
5               Urielle             Wynn                3
:
```

(6)

1.4 At the end of every month, the chief ranger is required to print a report to show the average number of each type of animal sighted, using all listings in the **tblSightings** table. Complete the code for menu **Option D** by formulating an SQL statement that will display the **animal** name and the average number sighted, rounded off to TWO decimal points (name this field **AvgSighted**). The output must be grouped according to the animal field.

Example of output (on the next page):

Animal	AvgSighted
Aardvark	13.0
Cheetah	17.46
Elephant	18.15
Giraffe	15.25
Impala	16.92
Kudu	16.18
Lion	15.54
Rhino	16.25

(6)

1.5 The results of sightings during poor weather conditions can sometimes be inaccurate. The park administrator is required to delete inaccurate sightings as they come up. Complete the code for menu **Option E** by allowing the user to enter the **SightingID** of a sighting and then formulate an SQL statement that will **delete** the record of the corresponding sighting.

Example of output:

```
Enter the ID of the sighting to delete
101

1 record deleted
```

HINT: Run **Option A** to verify that the record has been deleted. (4)

1.6 All rhinos sighted in the park happen to be white rhinos. Complete the code for menu **Option F** by formulating an SQL statement that will update the **animal field** to 'White Rhino' if the animal field contains the word 'Rhino'.

Example of output:

```
12 records updated
```

HINT: Run **Option A** to verify that the records have been updated. (5)

1.7 Information is needed regarding the details of rangers who sighted elephants after a given date. Complete the code for menu **Option G** by formulating an SQL statement to display the **SightingDate**, **Name** and **Surname** of the rangers that sighted elephants after **30/04/2010**.

Example of output:

SightingDate	Name	Surname
2010-05-29	Ivory	Frost
2010-05-31	Karleigh	Jones
2010-07-06	Jada	Harrison
2010-06-08	Odessa	Head

NOTE: The date on your output may be in a different format, depending on the settings on your computer. Any format of the date will be acceptable. (6)

- Enter your examination number as a comment in the first line of the file named **TestSightingsXXXX.java** containing the SQL statements.
- Save the file **TestSightingsXXXX.java**.
- A printout for the code of the **TestSightingsXXXX.java** file will be required. **[35]**

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING

The Big Five Game Park wants to launch a competition for the public. The competition will require each person to write down the type of animals that he/she sees (regardless of duplicates), during a day at the park. For the competition, the park divides animals into three categories, **large game**, **small game** and **birds**.

Each competitor will record his/her name and then his/her list of animals together with each animal's category.

Each competitor submits a text file with his/her sightings.

Each competitor's animal sightings will be processed to obtain the total number of animals sighted in each of the three categories. The park's management will award points to each competitor based on the animals they have sighted.

You are required to write a program (as indicated in QUESTION 2.1 and QUESTION 2.2) to process ONE such competitor text file.

The data stored in the text file named **Sightings.txt** in the folder **Question 2 Java** contains the information on the **sightings for a single competitor**. The format of the data in the file is as follows:

Name of competitor
Animal(Letter)
Animal(Letter)
etc.

where **Letter** represents one of the three categories of animals as defined above (**L** for large game, **S** for small game and **B** for birds).

Only these three categories (**L, S, B**) are valid. Any other category referred to in the text file, such as (**M**) for medium game, will be regarded as an invalid entry.

An example of the data in the text file:

Jane
Elephant(L)
Rhino(L)
Rhino(L)
Crocodile(M)
Eagle(B)
Owl(B)
Warthog(S)
Meerkat(S)
etc.

Do the following:

- Rename the folder **Question 2 Java** as **Quest2_X** (where X must be replaced by your examination number).
- Rename the **Competitor.java** file in the folder **Quest2_X** to **CompetitorXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Open the **CompetitorXXXX.java** file.
- Change the **class name and constructor method** to **CompetitorXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Add your examination number as a comment in the first line of the **CompetitorXXXX.java** class. Save the file.
- Rename the **TestCompetitor.java** file in the folder **Quest2_X** to **TestCompetitorXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Open the **TestCompetitorXXXX.java** file.
- Change the **class name** to **TestCompetitorXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number). Save the file.

2.1 The object class named **CompetitorXXXX.java** will represent the results for a single competitor including his/her name and how many animals in each category he/she sighted. Note the following:

- All fields in this class should be private and all methods public.
- Parts of this class have been inserted as comments in order to get the class to compile.
- In addition to modifying the given methods, you will be required to add code for new methods as described below.

Do the following in the **CompetitorXXXX.java** file:

2.1.1 Create private fields **with the following names** to hold the data. You should choose appropriate data types for these fields:

- **name** – name of competitor
- **largeGameCount** – total number of large game animals sighted
- **smallGameCount** – total number of small game animals sighted
- **birdCount** – total number of birds sighted

It is important that you use the field names given in bold above in order for the given code to operate correctly. (3)

2.1.2 (a) You have been provided with a **default constructor method**. Write an **additional constructor method** that has one parameter for the competitor's name. Initialise the name field using the parameter value and initialise the other fields to zero. (2)

(b) You have been provided with three methods named **spotLarge**, **spotSmall** and **spotBird**. Remove the comment symbols from the code provided in these methods. (1)

2.1.3 You have been provided with an incomplete typed method called **calculatePoints** that should return the total number of points for a competitor. The points are awarded as follows:

- Five points for each large game animal sighted
- Three points for each small game animal sighted
- Two points for each bird sighted

The method contains an incomplete statement that has been blocked out as a comment. The total points are supposed to be calculated by calculating the sum of the number of large game sighted multiplied by five, the number of small game sighted multiplied by three and the number of birds sighted multiplied by two.

Remove the comment symbols from the given statement and complete the code so that the method returns the correct result. (3)

2.1.4 Write a method called **totalAnimals** which returns the total number of animals sighted as an integer. The total is calculated by calculating the sum of the number of large game, small game and birds sighted. (2)

2.1.5 Write a 'get' method called **getName** to return the name of the competitor. (2)

2.1.6 Write a method called **mostSpotted** that will determine and return the category of animal ("Large Game", "Small Game" or "Bird") that the competitor spotted the most. (4)

2.1.7 You have been provided with a method called **toString** that constructs and returns a string with the name and sighting results of a competitor. However, the code provided in the method has been inserted as comments.

Remove the comment symbols so that the given statements will execute and complete the code so that it returns the information in the following format:

```
Competitor : name
Large : largeGameCount, Small : smallGameCount, Bird : birdCount
Total Animals : <tab>totalAnimals
```

Example of the output when the returned string is displayed (on the next page):

```
Competitor : Jane
Large : 9  Small : 6  Bird : 7
Total Animals :        22
```

(5)

2.2 In the **TestCompetitorXXXX.java** file (the main class) you have been given code to display the following menu when the program is executed:

```
          Menu

    Option A
    Option B

    Q - QUIT

    Your choice? :
```

Open the **TestCompetitorXXXX.java** file (the main class) and do the following:

- Add your examination number as a comment in the first line of the **TestCompetitorXXXX.java** class.
- Write code to do the following in the **TestCompetitorXXXX.java** file (the main class) in the given program:

2.2.1 Read information from the text file **Sightings.txt** according to the following steps:

(a) Test if the text file exists. Display a suitable message if the file does not exist and terminate the program. Continue with the remainder of the steps if the file exists.

(b) Read the first line from the text file and store this as the competitor's name.

(c) Using the competitor's name, create a **single object** of type **CompetitorXXXX**. You do not need to create an array of these objects as **only one competitor will be processed** each time the program is run.

(d) Use a loop to do the following:

- Read a line of text (one animal) from the text file.
- Test if the category of the animal that appears in brackets after the name of the animal is valid or not. Only the letters L (for large game), S (for small game) and B (for birds) are valid. Display an appropriate message which includes the name of the animal for an invalid category.
- If the category is valid, then call one of the following methods from your **CompetitorXXXX** class: **spotLarge**, **spotSmall**, **spotBird**. These methods increase the number of large game, small game and bird sightings respectively each time the method is called.

(e) Use two counter variables to keep track of how many valid and invalid categories were recorded.

(f) Display the total number of invalid and valid categories as shown below.

```
Crocodile is not in a valid category
Pigeon is not in a valid category
Springbok is not in a valid category

22 valid categories processed
3 invalid categories processed
```

(20)

2.2.2 Menu Option A

When the user selects this menu option, the program must display the name and the results for the competitor by using the **toString** method from the **CompetitorXXXX** class.

Example of output:

```
Competitor : Jane
Large : 9  Small : 6  Bird : 7
Total Animals :         22
```

(2)

2.2.3 Menu Option B

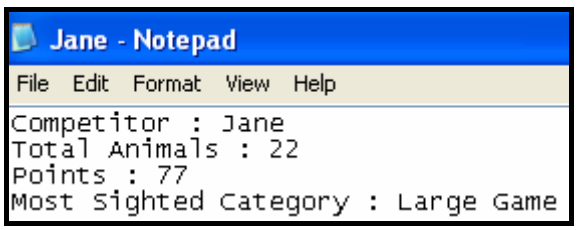
When the user selects this menu option the program must do the following:

- Create a new text file to save the name and results for the competitor. Construct a name for the text file that will contain the name of the competitor.

NOTE: Do not hardcode 'Jane.txt' as the file name since the name of the file should vary according to the name of the competitor.

- Call the appropriate methods from the **CompetitorXXXX** class to save the information to the created file.

Example of the contents of the text file:



- Display a message indicating that the information was successfully written to the file.

Example of output:

```
Results Successfully Written to File
```

(5)

- Make sure that your examination number is entered as a comment in the first line of the main class **TestCompetitorXXXX.java** as well as the object class **CompetitorXXXX.java**.
- Save all the files (File/Save All).
- Printouts of the code for the classes **TestCompetitorXXXX.java** and **CompetitorXXXX.java** will be required. **[49]**

QUESTION 3: JAVA – PROGRAMMING

The park rangers are doing some research on the movement of a **specific group of animals** relative to a **watering hole** situated at a specific location. They have monitored and recorded the locations of this group of animals over a specific period of time.

You have been given an incomplete program in the folder named **Question 3 Java**.

Do the following:

- Rename the folder named **Question 3 Java** to **Quest3_X** (where X should be replaced with your examination number).
- Rename the file **TestDistances** in this folder to **TestDistancesXXXX.java** (XXXX should be replaced by the last FOUR digits of your examination number).
- Open the file (incomplete program) **TestDistancesXXXX.java**. Change the name of the class to **TestDistancesXXXX.java**.
- Add your examination number as a comment in the first line of the program.
- Execute the program. A menu with the following options will be displayed:

```
Menu

Option A
Option B

Q - QUIT
```

NOTE: You may use one or more classes for this solution.

3.1 Do the following:

- Declare an array called **arrEntries** that must contain a maximum of 12 strings.
- Use the provided assignment statements to initialise the **arrEntries** array.
- Use the array appropriately in your program to answer the questions that follow.

Each array entry consists of a string indicating the coordinates of the location of the group of animals relative to the watering hole at a specific time. Each string has the following format:

X-Coordinate,Y-Coordinate:Time Recorded

Example of the first five strings assigned to the array (on the next page):

```
arrEntries[0]  := "12,15:02h00";
arrEntries[1]  := "13,10:05h00";
arrEntries[2]  := "9,20:06h00";
arrEntries[3]  := "10,15:09h00";
arrEntries[4]  := "7,8:10h00";
```

NOTE:
In the first entry:
- The x-coordinate is 12
- The y-coordinate is 15
- The time is 02h00

(3)

3.2 The coordinates of the watering hole are as follows:

X-coordinate = 10
Y-coordinate = 10

Complete the code for the option **Option A** as follows:

- Declare TWO variables to store the coordinates of the watering hole and assign values to these variables.

- For each recording in the **arrEntries** array, extract the x- and y-coordinates and the time recorded. Use the x- and y-coordinates to calculate the distance between the location of the animals and the location of the watering hole. The distance should be an integer which has been rounded off. Use the following formula to calculate the distance:

$$\text{Distance} = \sqrt{(\text{AnimalXpos} - \text{WaterXpos})^2 + (\text{AnimalYpos} - \text{WaterYpos})^2}$$

where

AnimalXpos represents the x-coordinate of where the animal was spotted
AnimalYpos represents the y-coordinate of where the animal was spotted
WaterXpos represents the x-coordinate of where the watering hole is situated
WaterYpos represents the y-coordinate of where the watering hole is situated

- For each entry in the **arrEntries** array, display the time, the distance from the watering hole and the x- and y-coordinates. Use a suitable heading and subheadings.

  Example of output (on next page):

Distances from the watering hole			
Time	Distance(km)	X-pos	Y-pos
02h00	5	12	15
05h00	3	13	10
06h00	10	9	20
09h00	5	10	15
10h00	3	7	8
11h00	0	10	10
14h00	8	12	18
17h00	8	7	18
19h00	3	11	7
20h00	0	10	10
23h00	12	2	1
24h00	7	12	17

(17)

3.3 The rangers had difficulty in tracking the animals. To facilitate the tracking of animals, it has been decided to create a tag for each animal in order to locate them more easily.

You are required to write code for **Option B** that will allow you to do the following:

- Enter the number of different types of animals in a mixed group of animals.
- Enter each type of animal in the group followed by the number of that type of animal in the group.

Your code must generate a tag for each of the animals in the group. The tag is generated as follows:

- Extract the first two letters from the name of the type of animal, for example "Rh" will be extracted from Rhino.
- Extract the last letter from the name of the type of animal, for example "o" will be extracted from Rhino.
- Randomly generate a three-digit even number per animal type.
- Combine the extracted letters and the number generated to form the first part of the tag.
- For each animal of a specific type, add a hyphen and a unique number beginning at 1 for the first animal of a specific type to the tag as indicated in the sample output.

You are required to display the name of the type of animal as part of a heading and a numbered list with a tag number for each animal of the specific type.

Example of input and output for a group of animals consisting of three different types of animals, for example five zebra, four black wildebeest and ten impala (on the next page):

```
Enter the number of different types of animals in the group : 3

Enter the name of animal type 1 : Zebra
Enter the number of animals of type 1 in the group : 5

Zebra                Tag number
1.                   Zea794-1
2.                   Zea794-2
3.                   Zea794-3
4.                   Zea794-4
5.                   Zea794-5

Enter the name of animal type 2 : Black Wildebeest
Enter the number of animals of type 2 in the group : 4

Black Wildebeest     Tag number
1.                   Blt780-1
2.                   Blt780-2
3.                   Blt780-3
4.                   Blt780-4

Enter the name of animal type 3 : Impala
Enter the number of animals of type 3 in the group : 10

Impala               Tag number
1.                   Ima726-1
2.                   Ima726-2
3.                   Ima726-3
4.                   Ima726-4
5.                   Ima726-5
6.                   Ima726-6
7.                   Ima726-7
8.                   Ima726-8
9.                   Ima726-9
10.                  Ima726-10
```

NOTE: Different random numbers will be generated for each execution of the program. (16)

- Enter your examination number as a comment in the first line of the class **TestDistancesXXXX.java**, as well as any other class(es) you may have created.
- Save the class(es).
- A printout of the code for the class **TestDistancesXXXX.java**, as well as any other class(es) you may have created, will be required. **[36]**

TOTAL SECTION B: 120
GRAND TOTAL: 120

ANNEXURE A: Table description sheet

This sheet shows the data structure and sample data for the tables used in the **SightingsDB** database

tblSightings Table Structure

tblSightings : Table

Field Name	Data Type	Description
SightingID	Number	A unique ID given to each sighting
SightingDate	Date/Time	The date of the sighting
Animal	Text	The animal that was sighted
NumAnimals	Number	The number of animals that were sighted
Young	Yes/No	If there were any young sighted
RangerID	Number	The ID of the ranger that sighted the animals

tblRanger

tblRangers : Table

Field Name	Data Type
RangerID	Number
Name	Text
Surname	Text
Rank	Text
DateAppointed	Date/Time

tblSightings Table – Data Sample

Microsoft Access - [tblSightings : Table]

File Edit View Insert Format Records Tools Window Help

SightingID	SightingDate	Animal	NumAnimals	Young	RangerID
101	5/3/2010	Giraffe	2	True	3
102	4/5/2010	Impala	23	False	5
103	1/15/2010	Lion	13	False	7
105	6/2/2010	Aardvark	1	True	8
106	4/3/2010	Elephant	1	False	3
107	6/21/2010	Lion	23	False	13
108	4/21/2010	Elephant	32	False	1
109	5/8/2010	Rhino	13	False	2
110	6/23/2010	Rhino	13	False	3
111	5/29/2010	Elephant	10	False	15
112	3/25/2010	Aardvark	7	True	19
113	3/26/2010	Giraffe	14	True	20
114	2/12/2010	Aardvark	25	True	14
115	4/1/2010	Cheetah	14	False	18
116	7/2/2010	Giraffe	7	False	20
117	5/31/2010	Elephant	17	False	14
118	1/19/2010	Impala	8	False	18
119	6/14/2010	Lion	8	True	7
120	1/22/2010	Giraffe	15	False	3

tblRangers

Microsoft Access - [tblRangers :

File Edit View Insert Format

RangerID	Name
1	Jada
2	Kenyon
3	Dylan
4	Sylvester
5	Urielle
6	Giselle
7	Amos
8	Tobias
9	Odessa
10	Charles
11	Ariel
12	Hammett
13	Tamara
14	Karleigh
15	Ivory
16	Caryn
17	Denise
18	Lev
19	Alyssa
20	Conan

ANNEXURE B: Instructions to create the database SightingsDB.mdb

If you cannot use the database provided, do the following:

- Use the two text files named **tblSightings.txt** and **tblRangers.txt** supplied. Create your own database named **SightingsDB** with a table named **tblSightings** and another table named **tblRangers** in the folder called **Question 1 Delphi** or **Question 1 Java**.
- Change the data types and the sizes of the fields in the two tables to the specifications given below.

The **tblSightings** table stores data on the sightings of animals for the game park. The fields in the **tblRangers** table are defined as follows:

Field Name	Type	Size	Comment
SightingID	Number	Integer	A unique ID given to each sighting
SightingDate	Date/Time	ShortDate	The date of the sighting
Animal	Text	20	The animal that was sighted
NumAnimals	Number	Integer	The number of animals that were sighted
Young	Yes/No	Boolean	If there were any young sighted
RangerID	Number	Integer	The ID of the ranger that sighted the animals

See **ANNEXURE A:** Example of the data contained in the **tblSightings** table.

The fields in the **tblRangers** table are defined as follows:

Field Name	Type	Size	Comment
RangerID	Number	Integer	A unique ID for each ranger
Name	Text	20	The name of the ranger
Surname	Text	20	The surname of the ranger
Rank	Text	20	The rank of the ranger
DateAppointed	Date/Time	ShortDate	The date the ranger was appointed

See **ANNEXURE A:** Example of the data contained in the **tblRangers** table.

ANNEXURE C: Instructions to connect to the database in Delphi

In Delphi: If you cannot use the database provided, do the following:

- Click on the ADOQuery component named **qrySightings**.
- Click on the Ellipse button (three dots) to the right of the 'ConnectionString' property in the Object Inspector.
- Click on the Build button which takes you to the Data Link Properties dialogue box.
- Click on the Provider tab to open the provider tab sheet and select Microsoft Jet 4.0 OLE DB Provider. Click on the Next button.
- The Connection tab sheet will be displayed. The first option on the Connection tab sheet provides an Ellipse button (three dots) that allows you to browse and look for the **SightingsDB** file. You will find this file in the **Question 1 Delphi** folder. Once you have found it, select the **SightingsDB** file and click on the Open button.
- Remove the user name Admin.
- Click on the Test Connection button.
- Click OK on each one of the open dialogue windows.

120

INFORMATION TECHNOLOGY PAPER 1
NOVEMBER 2010

INFORMATION SHEET *(to be completed by candidates)*

NAME OF PROVINCE _____

CENTRE NUMBER _____

EXAMINATION NUMBER _____

WORK STATION NUMBER _____

DATE OF EXAMINATION _____

Programming language used
(Mark appropriate box with a cross (X).)

Delphi	Java

FOLDER NAME _____

Enter the file name used for each answer and tick if saved.

Question number	File names	Saved (*tick* ✓)	Maximum mark	Mark achieved	Marker initial/ code
1			35		
2			49		
3			36		
TOTAL			**120**		

Comment (*for official use only)*

_____

_____

_____

_____

_____

_____

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2010

MEMORANDUM

MARKS: 120

This memorandum consists of 26 pages.

GENERAL INFORMATION:

- **Pages 2 – 11 contain the Delphi memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 12 – 20 contain the Java memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 21 – 26 contain Addenda A to F which includes a marking grid for each question for candidates using either one of the two programming languages.**
  Copies of the appropriate Addenda should be made for each learner to be completed during the marking session.

SECTION A: DELPHI

QUESTION 1: PROGRAMMING AND DATABASE

```
unit Question1_U;

interface

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

type
  TfrmSightings = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnB: TButton;
    btnE: TButton;
    btnA: TButton;
    btnD: TButton;
    btnF: TButton;
    BitBtn1: TBitBtn;
    btnC: TButton;
    qrySightings: TADOQuery;
    tblSightings: TDataSource;
    grdSightings: TDBGrid;
    btnG: TButton;
    procedure btnBClick(Sender: TObject);
    procedure btnEClick(Sender: TObject);
    procedure btnDClick(Sender: TObject);
    procedure btnCClick(Sender: TObject);
    procedure btnFClick(Sender: TObject);
    procedure btnAClick(Sender: TObject);
    procedure btnGClick(Sender: TObject);

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

var
  frmSightings: TfrmSightings;

implementation

{$R *.dfm}
```

```
procedure TfrmSightings.btnAClick(Sender: TObject);          // Question 1.1
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT * ✓FROM tblSightings✓ ORDER BY✓ SightingID
DESC✓';
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (4)
//=========================================================================

procedure TfrmSightings.btnBClick(Sender: TObject);          // Question 1.2
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young
= true✓';
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (4)
//=========================================================================

procedure TfrmSightings.btnCClick(Sender: TObject);          // Question 1.3
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year
(DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers'✓;
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (6)
//=========================================================================

procedure TfrmSightings.btnDClick(Sender: TObject);          // Question 1.4
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT Animal✓,format(Avg(NumAnimals)✓,"0.00"✓)  AS
[AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓';
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (6)
//=========================================================================
```

OR round(Avg(NumAnimals), 2)

```
procedure TfrmSightings.btnEClick(Sender: TObject);          // Question 1.5
var
  id : integer;
  numRecords : integer;
begin
  id := StrToInt(InputBox('Delete Sighting', 'Enter the ID of the sighting to
delete', '1')); ✓
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'DELETE✓ FROM tblSightings✓ WHERE SightingID = '+
IntToStr(id) ✓;
 numRecords := qrySightings.ExecSQL;
  MessageDlg (IntToStr(numRecords) + ' record deleted.',mtInformation,[mbok],0);
end;                                                                    (4)
//=========================================================================
```

```
procedure TfrmSightings.btnFClick(Sender: TObject);          // Question 1.6
var
  numRecords : integer;
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'UPDATE✓ tblSightings✓ SET✓ Animal = "White Rhino"✓ WHERE
Animal = "Rhino"✓';
  numRecords := qrySightings.ExecSQL;
  MessageDlg (IntToStr(numRecords) + ' records updated.', mtInformation,[mbok],0);
end;                                                                          (5)
//=============================================================================

procedure TfrmSightings.btnGClick(Sender: TObject);          // Question 1.7
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT SightingDate,Name,Surname✓ FROM tblSightings,
tblRangers✓ WHERE tblSightings.RangerID✓ = tblRangers.RangerID✓ AND Animal =
"Elephant"✓ AND SightingDate > #30/04/2010#'; ✓
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                          (6)
                                                                             [35]

end.
```

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

unit uCompetitorXXXX;

```
interface

uses SysUtils;
```

***//* Q 2.1.1**

```
type TCompetitor = class
   private ✓                (3)
    name : String; ✓
    largeGameCount : integer;   }
    smallGameCount : integer;   }    ✓
    birdCount : integer;

   public
    constructor Create; overload;
    constructor Create(sName : String); overload;
    function toString : String;
    function totalAnimals : integer;
    procedure spotLarge;
    procedure spotSmall;
    procedure spotBird;
    function calculatePoints : integer;
    function getName : String;
    function mostSpotted : String;
end;

implementation

{ Competitor }
//============================================================================
constructor TCompetitor.Create;
begin

end;
```

Q 2.1.1
(1) Four private variables
(1) Declare name as private string
(1) All count variables declared as private integers

***//* Q 2.1.2** **(3)**

```
constructor TCompetitor.Create(sName : String);   ✓
begin
  name := sName;
  largeGameCount := 0;
  smallGameCount := 0;      ✓
  birdCount := 0;
end;

procedure TCompetitor.spotLarge;
begin
   inc(largeGameCount);
end;

procedure TCompetitor.spotSmall;
begin
  inc(smallGameCount);          ✓
end;

procedure TCompetitor.spotBird;
begin
   inc(birdCount);
end;
//============================================================================
```

Q 2.1.2
(a) (1) Receive name as String
(1) Initialize all instance fields
(b) (1) Remove comment-signs from the code in the three given 'spot'-methods

Q 2.1.3
(3) Multiply each category with the correct value add the values and assign to Result
Subtract 1 mark for each type of error, not for the same type of error

```
// Q 2.1.3                          (3)
function TCompetitor.calculatePoints: integer;
begin
   Result := largeGameCount * 5 + smallGameCount * 3 + birdCount * 2 ✓✓✓;
end;
//=============================================================
```

Q 2.1.4
(1) Return type integer
(1) Return sum of animal counts

```
// Q 2.1.4                          (2)
function TCompetitor.totalAnimals: integer✓;
begin
  Result := largeGameCount + smallGameCount + birdCount; ✓
end;

//=============================================================================
```

Q 2.1.5
(1) Return type String
(1) Return name

```
// Q 2.1.5                          (2)
function TCompetitor.getName: String; ✓
begin
  Result := name ✓
end;

//=============================================================
```

Q 2.1.6
(1) Return type String
(1) If to get Large Game
(1) If to get Small game
(1) Get Birds

```
// Q 2.1.6                          (4)
function TCompetitor.mostSpotted: String✓;
begin
  if (largeGameCount > smallGameCount) AND (largeGameCount > birdCount) ✓ then
    Result := 'Large Game'
  else if (largeGameCount < smallGameCount) AND (smallGameCount > birdCount) ✓ then
    Result := 'Small Game'
  else ✓                           // Small game AND birds must have else statements, -
                                   // otherwise another if statement for birds
    Result := 'Bird';

end;                               // Correct variations of this code must be accepted
//=============================================================================
```

Q 2.1.7
(1) New line (#13)
(1) Add strings
(1) All count elements added
(1) Last line added
(1)Tab (#9) added

```
//Q 2.1.7                          (5)
function TCompetitor.toString: String;
var
  objStr : String;
begin

  objStr := 'Competitor : ' + name + #13✓;
  objStr := objStr✓ + 'Large : ' + IntToStr(largeGameCount) + '  Small : '
+ IntToStr(smallGameCount) + '  Bird : ' + IntToStr(birdCount) + #13✓;
  objStr := objStr + 'Total Animals : ' + #9✓ + IntToStr(totalAnimals) ✓;
  Result := objStr;
end;               // May use more tabs (#9) than indicated here
//=============================================================================
```

unit QuestTwoXXXX_U;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, Menus;

type
  TfrmQuest2 = class(TForm)
    redOutput: TRichEdit;
```

```
    MainMenu1: TMainMenu;
    mnuAQuestion2: TMenuItem;
    mnuBQuestion2: TMenuItem;
    Quit1: TMenuItem;
    procedure Quit1Click(Sender: TObject);
    procedure mnuAQuestion2Click(Sender: TObject);
    procedure mnuBQuestion2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuest2: TfrmQuest2;

implementation

{$R *.dfm}
```

//Q 2.2.1 (20)

```
uses
  uCompetitorXXXX; ✓
var
  Competitor : TCompetitor;  ✓

procedure TfrmQuest2.Quit1Click(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TfrmQuest2.FormActivate(Sender: TObject);
var
  tFile : textfile;
  sName, sAnimal, sAnimalName : String;
  iValid, iInvalid, iBracket : integer;
  sLetter :char;
begin

  redOutput.Lines.Clear;
  iValid := 0;
  iInvalid := 0;  ✓

  If FileExists('Sightings.txt') <> TRUE then ✓
    begin
      ShowMessage('File not found');  ✓
      Application.Terminate;
    end
    AssignFile(tFile, 'Sightings.txt');   ✓
    Reset(tFile); ✓
    ReadLn(tFile, sName); ✓
    Competitor := TCompetitor.Create✓(sName)✓;
    while NOT EOF(tFile) DO  ✓
        begin
         ReadLn(tFile, sAnimal);  ✓
         iBracket := pos('(', sAnimal);
         sAnimalName := copy(sAnimal,1,iBracket - 1); ✓
         sLetter := sAnimal[iBracket + 1]; ✓
```

Q 2.2.1
(1) uses object class
(1) Declare object variable
(1) Initialise two counter variables
(1) Test if file exists
(1) If not, display message & terminate
(1) Assign file
(1) Open file for reading
(1) Read name of competitor from file
(2) Create competitor object
(1) Loop with begin & end in correct places
(1) Read line from text file
(1) Copy animal name from line
(1) Copy animal category from line
(1) Case (OR if-statements)
for each valid category
(1) call method to inc counter
(1) add 1 to valid counter
(1) For invalid category
(1)inc counter & display message
(1) Display the number of valid & invalid categories processed

```
        case sLetter of  ✓
          'L' : begin
                  Competitor.spotLarge;
                  inc(iValid);
                end;
          'S' : begin
                  Competitor.spotSmall;
                  inc(iValid);
                end;
          'B' : begin
                  Competitor.spotBird;
                  inc(iValid);
                end;
                                                    ✓✓✓
          else begin
                  redOutput.Lines.Add(sAnimal + ' is not a valid animal'); ✓
                  Inc(iInvalid);

                end;
        end; // case

      end;
    CloseFile(tFile);
    redOutput.Lines.Add('');
    redOutput.Lines.Add(IntToStr(iValid) + ' valid categories processed');  ✓
    redOutput.Lines.Add(IntToStr(iInvalid) + ' invalid categories
                                                          processed');
end;
//============================================================================
```

***//* Q 2.2.2 (2)**

```
procedure TfrmQuest2.mnuAQuestion2Click(Sender: TObject);
begin
   redOutput.Lines.Clear;
   redOutput.Lines.Add(Competitor.toString);  ✓✓
end;
//============================================================================
```

Q 2.2.2
(1) Call the toString method of the object
(1) in a display statement

***//* Q 2.2.3 (5)**

```
procedure TfrmQuest2.mnuBQuestion2Click(Sender: TObject);
var
  tFile : textfile;
  fileName: String;
begin
  fileName := Competitor.getName + '.txt';  ✓
  AssignFile(tFile, fileName);
  Rewrite(tFile);                            ✓

  Writeln(tFile, 'Competitor : ' + Competitor.getName);
  WriteLn(tFile, 'Total Animals : ' + IntToStr(Competitor.totalAnimals));  ✓✓
  WriteLn(tFile, 'Points : ' + IntToStr(Competitor.calculatePoints));
  WriteLn(tFile, 'Most Sighted Category : ' + Competitor.mostSpotted);

  CloseFile(tFile);  ✓

  redOutput.Lines.Clear;
  redOutput.Lines.Add('Results Successfully Written to File');
end;
end.
//============================================================================
```

Q 2.2.3
(1) Construct the name of the new file correctly
(1) Open a new file
(1) Construct the information to write to the file correctly
(1) Write to the file
(1) Close the file

[49]

QUESTION 3: DELPHI PROGRAMMING

NB: This is only a sample – learners may answer this question in any way they see fit. Make use of the generalized rubric in the mark sheets for marking.

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    redOutput: TRichEdit;
    pnlButtons: TPanel;
    btnA: TButton;
    btnB: TButton;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);

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

var
  frmQuestion3: TfrmQuestion3;

implementation

{$R *.dfm}
```

//Question 3.1 (3)

```
var
  arrEntries : array[1..12] ✓of string; ✓

procedure TfrmQuestion3.FormCreate(Sender: TObject);
 begin
  arrEntries[1]  := '12,15:02h00';
  arrEntries[2]  := '13,10:05h00';
  arrEntries[3]  := '9,20:06h00';
  arrEntries[4]  := '10,15:09h00';
  arrEntries[5]  := '7,8:10h00';
  arrEntries[6]  := '10,10:11h00';
  arrEntries[7]  := '12,18:14h00';
  arrEntries[8]  := '7,18:17h00';
  arrEntries[9]  := '11,7:19h00';
  arrEntries[10] := '10,10:20h00';
  arrEntries[11] := '2,1:23h00';
  arrEntries[12] := '12,17:24h00'; ✓
end;
//============================================================================
```

Q 3.1
(2) Declare array of strings
(1) Remove the comments-signs from given statements assigning strings to array

//Quest 3.2 (17)

```
procedure TfrmQuestion3.btnAClick(Sender: TObject);
var
   K, iDistance :integer;
   iWaterX, iWaterY, xPos, yPos   :integer;
   iComma, iColon :integer;
   sTime:string;
begin
 iWaterX := 10;  }      ✓
 iWaterY := 10;  }

 redOutput.Paragraph.TabCount := 4;
 redOutput.Paragraph.Tab[0] := 60;
 redOutput.Paragraph.Tab[1] := 100;
 redOutput.Paragraph.Tab[2] := 130;
 redOutput.Paragraph.Tab[2] := 150;

  redOutput.Lines.Add('Distances from the watering hole'); ✓
  redOutput.Lines.Add('Time' + #9 + 'Distance(km)  ' + #9 + 'X-pos   Y-pos'); ✓
  for K := 1 to 12 do✓
    begin
     iComma := pos(',', arrEntries[K]); ✓
     xPos := StrToInt(copy(arrEntries[K],1,iComma-1)); ✓
     iColon := pos(':', arrEntries[K]); ✓
     yPos := StrToInt(copy(arrEntries[K],iComma + 1,iColon - iComma-1)); ✓
     iDistance:=  Round✓ (Sqrt✓ (Sqr(xPos - iWaterX  ) ✓ + Sqr(yPos –
                                                    iWaterY ))); ✓
     delete(arrEntries[K],1,iColon); ✓
     sTime:= arrEntries[K]; ✓
     redOutput.Lines.Add(sTime ✓+ #9 + '       ' + IntToStr(iDistance) ✓ + #9
                 + #9+ IntToStr(xPos) + #9 + IntToStr(yPos)); ✓
   end;
end;

//=====================================================================
```

Q 3.2
(1) Initialise water x & y positions
(2) Display heading & subheadings
(1) Loop with begin & end in correct places
Inside loop:
(2) Extract x-pos,
(2) Extract y-pos,
(2) Extract time from array string
(3) Calculate distance,
(1) Round answer off,
(3) Display all information

//Question 3.3 (16)

```
procedure TfrmQuestion3.btnBClick(Sender: TObject);
var
  K, iNumber, iCount, iLength  :integer;
  iRandomNumber, L :integer;
  sAnimal, sTag :string;

begin
  redOutput.Paragraph.TabCount := 1;
  redOutput.Paragraph.Tab[0] := 70;
  Randomize;
  redOutput.Clear;
  iNumber := StrToInt(InputBox('Animal tags',
           'Enter the number of different types of
             animals in the group?', '')); ✓
  for K := 1 to iNumber do✓
    begin✓
      sAnimal := InputBox('Animal Tags', 'Enter the
            name of animal type ' + intToStr(K), '');✓
      iCount := StrToInt(InputBox('Animal Tags',
                 'Enter the number of animals of type '
               + intToStr(K) + ' in the group', ''));✓

      sTag := copy(sAnimal, 1, 2); ✓
```

Q 3.3
(1) Enter the number of different types of animals,
(1) Loop from 1 to the number of different types
(1) Inside loop:
(1) Enter animal type
(1) Enter the number in the group
(1) Extract first 2 letters and assign to tag
(2) Extract the last letter,
(2) Generate random number in correct range,
(1) Validate an even number,
(1) Add random number to tag, and hyphen (1) Heading
(1) Inner loop to add the number of animals in each group,
(2) Display the correct data aligned correctly inside inner loop

```
      iLength := length(sAnimal);
      sTag := sTag + copy(sAnimal, iLength✓,1);✓
      repeat
        iRandomNumber := Random(900) ✓ + 100; ✓
      until iRandomNumber mod 2 = 0; ✓
      redOutput.Lines.Add('');
      sTag := sTag + IntToStr(iRandomNumber) ✓;
      redOutput.Lines.Add(sAnimal + #9 + 'Tag number');✓
      for L := 1 to iCount do✓
        begin
         redOutput.Lines.Add(intToStr(L) + '.' + #9 + sTag ✓+ '-' +intToStr(L)); ✓
        end;
    end;
end;
//============================================================================
end.                                                                    [36]
```

OR

```
iRandomNumber:= Random(899)+ 100;
if iRandomNumber mod 2 <> 0 then
      inc(iRandomNumber);
```

END OF SECTION A: DELPHI **TOTAL SECTION A: 120**

SECTION B: JAVA

QUESTION 1: PROGRAMMING AND DATABASE

```
import java.io.*;
import java.sql.*;
import javax.swing.*;
import java.util.Scanner;

public class TestSightings
 {
  public static void main (String[] args) throws SQLException,IOException
   {
    BufferedReader inKb = new BufferedReader (new InputStreamReader(System.in));

    Sightings DB = new Sightings();
    System.out.println();
    char choice = ' ';
     do
      {
         System.out.println("      MENU");
         System.out.println();
         System.out.println("    Option A");
         System.out.println("    Option B");
         System.out.println("    Option C");
         System.out.println("    Option D");
         System.out.println("    Option E");
         System.out.println("    Option F");
         System.out.println("    Option G");
         System.out.println();
         System.out.println("    Q - QUIT");
         System.out.println(" ");
         System.out.print("    Your Choice? ");
         choice = inKb.readLine().toUpperCase().charAt(0);
         System.out.println(" ");
         String sql = "";
         switch(choice)
         {
          case 'A':                                          //Question 1.1
          {
            sql = "SELECT * ✓FROM tblSightings✓ ORDER BY ✓SightingID DESC✓";
            DB.A(sql);
            break;
          }                                                                  (4)
//=========================================================================
         case 'B':                                          //Question 1.2
         {
            sql = "SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young =
                                                            true✓";
           DB.B(sql);
           break;
         }                                                                   (4)
//=========================================================================
        case 'C':                                           //Question 1.3
         {
            sql = "SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year
            (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers"✓;
            DB.C(sql);
            break;
         }                                                               (6)
//=========================================================================
```

```
       case 'D':                                          //Question 1.4
          {
             sql = "SELECT Animal✓, format(Avg(NumAnimals) ✓, '0.00'✓) AS
                   [AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓";

             DB.D(sql);
             break;
          }                                                                  (6)
//=========================================================================
      case 'E':   //Question 1.5
          {
             Scanner kb = new Scanner (System.in);
             System.out.println();
             System.out.println("Enter the ID of the sighting to delete");
             int id = kb.nextInt();✓
             sql = "DELETE✓FROM tblSightings✓ WHERE SightingID = "+id+"✓";
             DB.E(sql);
             break;
          }                                                                  (4)
//=========================================================================
          case 'F':                                          //Question 1.6
          {
             sql = "UPDATE✓ tblSightings✓ SET✓ Animal = 'White Rhino'✓ WHERE
                   Animal = 'Rhino'✓";
             DB.F(sql);
             break;
          }                                                                  (5)
//=========================================================================
          case 'G':                                          //Question 1.7
           {
            sql = "SELECT SightingDate,Name,Surname✓ FROM tblSightings,
                   tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND
                   Animal = 'Elephant'✓ AND SightingDate > #30/04/2010#";✓

            DB.G(sql);
            break;
          }                                                                  (6)
//=========================================================================

       } //switch
      }while (choice != 'Q');

      DB.disconnect();
      System.out.println("Done");
    }
  }
```

[35]

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

CompetitorXXXX.java

```
public class CompetitorXXXX
{
    // Q 2.1.1                          (3)
    private✓ String name; ✓
    private int largeGameCount;
    private int smallGameCount;          ✓
    private int birdCount;

        public CompetitorXXXX ()
    {

    }
    //=========================================================================
    // Q 2.1.2                          (3)
    public CompetitorXXXX (String aName) ✓
    {
        name = aName;

        largeGameCount = 0;
        smallGameCount = 0;              ✓
        birdCount = 0;
    }

    public void spotLarge()
    {
        largeGameCount++;
    }

    public void spotSmall()
    {
        smallGameCount++;                ✓
    }

    public void spotBird()
    {
        birdCount++;
    }

     //=========================================================================
    // Q 2.1.3                          (3)



    public int calculatePoints()
    {
      return largeGameCount * 5✓ + smallGameCount * 3 ✓+ birdCount * 2✓;
    }
//=============================================================================

    // Q 2.1.4                          (2)
    public int✓ totalAnimals()
    {
        return largeGameCount + smallGameCount + birdCount; ✓
    }
```

Q 2.1.1
(1) Four private variables
(1) Declare name as private string
(1) All count variables declared as private integers

Q 2.1.2
(1) Receive name as String
(1) Initialize all instance fields
(1) Remove comment-signs from code in the three given spot-methods

Q 2.1.3
(3) Multiply each category with the correct value, add the values and return value
Subtract 1 mark for each type of error, not for the same type of error

Q 2.1.4
(1) Return type integer
(1) Return sum of animal counts

```
//============================================================================
// Q 2.1.5                                    (2)
public String getName()✓
{
    return name; ✓
}
//============================================================================
```

Q 2.1.5
(1) Return type String
(1) Return name

```
// Q 2.1.6                                    (4)
public String✓ mostSpotted()
{
    if (largeGameCount > smallGameCount & largeGameCount > birdCount) ✓
    {

        return "Large Game";
    }
    else if (smallGameCount > largeGameCount & smallGameCount > birdCount)✓
    {
        return "Small Game";
    }
    else✓               // Small game AND birds must have else statements, -
                        // otherwise another if statement for birds
    {
        return "Bird";
    }                           // Correct variations of this code must be accepted
}
//============================================================================
```

Q 2.1.6
(1) Return type String
(1) If to get Large Game
(1) If to get Small game
(1) Get Birds

```
//Q 2.1.7                                    (5)
public String toString()
{
    String objStr = "Name : " + name + "\n"✓;
    objStr = objStr ✓+ "Large : " + largeGameCount + "  Small : "
+ smallGameCount + "  Bird : " + birdCount + "\n"✓;
    objStr = objStr + "Total Animals: \t"✓ + totalAnimals();✓

    return objStr;
}

}       // Accept correct use of formatter to construct the string

//============================================================================
```

Q 2.1.7
(1) New line (\n)
(1) Add strings
(1) All count elements added
(1) Add last line
(1) Tab (\t) added

TestCompetitorXXXX.java

```
import javax.swing.*;
import java.io.*;
import java.util.*;

public class TestCompetitorXXXX
{
```

// Q 2.2.1 (20)

```
public static void main(String args[]) throws Exception
{
 CompetitorXXXX Competitor = new CompetitorXXXX ();✓

 File f = new File("Sightings.txt");✓
 if (!f.exists())✓
 {
   System.out.println("File not found");
   System.exit(0);                                  } ✓
 }
    Scanner sc = new Scanner(f); ✓

    String sName = sc.nextLine()✓;
    Competitor✓  = new CompetitorXXXX ✓ (sName✓);

    int valid = 0;    } ✓
    int invalid = 0;  }

    while (sc.hasNextLine())✓
    {
      String sAnimal = sc.nextLine();✓
      int bracket = sAnimal.indexOf('(');
      String animal = sAnimal.substring(0,bracket); ✓
      char letter = sAnimal.charAt(bracket + 1); ✓

      switch (letter) ✓
      {
        case 'L' : Competitor.spotLarge();
                   valid++;
                   break;
        case 'S' : Competitor.spotSmall();
                   valid++;                          } ✓✓✓
                   break;
        case 'B' : Competitor.spotBird();
                   valid++;
                   break;
        default: System.out.println(animal + " is not
                 in a valid category");
                 invalid++;✓
                 break;
       }

 System.out.println();
 System.out.println(valid + " valid categories processed");     } ✓
 System.out.println(invalid + " invalid categories processed");}

 sc.close();
 }
//=============================================================================
```

Q 2.2.1

(2) Declare object variable
(1) Initialise two counter variables
(1) Test if file exists
(1) Display message & terminate
(1) Assign file
(1) Open file for reading
(1) Read name of competitor from file
(3) Create competitor object
(1) Loop with curly brackets in correct places
(1) Read line from text file
(1) Copy animal name from line
(1) Copy animal category from line
(1) switch OR if-statements
(1) For each valid category
(1) Call method to inc counter
(1) Add 1 to valid counter
For invalid category
(1) Inc counter & display message
(1) Display the number of valid & invalid categories processed

```
BufferedReader inKb = new BufferedReader (new InputStreamReader (System.in));
 char ch = ' ';
 while (ch != 'Q')
  {
      System.out.println();
      System.out.println("       MENU");
      System.out.println(" ");
      System.out.println("       Option A");
      System.out.println("       Option B");
      System.out.println(" ");
      System.out.println("       Q - QUIT");
      System.out.println(" ");
      System.out.print("       Your choice? ");
      ch = inKb.readLine().toUpperCase().charAt(0);

      switch (ch)
      {
   // Q 2.2.2                          (2)
       case 'A':
        {
          System.out.println();
          System.out.println(Competitor.toString());✓✓
          break;
        }
//=========================================================================
   // Q 2.2.3                          (5)
       case 'B':
       {
         PrintWriter fOut = new PrintWriter(new File✓ (Competitor.getName()+
                                                       ".txt"));✓

          fOut.println("Competitor : " + Competitor.getName());
          fOut.println("Total Animals : " + Competitor.totalAnimals());
          fOut.println("Points : " + Competitor.calculatePoints());
          fOut.println("Most Sighted Category : " + Competitor.mostSpotted());

          fOut.close();✓

         System.out.println();
         System.out.println("Results Successfully Written to File");
         break;
       }

       case 'Q':
       {
         System.exit(0);
       } // case

     } // switch

  } // while

 } // main

} // class
//=========================================================================
```

✓✓

Q 2.2.2
(1) Call the toString method of the object
(1) in a display statement

Q 2.2.3
(1) Construct the name of the new file correctly
(1) Open a new file
(1) Construct the information to write to the file correctly
(1) Write to the file
(1) Close the file

[49]

QUESTION 3: JAVA PROGRAMMING

NB: This is only a sample – learners may answer this in any way they see fit. Make use of the generalized rubric in the mark sheets for marking.

TestDistances.java

//Question 3.1 (3)

```
import java.util.Scanner;
import java.io.*;
import javax.swing.*;
public class TestDistances
{
  String ✓ [] arrEntries = new String [12]; ✓
  public TestDistances()
   {
      arrEntries[0] =  "12,15:02h00 ";
      arrEntries[1] =  "13,10:05h00 ";
      arrEntries[2] =  "9,20:06h00 ";
      arrEntries[3] =  "10,15:09h00 ";
      arrEntries[4] =  "7,8:10h00 ";
      arrEntries[5] =  "10,10:11h0";
      arrEntries[6] =  "12,18:14h00";
      arrEntries[7] =  "7,18:17h00";
      arrEntries[8] =  "11,7:19h00";
      arrEntries[9] =  "10,10:20h00";
      arrEntries[10]=  "2,1:23h00";
      arrEntries[11]=  "12,17:24h00";✓
   }
//=====================================================================
```

Q 3.1
(2) Declare array of strings
(1) Uncomment given statement assigning string to array

//Question 3.2 (17)

```
public void displayDistances()
 {
   int waterX = 10;
   int waterY = 10;   ✓
   int xPos;
   int yPos;
   System.out.println("Distances from the
                          watering hole");✓
   System.out.printf("%-10s%-15s%-10s%-10s",
                        "Time","Distance(km)",
                        "X-pos","Y-pos");✓
   System.out.println();
   int distance;
   String time;
   for (int i=0;i<12;i++)✓
    {
      String line = arrEntries[i];
      int psnComma = line.indexOf(','); ✓
      xPos = Integer.parseInt(line.substring(0,psnComma)); ✓
      int psnColon = line.indexOf(':'); ✓
      yPos = Integer.parseInt(line.substring(psnComma +
                                1,psnColon)); ✓
      time = line.substring(psnColon+1); ✓✓



      distance =  (int)(Math.round✓(Math.sqrt✓ (Math.pow((xPos– waterX),2) ✓
```

Q 3.2
(1) Initialise waterX & waterY
(2) Display heading & subheadings
(1) Loop with curly brackets correctly placed
Inside loop:
(2) Extract x-pos,
(2) Extract y-pos,
(2) Extract time from array string
(3) Calculate distance,
(1) Round answer off,
(3) Display all information

```
                                + Math.pow((yPos - waterY ),2))); ✓


      System.out.printf("%-10s%-15s%-10s%-10s", time, ✓distance ✓,xPos,yPos); ✓
      System.out.println();

    }
 }
//=====================================================================
```

//Question 3.3 (16)

```
public void tags()
      {
System.out.print("Enter the number of different
                 types of animals in the group: ");
int n = inKb.nextInt();✓
for (int i = 1; i <= n; i++)✓
{✓
  inKb.nextLine();
  System.out.println();
  System.out.print("Enter animal type "+ i + ": ");
  String animal = inKb.nextline();✓

  System.out.print("Enter the number of this type of
                        animal in the group: ");
  int number = inKb.nextInt();✓

  String tag = animal.substring(0,2); ✓

  tag = tag + animal.charAt(animal.length()✓ - 1);✓
  int randomNum;
  do
  {
    randomNum = (int)(Math.random()✓ * 900 + 100); ✓
  }
  while (randomNum % 2 !=0); ✓


  tag = tag + randomNum; ✓
  System.out.printf("%-20s%-20s",animal,"Tag number");✓
  System.out.println();
  for (int j = 1; j<= number; j++)✓
   {
     System.out.printf("%-20s%-20s",j+".",tag✓+"-"+j✓);
     System.out.println();
   }
  }
}
//=====================================================================

public static void main(String[] args) throws Exception
 {
   Scanner input = new Scanner(System.in);
   TestDistances obj = new TestDistances();
   char ch = ' ';
```

Q 3.3
(1) Enter the number of different types of animal
(1) Loop from 1 to the number of different types
(1)Inside loop:
(1) Enter animal type
(1) Enter the number in the group
(1) Extract first 2 letters and assign to tag
(2) Extract the last letter
(2) Generate number in correct range
(1) Validate even number
(1) Add random number to tag
(1) Heading
(1) Inner loop to the number of animals in each group
(2) Display the correct data aligned correctly inside inner loop

```
OR
randomNum =(int)(Math.random()*899+100);
if (randomNum % 2 !=0)
   randomNum++;
```

```
   while (ch != 'Q')
   {
       System.out.println();
       System.out.println("        Menu");
       System.out.println(" ");
       System.out.println("       Option A");
       System.out.println("       Option B");
       System.out.println(" ");
       System.out.println("       Q - QUIT");
       System.out.println(" ");
       System.out.print("       Your choice? ");

       ch = input.nextLine().toUpperCase().charAt(0);

       switch (ch)
       {
          case 'A':
          {
            obj.displayDistances();
            break;
          }

          case 'B':
          {
            obj.tags();
            break;
          }

          case 'Q':
          {
             System.exit(0);
          } // case

        } // switch

      } // while
   }
 }
//===========================================================================

}
```

Alternative structure: not advised but must be accepted:
The code can be used in place of the case statement. See electronic solution.

[36]

END OF SECTION B: JAVA **TOTAL SECTION B: 120**

GRAND TOTAL: 120

ADDENDUM A

QUESTION 1: DELPHI – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	EXAMINATION NUMBER:……

QUESTION 1: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	SELECT * ✓FROM tblSightings✓ ORDER BY ✓SightingID DESC✓	4	
1.2	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = true✓	4	
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = Yes✓		
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = On✓		
1.3	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers	6	
	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS TotalYears✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ 2010 ✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ year(Date())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
1.4	SELECT Animal✓, format(Avg(NumAnimals) ✓, "0.00"✓) AS [AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓	6	
	SELECT Animal✓, round(Avg(NumAnimals) ✓,2✓) AS AvgSighted✓ FROM tblSightings✓ GROUP BY Animal✓		
1.5	Input statement✓ DELETE✓FROM tblSightings✓ WHERE SightingID = +IntToStr(id)✓	4	
1.6	UPDATE✓ tblSightings✓ SET✓ Animal = "White Rhino"✓ WHERE Animal = "Rhino"✓	5	
	UPDATE✓ tblSightings✓ SET✓ Animal = "White Rhino"✓ WHERE Animal Like "%Rhino%"✓ **//OR** "%Rhino"		
1.7	SELECT SightingDate,Name,Surname✓ FROM tblSightings, tblRangers✓ WHERE tblSightings.RangerID = ✓ tblRangers.RangerID✓ AND Animal = "Elephant"✓ AND SightingDate > #30/04/2010#	6	
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal = "Elephant"✓ AND month(SightingDate) > 4		
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal Like "Elephant"✓ AND SightingDate > #30/04/2010#		
	TOTAL:	35	

ADDENDUM B

QUESTION 2 - DELPHI: OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:……	EXAMINATION NUMBER: ……

QUESTION 2 DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Define attributes for TCompetitor**: four private (1) fields, three integers named correctly (1) and one string (1)	**3**	
2.1.2	**Constructor:** name parameter (1) initialise all instance fields (1) remove the comment-signs from code in the 'spot'-methods (1)	**3**	
2.1.3	**calculatePoints:** (3) multiply counters by correct values and add results	**3**	
2.1.4	**totalAnimals:** returns integer (1) containing sum of counters (1)	**2**	
2.1.5	**getName function:** Correct name and return type (1), Correct field returned (1)	**2**	
2.1.6	**mostSpotted function:** returns String (1) two if .. with else-statement each to determine highest count (3)	**4**	
2.1.7	**toString:** name and new line (1) and appends string (1) with category-points (1) add last line(1) with tab (1)	**5**	
2.2			
2.2.1	Declare a single TCompetitor object (2) Initialise counters to zero (1) If file does not exist (1)display message and exit (1) if file exists then assignfile (1) open file to read from(1) Read first line outside of loop into name variable (1) Call constructor method of TCompetitor (1) using name as a parameter (1) loop with begin and end correctly placed (1) Inside loop: Read line from file (1), get animal name (1) and category- letter (1) from string. Use letter and compare result (1) call appropriate method to increase count based on result (3) to handle all valid categories. Else increase invalid counter and display animal name in invalid category (1). Display number of valid and invalid entries (1)	**20**	
2.2.2	Call toString method of TCompetitor object (1) to display information(1)	**2**	
2.2.3	Use name of competitor to construct filename (1) ready the file for writing to new file (1) Call methods to construct output (1) and write to file (1), close the file (1)	**5**	
	TOTAL:	**49**	

ADDENDUM C

QUESTION 3: DELPHI PROGRAMMING

CENTRE NUMBER: ……	EXAMINATION NUMBER: ……

QUESTION 3 DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
3.1	(2) Declare arrEntries array (1) Remove comment-signs from statements assigning strings to array	**3**	
3.2	**Option A** (1) Assign 10 to waterX and waterY (2) Heading and subheadings (1) Loop through given array Inside Loop: (2) Extract xPos and (2) yPos (2) Extract time (3) Calculate distance, (1) Round down (3) Display info in loop	**17**	
3.3	**Option B** (1) Enter the number of different types of animals (1) Loop from 1 to the number of different types (1) Inside loop: (1) Enter animal type (1) Enter the number in the group (1) Extract first 2 letters and assign to tag (2) Extract the last letter (2) Generate number in correct range (1) Validate even number (1) Add random number and hyphen to tag (1) Heading (1) Inner loop to add the unique number of animals in each group (2) Display the correct data aligned correctly inside inner loop	**16**	
	TOTAL:	**36**	

ADDENDUM D

QUESTION 1: JAVA – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	EXAMINATION NUMBER:……

QUESTION 1: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	SELECT * ✓FROM tblSightings✓ ORDER BY ✓SightingID DESC✓	4	
1.2	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = true✓	4	
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = Yes✓		
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = 1✓		
1.3	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers	6	
	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS TotalYears✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ 2010✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ year(DATE())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
1.4	SELECT Animal✓, format(Avg(NumAnimals) ✓, '0.00'✓) AS [AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓	6	
	SELECT Animal✓, round(Avg(NumAnimals) ✓,2✓) AS AvgSighted✓ FROM tblSightings✓ GROUP BY Animal✓		
1.5	Input statement✓ "DELETE✓FROM tblSightings✓ WHERE SightingID = "+id✓ // **or** "+id+""	4	
1.6	UPDATE✓ tblSightings✓ SET✓ Animal = 'White Rhino'✓ WHERE Animal = 'Rhino'✓	5	
	UPDATE✓ tblSightings✓ SET✓ Animal = 'White Rhino'✓ WHERE Animal Like '%Rhino%'✓		
1.7	SELECT SightingDate,Name,Surname✓ FROM tblSightings, tblRangers✓ WHERE tblSightings.RangerID =✓ tblRangers.RangerID✓ AND Animal = 'Elephant'✓ AND SightingDate > #30/04/2010#	6	
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal = 'Elephant'✓ AND month(SightingDate) > 4		
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal like 'Elephant'✓ AND SightingDate > #30/04/2010#		
	TOTAL:	35	

ADDENDUM E

QUESTION 2 - JAVA: OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:……	EXAMINATION NUMBER: ……

QUESTION 2 JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Define attributes for Competitor**: four private (1) fields, three integers named correctly (1) and one string (1)	**3**	
2.1.2	**Constructor:** name parameter (1) initialise all instance fields (1) uncomment statements give in the 'spot'-methods (1)	**3**	
2.1.3	**calculatePoints:** (3) multiply counters by correct values and add results to get total points	**3**	
2.1.4	**totalAnimals:** returns integer (1) containing sum of counters (1)	**2**	
2.1.5	**getName method:** Correct name and return type (1), Correct field returned (1)	**2**	
2.1.6	**mostSpotted method:** returns String (1) two if .. with else-statement each to determine highest count (3)	**4**	
2.1.7	**toString:** name and new line (1) and appends string (1) with category-points (1) add last(1) line with tab (1)	**5**	
2.2			
2.2.1	Declare a single Competitor object (1) Create a new File object (1) If file does not exist (1) display message and exit (1) if file exists then open file for reading (2) Read first line outside of loop (1) into name variable (1) Call constructor method of Competitor (1) using name as a parameter (1) and assign to object variable(1) initialise counters to zero (1) loop with braces correctly placed (1) Inside loop: Read line from file (1), get animal name (1) and category-letter (1) from string. Use letter and compare result (1) call appropriate method to increase count based on result (3) to handle all valid categories. Else increase invalid counter and display animal name in invalid category (1). Display number of valid and invalid entries (1)	**20**	
2.2.2	Call toString method of Competitor object (1) to display information (1)	**2**	
2.2.3	Use name of competitor to construct filename (1) ready the file for writing to new file (1) Call methods to construct output (1) and write to file (1), close the file (1)	**5**	
	TOTAL:	**49**	

ADDENDUM F

QUESTION 3: JAVA PROGRAMMING

CENTRE NUMBER: ……..	EXAMINATION NUMBER: …….		
QUESTION 3 JAVA – MARKING GRID			
QUESTION	**ASPECT**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	(2) Declare arrEntries array (1) Remove comment-signs from statements assigning strings to array	**3**	
3.2	**Option A** (1) Assign 10 to waterX and waterY (2) Heading and subheadings (1) Loop through given array Inside Loop: (2) Extract xPos and (2) yPos (2) Extract time (3) Calculate distance, (1) round down, (3) Display info in loop	**17**	
3.3	**Option B** (1) Enter the number of different types of animals (1) Loop from 1 to the number of different types (1) Inside Loop: (1) Enter animal type inside loop (1) Enter the number in the group inside loop (1) Extract first 2 letters and assign to tag (2) Extract the last letter (2) Generate number in correct range (1) Validate even number (1) Add random number and hyphen to tag (1) Heading (1) Inner loop to add the unique number of animals in each group (2) Display the correct data aligned correctly inside inner loop	**16**	
	TOTAL:	**36**	

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P2

NOVEMBER 2010

MARKS: 180

TIME: 3 hours

This question paper consists of 18 pages.

INSTRUCTIONS AND INFORMATION

1. This question paper consists of FIVE sections subdivided as follows:

SECTION A: Multiple-choice questions	(10)
SECTION B: Hardware and software	(58)
SECTION C: Applications and implications	(20)
SECTION D: Programming and development of software	(47)
SECTION E: Integrated scenario	(45)

2. Answer ALL the questions.

3. Read ALL the questions carefully.

4. Number the answers correctly according to the numbering system used in this question paper.

5. Write neatly and legibly.

SECTION A: MULTIPLE-CHOICE QUESTIONS

QUESTION 1

Various options are given as possible answers to the following questions. Choose the answer and write only the letter (A – D) next to the question number (1.1 – 1.10) in the ANSWER BOOK.

1.1 Which of the following does NOT break the copyright law?

A Downloading music from the Internet
B Downloading music made available by podcasters from the Internet
C Installing a copy of the school's single-user license on your home computer.
D Copying information from a web page into your assignment (1)

1.2 Open-source software is ...

A free ONLY if you are a registered company.
B software for which the source code is freely available.
C free for a trial period.
D software that cannot be affected by a virus. (1)

1.3 A digital … is a secure digital identity that certifies the identity of the holder on the Internet.

A certificate
B password
C signature
D username (1)

1.4 The most popular LAN-communication standard used today is ...

A TCP/IP.
B ADSL.
C Token Ring.
D Ethernet. (1)

1.5 Which one of the following is the ODD ONE OUT?

A Norton Antivirus
B Windows XP
C Linux
D MS-DOS (1)

1.6 People with the same interests who want to chat online will make use of ...

A a forum.
B e-mail.
C IRC.
D a blog. (1)

1.7 … is high-speed memory used for the temporary storage of data or instructions likely to be needed next by the processor.

A Virtual memory
B Cache memory
C Random access memory
D Read-only memory (1)

1.8 Which one of the following statements is NOT associated with green computing?

A Companies should refill toner cartridges rather than buy new ones.
B One of the advantages of using ICT is creating 'paperless' offices.
C Energy Star compliant hardware should be used as a rule.
D Companies should replace their computers regularly. (1)

1.9 A … port is a high-speed port most suited to connecting digital and video cameras to a computer.

A USB
B FireWire
C MIDI
D SCSI (1)

1.10 A technique used by intruders to make their network or Internet transmission appear legitimate to a victim is referred to as …

A phishing.
B spoofing.
C spam.
D a hoax. (1)

TOTAL SECTION A: 10

SCENARIO

A large, very successful computer company called Cosmo Computing has opened a new branch in a metropolitan city. The board of directors are currently conducting interviews with various potential candidates for the position of manager of the new branch.

SECTION B: HARDWARE AND SOFTWARE

QUESTION 2

Dube is a young, motivated and dynamic graduate who has three years of working experience in the Information Technology sector. He wants to migrate to the retail sector and is short-listed for the job of manager of the new branch. The panel has a list of questions that Dube will be required to answer at the interview.

2.1 A client has recently upgraded the operating system of her computer from Windows XP to Windows Vista. She needs to be advised regarding issues related to hardware and software.

2.1.1 The computer's external hard drive that used to work on the old operating system no longer works with the new operating system.

Give a possible reason why this may be so. (2)

2.1.2 The client was advised that in addition to the external hard drive, she should also partition the existing internal hard drive.

(a) What does it mean to divide a hard drive into partitions? (2)

(b) Name TWO advantages of partitioning a hard drive. (2)

2.1.3 The client observed that the operating system sometimes automatically detects and makes a new hardware or peripheral device ready for use without user intervention. What is the term that describes this concept? (1)

2.1.4 You need to assure the client that it is safe to disconnect a device and reconnect another device to the computer while it is on. What is the term that best describes this action? (1)

2.1.5 The client also complained that her computer has slowed down considerably since changing the operating system. Name TWO hardware components that might need to be upgraded in order to improve the performance of the computer. (2)

2.1.6 Someone advised the client that she should have a heat sink and a fan installed in her computer to prevent overheating. The client feels that having both will increase her expenses and she decides to purchase only one. Explain how each of these items prevents overheating of the processor. (2)

2.1.7 The client is undecided about whether to purchase a 17" LCD flat-screen monitor or a CRT-monitor. Give THREE reasons why LCD monitors are preferable to the older CRT-monitors. (3)

2.2 Schools are one of the biggest clients of Cosmo Computing. The company sets up computer laboratories and networks and offers a maintenance program thereafter. A teacher approaches the company about setting up a network at his school.

2.2.1 (a) Explain to the teacher the advantages of having a network. (3)

(b) The school is considering a client-server network. Give TWO reasons why a client-server network may NOT be such a good choice for the school environment. (2)

(c) One of the teacher's major concerns is security on the network. Learners should not be able to get unauthorised access to computers on the network. Explain to the teacher the duties of the network administrator regarding security. (2)

2.2.2 The school requires the computers in the computer laboratory to be linked in a network. The company recommends that the computers be connected using a star topology.

(a) Give the teacher a simple diagrammatical representation of a star topology. (3)

(b) What type of cable will be used to link the computers in the computer laboratory? Give TWO reasons for your choice. (3)

(c) Name TWO functions of a switch in a network. (2)

(d) The teacher would like to have a wireless network instead of one using cables. He wants WiFi to be installed.

(i) Explain the term *WiFi*. (2)

(ii) Give TWO reasons why WiFi is NOT suitable for a school computer laboratory. (2)

(iii) If the teacher insists on WiFi technology, name TWO devices that the school will have to purchase to facilitate WiFi technology. (2)

2.3 Cosmo Computing's manager needs to conduct workshops with his staff whenever there is an upgrade/update in any hardware or software.

2.3.1 Explain to the staff the difference between DRAM and DDR2 RAM. (2)

2.3.2 Briefly explain each of the following terms:

(a) Pipelining (3)

(b) Superscalar (1)

(c) Hyperthreading (2)

(d) RAID (2)

(e) Clock multiplication (2)

2.3.3 Devices such as smart phones, PDAs and other hand-held devices use embedded operating systems. Name an open-source multitasking operating system used in smart phones. (1)

2.3.4 Give a practical example to explain the purpose of each of the following:

(a) File compression utility software (2)

(b) File conversion utility software (2)

2.3.5 Explain how a 32-bit operating system differs from a 64-bit operating system. (2)

2.4 The national Department of Transport is also a client of Cosmo Computing. They have employees all over the country and have decided to use video conferencing to reduce costs.

2.4.1 The Department will require a web cam for video conferencing.

(a) Describe the function of a web cam. (1)

(b) Give an example of software that will make use of a web cam. (1)

2.4.2 Name ONE other hardware device that will be required to conduct video conferences. (Do NOT use web cams, CPUs, keyboards, mouses or monitors in your answer.) (1)

TOTAL SECTION B: 58

SECTION C: APPLICATIONS AND IMPLICATIONS

QUESTION 3: e-COMMUNICATION

The Internet department in the new branch is always the busiest. The reason is that most clients are not familiar with simple terminology and often need assistance. Dube has to assist with the requests of a vehicle sales consultant visiting the Internet department.

3.1 The sales consultant would like to create podcasts and publish them so that they can be downloaded by clients.

3.1.1 Explain to the consultant the term *podcasting*. (2)

3.1.2 Explain how podcasting will benefit the company. (2)

3.2 The sales consultant is confused with the terms IM and e-mail. Give TWO clear differences between *IM* and *e-mail* in table format. (4)

3.3 The sales consultant would like an alternative website with a .mobi domain.

3.3.1 What does the domain 'mobi' represent? (1)

3.3.2 Name ONE significant advantage of a mobi website. (1)
[10]

QUESTION 4: SOCIAL AND ETHICAL ISSUES

Cosmo Computing is concerned about their social responsibility to local communities. They would like to improve the computer literacy of the surrounding community. They also have old computers that are no longer being used that they would like to donate to a local school.

4.1 Name ONE type of software package that could be taught to novice computer users in the local community in order to make them more employable. Justify your answer. (2)

4.2 Name TWO ways in which the school can use the old computers for educational purposes. (2)

4.3 You inform the company that it is not socially responsible to donate very old computers to schools. Identify TWO potential problems of donating old computers to a school. (2)

4.4 The company decides to rather give the school a fully equipped new computer laboratory. Name TWO job positions that will be required in order to make this project a success. (2)

4.5 The company considers e-mailing all their clients about the project mentioned in QUESTION 4.4 inviting them to donate money to the project. Would you regard these e-mails as spam? Justify your answer. (2)
[10]

TOTAL SECTION C: 20

SECTION D: PROGRAMMING AND SOFTWARE DEVELOPMENT

QUESTION 5: ALGORITHMS AND PLANNING

Cosmo Computing has a division that does repairs and gives technical support to clients. The company requires software which will allow them to keep track of their clients' support requests as well as details of the technicians who deal with these requests. They have hired a software developer but would also like your input.

5.1 The software developer has created two tables **tblClients** and **tblJobs**:

tblClients

CustomerID	Name	Surname	ContactNumber
1	John	Smith	0821475869
2	Phumzile	Mbata	0714567895
3	Delia	Juniper	0847541225
4	Reyahd	Khan	0724578892

tblJobs

JobID	CustomerID	DateOfJob	Description	Technician	Rating	FullTime
1	1	2010/10/06	Data recovery	Lionel Torvalds	9	☑
2	1	2010/11/10	Software upgrade	Gill Bates	5	☐
3	2	2010/05/10	Re-install operating system	Gill Bates	5	☐
4	4	2010/12/07	Set up Internet	Lionel Torvalds	9	☑
5	2	2010/01/07	Set up Internet	Gill Bates	5	☐
6	3	2010/09/29	Remove virus	Lionel Torvalds	9	☐
7	3	2010/04/10	Re-install operating system	Steve Hobbs	8	☑

5.1.1 What is the purpose of the primary key in a database table? (1)

5.1.2 Identify the primary key in the table **tblClients**. (1)

5.1.3 The **CustomerID** field from **tblClient** is repeated in **tblJobs**. What is the purpose of also having the **CustomerID** field in the table **tblJobs**? (2)

5.1.4 The field-type for the **ContactNumber** field in the **tblClients** table has been specified as 'Text'. Explain how the data will be affected if the data type for this field is rather specified as a 'Number'. (2)

5.2 You suggest that the table **tblJobs** should be normalised and therefore be replaced by two separate tables (**tblTechnicians** and **tblJobsOnly**) as indicated below.

tblTechnicians

TechnicianID	Name	Rating	FullTime
1	Lionel Torvalds	9	☑
2	Gill Bates	5	☐
3	Steve Hobbs	8	☑

tblJobsOnly

JobID	CustomerID	DateOfJob	Description	TechnicianID
1	1	2010/06/10	Data recovery	(a)
2	1	2010/11/10	Software upgrade	(b)
3	2	2010/05/10	Re-install operating system	(c)
4	4	2010/12/07	Set up Internet	(d)
5	2	2010/01/07	Set up Internet	(e)
6	3	2010/09/29	Remove virus	(f)
7	3	2010/04/10	Re-install operating system	(g)

5.2.1 Name TWO advantages of normalising tables in general. (2)

5.2.2 In the suggested **tblJobsOnly** table the **TechnicianID** field is a foreign key. The **TechnicianID** field in each record contains the characters (a) to (g) at the moment but should contain values indicating the correct **TechnicianID** for each record. Use the **original tblJobs** table (on the previous page) and the **new tblTechnicians** table (above) to determine the correct **TechnicianID** for each record. Write down the letters (a) to (g) as well as the correct value for the **TechnicianID** field for each record. (3)

5.3 The company decides that they want to award performance bonuses to part-time technicians using the following criteria:

- **An amount of R300,00 for all part-time technicians with a rating of 7 or higher.**

The developer has provided the following **incorrect algorithm** as part of the solution to select the technicians who qualify for this bonus:

Algorithm

```
Line 1      IF RATING >= 7 OR FULLTIME = NO THEN
Line 2      BONUS ← 300
```

5.3.1 Use the table below to test the suggested **algorithm**. Write down the letters (a) to (e) and your answers to correctly complete the table. The first two columns contain the given values for the variables RATING and FULLTIME for three different technicians.

NOTE: Do NOT redraw the table.

VALUE OF **RATING**	VALUE OF **FULLTIME**	RESULT OF **RATING >= 7?**	RESULT OF **FULLTIME = NO?**	RESULT OF **RATING >= 7 OR FULLTIME = NO?**	VALUE OF **BONUS**
3	NO	FALSE	TRUE	(a)	
6	YES	(b)	(c)	(d)	
9	NO	TRUE	TRUE	(e)	

(5)

5.3.2 The results in the table reflect that the algorithm is incorrect and does not match the requirements of the initial criteria. Rewrite only Line 1 of the algorithm in order to produce the correct results. (1)

5.4 Cosmo Computing has stored the number of jobs they completed each month in an array called **JOBSTATS**. Each position represents the number of jobs completed in each month of the year. Position 0 refers to the number of jobs completed in January and position 11 refers to the number of jobs completed in December. A visual representation of the array is given below.

JOBSTATS

	[0]	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**[10]**	**[11]**
Value	64	57	43	36	69	86	94	72	83	59	67	75

For example: 43 jobs were completed in March (position 2). The company wants to calculate the average number of completed jobs for the last 6 months of the year. The software developer has compiled the following algorithm:

```
LINE
1       TOTAL_JOBS ← 0
2       COUNTER ← 6
3       AVERAGE_JOBS ← 0
4       LOOP UNTIL COUNTER > 12
5               TEMP ← JOBSTATS[COUNTER]
6               TOTAL_JOBS ← TOTAL_JOBS + TEMP
7               COUNTER ← COUNTER + 1
8       END LOOP
9       AVERAGE_JOBS ← TOTAL_JOBS / COUNTER
10      DISPLAY AVERAGE_JOBS
```

5.4.1 Suggest a suitable data type for the variable COUNTER. (1)

5.4.2 When the program is executed, a run-time error occurs.

(a) The run-time error indicates that line 5 of the given algorithm caused the error. Rewrite line 4 of the algorithm in order to correct the error. (2)

(b) Explain what a *run-time error* is in general. (1)

5.4.3 Assume that the error in line 4 has been corrected. When the program is executed, an incorrect value for the variable AVERAGE_JOBS is displayed.

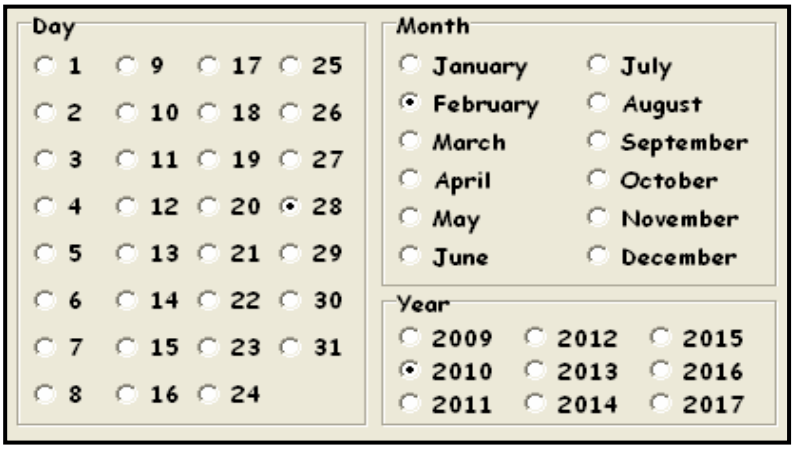
(a) What type of programming error is generally associated with incorrect output? (1)

(b) Explain why the algorithm displays an incorrect output for AVERAGE_JOBS. (2)

(c) Rewrite a **single line** from the algorithm above which will fix the error. Indicate the line number of the statement that you are rewriting in your answer. (2)

5.5 The clients register on the website of Cosmo Computing and get free advice on computer problems. The software developer compiled the following four possible interfaces for capturing the date of birth of the client:

Website A (Radio Buttons)



Website B (Text Box)

Website C (Single Drop Down List)

Website D (Multiple Drop Down Lists)

5.5.1 Which ONE of the websites (A to D) provides an input component that makes it almost impossible for a user to select/enter an invalid date? (1)

5.5.2 The correct test data should be used to test whether program code delivers the correct output in all cases. Give a suitable example of a date as input in each of the following instances where the date will qualify as the following:

(a) Normal test data (1)

(b) Erroneous test data (1)

(c) Extreme test data (1)

5.5.3 Using dates as examples, explain the difference between *invalid* and *incorrect* data. (2)

5.5.4 Study the input interface in Website B on the previous page. Name ONE way that the programmer can ensure that the user is guided to enter the date using the correct format. (1)

5.5.5 Other than allowing users to select incorrect dates, name ONE disadvantage of the interface in Website C on the previous page. (1)

5.5.6 An error message should be provided when an invalid date is entered. Give THREE general guidelines for compiling good error messages. (3)

5.6 The manager admits that he does not know much about OOP. You have compiled the following class diagram representing a **Technician** object to explain certain OOP concepts.

Study the class diagram. NOTE: The '-' sign indicates a private declaration and the '+' sign indicates a public declaration.

Technician	
Fields	**Methods**
- fTechnicianID:String - fName:String - fRating:int - fFullTime:boolean	+ constructor() + constructor(ID, name) + getID():String + getRating():int + getFullTime():boolean + setRating(rating) + toString()

5.6.1 Explain the difference between *public* and *private declarations* in terms of OOP. (2)

5.6.2 There are two constructor methods in the class diagram above, viz.

(a) +constructor()

(b) +constructor(ID, name)

Give the generic name for EACH type of constructor given above. (2)

5.6.3 Write down ONE example of each of the following methods from the given **Technician** class diagram:

(a) Accessor method (1)

(b) Mutator method (1)

5.6.4 Explain the purpose of a '*toString*' method in a class. (2)

5.6.5 The programmer needs to write a method to return the name of a technician. Write a statement for the Methods-column of the class diagram that will indicate this return method. (2)

TOTAL SECTION D: 47

SECTION E: INTEGRATED SCENARIO

QUESTION 6

After a successful interview Dube has been appointed as the manager of the new branch of Cosmo Computing. He has to guide other team members so that they can manage this branch successfully as a team and ensure client satisfaction.

6.1 It has been suggested that Radio Frequency Identification (RFID) tags are placed on the items in the store to read information directly from the items when purchased or when stocktaking is done. Dube has to investigate this option.

6.1.1 Name TWO specific features of RFID. (2)

6.1.2 Other than the tags on items in the store name TWO other practical applications of where RFID can be used. (2)

6.1.3 Name TWO electronic devices, other than RFID tags, that are widely used to capture data directly from source documents. (2)

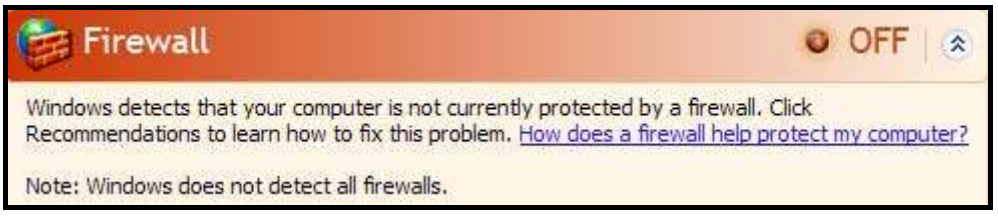
6.2 WiMax is a possible alternative for clients to connect to the Internet.

6.2.1 What medium of transfer does WiMax use? (1)

6.2.2 What type of network is a WiMax network – WAN/MAN/LAN? (1)

6.2.3 An Internet service provider (ISP) is required for any network connection to the Internet. Name THREE guidelines for choosing an ISP. (3)

6.3 One of the clients at the new branch is experiencing security problems on their computer system. Dube investigates the matter. The following information is displayed by a utility program:



6.3.1 What is a *utility program*? (3)

6.3.2 What kind of protection is offered by a firewall? (2)

6.3.3 The following information has also been displayed:

(a) What is a *computer virus*? (3)

(b) Name THREE common symptoms that indicate that a computer system has been infected by a computer virus. (3)

6.4 Clients often lose important data due to laptops being stolen, power failures, etc. Give THREE hints to clients to prevent the loss of data as a result of unforeseen circumstances. (3)

6.5 The performance of their computer is often the most important feature that clients are concerned about.

6.5.1 The CPU can play a significant role in the performance of a computer. It consists of a control unit and an arithmetic logic unit (ALU) as well as an instruction set.

(a) Name TWO well-known and popular manufacturers of processors used in personal computers. (2)

(b) Explain what the *instruction set* of a CPU is. (2)

(c) Most processor chip manufacturers now offer quad-core processors. What is the main characteristic of a quad-core processor? (2)

6.5.2 Communication between components in a computer system should be well managed. The chipset of a modern computer system has a North bridge and South bridge managing the transfer of data.

(a) Which bridge controls the Front Side Bus (FSB)? (1)

(b) Briefly explain the function of the Front Side Bus. (1)

6.5.3 Storage devices can play a role in the overall performance of a computer system.

(a) The performance of storage devices, such as hard disks, can be improved by running the disk defragmenter. What does this software do? (2)

(b) A hard disk is managed by a disk controller such as SCSI or SATA. Name TWO advantages of using a SCSI disk controller. (2)

6.6 One of the clients works with highly confidential files all the time. The client is concerned about the privacy of information on their computer system. Dube advises the client to make sure that no unauthorised access takes place. He advises him to apply strict identification and authentication verification.

6.6.1 One way of applying identification verification is to enter a user name and a password to log into a program. Give TWO hints to users about selecting passwords that will not be easy to decipher. (2)

6.6.2 Dube suggests that the client considers installing the device shown below for identification verification:

(a) What is the term that describes the use of personal characteristics to authenticate the identity of a person? (1)

(b) Name TWO other personal characteristics that are often used for authentication verification purposes. (2)

6.6.3 The client needs to transfer confidential information to clients via the Internet on a daily basis. Dube suggests that he uses the Secure Sockets Layer (SSL) protocol.

(a) What is a *protocol in a network environment*? (2)

(b) What technique does the SSL protocol apply to ensure that the data is transferred in a secure way? (1)

TOTAL SECTION E: 45
GRAND TOTAL: 180

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P2

NOVEMBER 2010

MEMORANDUM

MARKS: 180

This memorandum consists of 15 pages.

SECTION A: MULTIPLE-CHOICE QUESTIONS

QUESTION 1

1.1 Which of the following does NOT break the copyright law? **B**✓ (1)
(Downloading music made available by podcasters from the internet)

1.2 Open-source software is ... **B**✓ (1)
(software for which the source code is freely available)

1.3 A digital … is a secure digital identity that certifies the identity of the holder on the Internet. **A or C**✓ (1)
(certificate) or (signature)

1.4 The most popular LAN-communication standard used today is … **D**✓ (1)
(Ethernet)

1.5 Which one of the following is the ODD ONE OUT? **A or C or D**✓ (1)
(Norton Antivirus) of (Linux) of (MS – DOS)

1.6 People with the same interests who want to chat online will make use of **C**✓ (1)
(IRC)

1.7 … is high speed memory used for the temporary storage of data or instructions likely to be needed next by the processor. **B**✓ (1)
(Cache memory)

1.8 Which one of the following statements is NOT associated with green computing? **D**✓ (1)
(Companies should replace their computers regularly.)

1.9 A … port is a high speed port most suited to connecting digital and video cameras **B**✓ (FireWire) (1)

1.10 A technique used by intruders to make their network or internet transmission appear legitimate to a victim is referred to as **B**✓ (spoofing) (1)

TOTAL SECTION A: 10

SECTION B: HARDWARE AND SOFTWARE

QUESTION 2

2.1 2.1.1 She would be required to install the **drivers** for the new software✓✓OR (2)
The hard drive could be formatted using NTFS which has security implications which means access is restricted.

2.1.2 (a) When the hard drive is partitioned, it would appear as if there is more than one hard drive. ✓✓ (2)
OR When one physical hard drive is 'divided' into several logical drives.

(b) Any TWO: ✓✓ (2)
- Different operating systems can be installed
- Better security can be maintained
- Better for backup procedures if the data is on one partition and the programs on another.
- The second partition is less prone to be infected by a virus

2.1.3 Plug and Play✓ (1)

2.1.4 Hot swappable✓ or Hot pluggable (1)

2.1.5 Any TWO: ✓✓ (2)
- Processor or CPU
- RAM or memory
- Hard drive
- Graphics card

2.1.6 Heat Sink: Any ONE:✓ (2)
- Directs the heat away from the processor (or similar description)
- Absorbs and ventilates heat produced by the processor
- Increases the surface area for heat exchange

Fan: blows the hot air out of the box✓

2.1.7 Any THREE: ✓✓✓ (3)
- More compact, less bulky **/** smaller **/** take up less space
- Quality of the display is much better / better resolution / sharper images
- Produces less heat
- Reduces power consumption
- Easily accessible / more available than CRT screens, support (maintenance services) is more available
- Area of visibility is smaller (images only visible when in front of the screen) / information on screen more private
- Flicker-free images / no refreshing required

2.2 2.2.1 (a) Any THREE: ✓✓✓(Explanations were required) (3)

- Sharing of hardware
- Sharing of software
- Saves time (reason **must** be provided e.g. faster to install software on all the computers / faster distribution of information)
- Centralisation of data
- Cost effective (reason **must** be provided – other reason than already mentioned e.g. sharing an Internet connection
- Improves communication
- Entertainment (provide an example e.g. playing LAN games)
- Making of backups is easier
- Improves control in terms of rights and privileges / software installed / access
  Or an explanation of any other valid advantage of a having a network

b) Any TWO: ✓✓ (2)

- Need a technician to maintain the network / not easy to maintain / need a specialist to be responsible for maintaining the network
- Requires a network-oriented operating system / requires network software
- Cost **with** a valid reason.
- The network is down when the server is down.

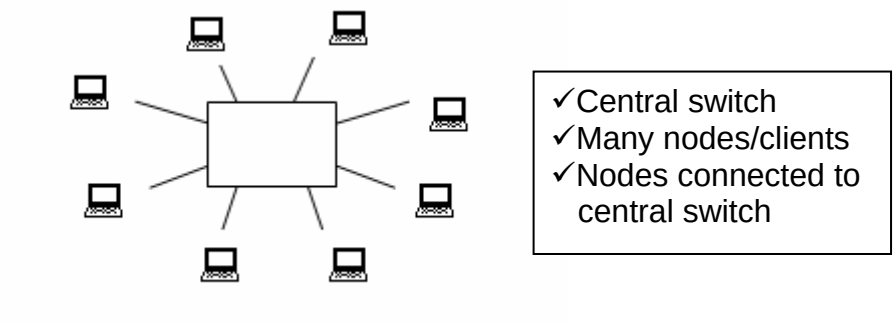
OR other correct answers – not repeating any previous answers

(c) Any TWO: (regarding preventing unauthorised access)✓ ✓ (2)

- Setup of users' rights and privileges to establish privacy and security for users
- Monitor suspicious activities on the network
- Managing accounts or delete accounts or creates accounts Set up passwords on accounts
- Controlling Internet access

NB – Do not allocate marks when answers repeat the content of previous answers.

2.2.2 (a) Labels are optional. If provided the labels must be correct. (3)



✓Central switch
✓Many nodes/clients
✓Nodes connected to central switch

(b) UTP cables ✓/Unshielded twisted pair/Twisted pair/CAT 5 or CAT 5e (accept from CAT 3) (3)
ANY TWO✓✓
Cheaper
Easy to set up
High speed over short distances
OR
Fibre Optic Cables✓
Any TWO:✓✓
Less susceptible to EMI
High speed data transfer
Less susceptible to attenuation

(c) Any TWO: ✓✓ (2)
- Connects several devices in a network.
- Transmits signals/Amplifies and transmits signals
- Can detect errors and isolate the error so that the network can still be functional
- Can send data to the required destination on the network or chooses the correct / best path to follow OR Intelligent pathing OR controls / reduces traffic OR manages the bandwidth
- Amplifies signals

(d) (i) The name of a popular wireless networking technology that uses radio waves to provide wireless high-speed Internet and network connections (2)
✓ wireless / radio waves
✓ connect two or more devices / network / communicate

(ii) Any TWO: ✓✓ (2)
- Speed/bandwidth can be affected with many users
- Security/uncontrolled access
- WiFi is most suited for mobile devices, schools have
- standard computer equipment that is fixed in the room

(iii) Any TWO: ✓✓ (2)
- WiFi cards
- (Wireless) router
- Wireless Access point
- Antenna

2.3 2.3.1 DRAM – slower than DDR2✓✓ (2)
OR
DDR2 – allows transfer of data twice as fast per tick of the system clock as DRAM

2.3.2 (a) Pipelining – occurs when the CPU is able to read and execute new instructions✓ from memory/ which are loaded✓ before the instruction that is busy being processed is completed. ✓ (3)

(b) Superscalar – Any ONE: ✓ (1)
- Having more than one pipeline
- Processor can execute more than one instruction per clock signal.

(c) Hyperthreading – allows the operating system to view a single processor ✓as having TWO processors. ✓ (2)

OR

One CPU can mimic the power of two processors.

NB. No marks for explaining multithreading.

(d) Any TWO facts: ✓✓ (2)
- RAID – stands for Redundant Array of Inexpensive / Independant Disks
- RAID is a number of disks grouped together by a special controller and treated as one big disk.
- It can be used to improve data reliability as it duplicates or spreads data across the disks
- The use of the word "backup" may be accepted if used in the context of "restoring data or information".

NOTE: Also accept more technical facts if the facts are correct

(e) Clock multiplication: Allows the CPU to run at faster speeds✓ than the supporting motherboard. ✓ (2)
OR For every tick of the system clock, the CPU processes more than one instruction / at a faster rate
OR The CPU's own clock ticks faster than that of the motherboard / generates at least 10 ticks for very tick of the system clock
OR Multiplying the ticks of the system clock with a factor to determine the processing speed of the CPU.

2.3.3 Any ONE: ✓ (1)
Symbian OS / Android / WebOS / Linux

2.3.4 (a) File compression utility – makes the size of the file smaller ✓ e.g. when a file is too big to fit on a storage device such as a flash disk.✓ (2)
OR any other sensible and correct example.

(b) File conversion utility – changes the format of a file ✓e.g. when a Windows Audio file must be converted to a MP3 file to be able to play the music on your computer✓ (2)
OR any other sensible and correct example.

2.3.5 A 32-bit operating system uses 32-bit architecture/registers/ computer words✓ and a 64-bit operating system uses 64-bit architecture/registers/computer words✓ (2)
OR 64-bit OS can address more memory than 32-bit OS

2.4 2.4.1 (a) a device that is used to capture video and still images. ✓ (1)
OR a device that is used to capture video images.
OR uploads an image onto the Internet.

(b) ANY ONE ✓ (1)
Skype / NetMeeting / Windows Media Player / Cam Studio / DumDum
OR any other recognized software that will make use of a web cam.

2.4.2 Any ONE: ✓ (1)
Microphone / Modem / Speakers / Network Card / HeadSet

TOTAL SECTION B: 58

SECTION C: APPLICATIONS AND IMPLICATIONS

QUESTION 3: e-COMMUNICATION

3.1 3.1.1 Podcasting is a method of distributing recorded audio ✓in a format✓ (usually an MP3 file) suitable for devices such as iPods. These audio files are published as audio files on the Internet and can be downloaded to a computer or portable media player. (2)

3.1.2 Any TWO: ✓✓ (2)

- Published on the Internet which will reach more potential clients.
- Can advertise the company in a fun way.
- Cost-effective way of advertising.
- Can be used for educational purposes.
- Information can be added to ensure client satisfaction.

3.2 (4)

IM	E- Mail
Instant/live/ Synchronous ✓	Does not have to be live/ ASynchronous ✓
Does not allow for attachments ✓	An attachment is allowed✓
Initiated by mutual consent	Not necessarily mutual consent
Users must have the same IM service	Does not have to have the same service
The size of messages are limited	The size of messages are not ot limited

3.3 3.3.1 These spaces/sites have been specially designed for viewing on mobile devices ✓ /hand held devices such as cellphones. (1)

3.3.2 Any ONE: ✓ (1)
The site is accessible to a larger group of people
A computer is not required.
Access from a mobile device

[10]

QUESTION 4: SOCIAL AND ETHICAL ISSUES

4.1 Any ONE software package suitable for the business world✓ (2)

- E-mail
- Word Processor
- Spreadsheet
- Web Browser

Appropriate motivation for package selected related to a working environment.✓

4.2 Any TWO: ✓✓ (2)

- Learners can take apart computers and learn about hardware
- Teachers can set up a display of computer parts and use those for instruction purposes
- Internet Access
- Running tutorial software
- Computer Literacy

OR ANY other reason relating to education

4.3 Any TWO: ✓✓ (2)

- Old computers run old software – no support.
- Learners will be trained in old software packages not being use.
- Cannot source replacement hardware easily.
- Computers will be slow and frustrate learners.
- Drivers may not be available.
- Funds may be required for upgrading purposes.
- More expensive to maintain than new computers.
- Does not support green computing as these computers may contain harmful substances.
- Difficult to dispose off old computers

4.4 Any TWO: ✓✓ (2)

- Network Administrator
- Computer Trainer/Teacher
- IT Technician / Specialist
- Project Manager

4.5 Yes/No✓ NOTE: Award 0 for a Yes/No answer with no justification (2)
Justification✓

Yes
Email is a bulk email and clients might not have requested to receive emails about outreach projects

OR

No
The project is a worthy cause and is not used to make a profit for the company

[10]

TOTAL SECTION C: 20

SECTION D: PROGRAMMING AND SOFTWARE DEVELOPMENT

QUESTION 5: ALGORITHMS AND PLANNING

5.1 5.1.1 A primary key is a field which has a unique value for each record in the table OR uniquely identifies a record in the table ✓ (1)

5.1.2 CustomerID✓ (1)

5.1.3 It is a used to link the TWO tables ✓✓OR used to create a relationship between the two tables (2)

5.1.4 Any ONE: ✓✓ (2)
- Numbers will not allow for the leading zeros
- The size of most numbers is not sufficient to store the full telephone number
- The format of all text cannot be converted to a number format

5.2 5.2.1 Any TWO: ✓✓ (2)
- Less duplication of data (e.g. repeating technician details)
- Prevents update anomalies (e.g. if a technician's details are changing the change only happens in one place)
- Prevents delete anomalies (e.g. all related data will be deleted when a record is deleted)
- Easier to query database

5.2.2 Apply negative marking (subtract 1 mark for each incorrect answer up to a maximum of 3 marks) (3)
(a) 1
(b) 2
(c) 2
(d) 1
(e) 2
(f) 1
(g) 3

5.3 5.3.1 (a) TRUE✓ or YES (5)
(b) FALSE ✓or NO
(c) FALSE ✓or NO
(d) FALSE✓ or NO
(e) TRUE✓ or YES

5.3.2 IF RATING >= 7 **AND** ✓FULLTIME = NO THEN (1)

NOTE: Learners only had to replace the OR-operator with an AND-operator but if the conditions were changed, the entire statement must test for the conditions correctly

5.4 5.4.1 Int/Integer/Byte/ShortInt/Long/LongInt/Word (1)

5.4.2 (a) LOOP UNTIL COUNTER = 12✓✓ (2)
OR
LOOP UNTIL COUNTER > 12-1
OR
LOOP UNTIL COUNTER > 11
OR
LOOP WHILE COUNTER <= 11
OR
LOOP WHILE COUNTER < 11
OR
LOOP WHILE COUNTER >=12
OR
LOOP FROM 6 TO 11

(b) An error which interrupts the executing of a program✓ OR Terminates the running of the program. (1)

5.4.3 (a) Logical error✓ (1)

(b) The value of counter would be incorrect in Line 9✓✓ (2)
OR
LINE 9 should be
AVERAGE_JOBS ← TOTAL_JOBS / **6**
NOTE: COUNTER to be divided by 2 OR (COUNTER - 6)

(c) LINE 9 should be (2)
AVERAGE_JOBS ← TOTAL_JOBS / **6** ✓✓
NOTE: COUNTER to be divided by 2 OR (COUNTER - 6)

An incorrect answer from 5.4.2(a) must be followed through for Question 5.4.3 (c).

5.5 5.5.1 Website C✓ (1)

5.5.2 (a) Normal: 28/02/2010✓ (or similar data well within the acceptable range of data for the application) (1)
(b) Erroneous: 246734 or some Text✓ (or similar data that is clearly not within the acceptable range of data for the application) (1)
(c) Extreme: 31/12/5943✓ (or similar data that will test extreme values) (1)

5.5.3 Invalid – Value not in a required range – can usually be validated using programming code e.g. select 30 as days but month as 2 – not allowed ✓ (2)

Incorrect – data is allowed and meets the requirements for being valid, but the data is not correct e.g. The date to be entered is 30 March but the user enters the date 20 March ✓

5.5.4 Any ONE: ✓ (1)
- Provide a label / tooltip text with the format
- Use Input mask / Use a masked edit box
- Provide an example of the format / contextual help

5.5.5 Website C: Could take user too long to find the correct date✓ (1)
OR
Programmer would need to constantly change the range of years as time goes on.

5.5.6 Any THREE: ✓✓✓ (Regarding the construction of the error message. NOT where it has to be displayed or how long etc.) (3)
- Clearly specify why the error occurred
- Give a clear description of the error.
- Indicate where/when the error has occurred.
- Include a solution to the problem.
- Make the error message easy to read and understand.
- Do not use technical terms/abbreviations.

5.6 5.6.1 Public means the entity can be accessed from outside the class ✓ and private only from within the class/unit where the class is defined✓ (2)

5.6.2 (a) default constructor✓ (1)
(b) parameterised constructor✓ (1)

5.6.3 (a) Any ONE of the get methods✓ (1)
(b) setRating✓ (1)

5.6.4 To provide code which describes the values of the object attributes✓ used for display/output purposes✓ (2)
OR
Combines fields into a single string

Delphi Candidates only:
Function used to convert numbers to a string value

5.6.5 + getName():String✓✓ (2)
OR any descriptive variable that describes the technician with the correct data type

TOTAL SECTION D: 47

SECTION E: INTEGRATED SCENARIO

QUESTION 6

6.1 6.1.1 Any TWO: ✓✓ (2)

- RFID tags have memory chips that can store information and communicate/radio signals used with a computer system.
- Can be embedded in glass, labels or cards.
- No direct contact required.
- Not transmitting all the time.
- Does not need a power source.

6.1.2 Any TWO: ✓✓ (2)

- Identify/Tracking times of runners in a marathon,
- tracking location of soldiers,
- tracking airline baggage or stolen goods,
- tracking of library books,
- tracking payment as vehicles pass through booths at toll gate systems,
- access control to areas,
- track goods being manufactured.

6.1.3 Any TWO: ✓✓ (2)
Scanners, bar code readers, magnetic-strip readers, magnetic-ink character recognition readers.

6.2 6.2.1 Radio waves✓ (1)

6.2.2 WAN OR MAN ✓ (1)

6.2.3 Any THREE: ✓✓✓ (Or any other correct guideline) (3)

- Enough bandwidth so that the connection is fast enough
- Affordable
- Host website at no additional cost
- Provide several types of connections
- Is there a limit on the volume of data that you are allowed to download
- Is the connection reliable – not offline most of the time
- Good reputation
- Provide protection against viruses and spam.

6.3 6.3.1 Type of system software / program✓ that allows a user to perform maintenance-type tasks ✓ usually related to managing a computer✓, its devices or its programs (3)

6.3.2 Any ONE✓✓ (2)

- A firewall helps protect your computer by preventing unauthorized access to the network from the Internet or another network.
- Prevents ports from being illegally used
- Prevents programs from illegally communicating with the computer.

6.3.3 (a) A computer virus is a potentially damaging computer program✓ that affects or infects a computer negatively✓ by altering the way the computer works without the user's knowledge or permission ✓OR copies itself. (3)

(b) Any THREE: ✓✓✓ (3)
- Screen displays unusual messages
- Unusual sounds play
- Available memory is less than expected
- Existing programs and files disappear
- Files become disrupted
- Programs or files do not work properly
- Unknown programs or files appear mysteriously
- System properties change mysteriously
- Operating system runs much slower than usual
- Warning from your anti virus program

6.4 Any THREE: ✓✓✓ (3)
- Make backups of data regularly
- Make use of an UPS to give you time to save data in the event of a power failure
- Save your work regularly while you work
- Make backup on another device such as a CD or DVD (not your laptop) and keep it somewhere safe
- RAID
- Updated your anti virus program regularly

Any other hint that make sense.

6.5 6.5.1 (a) Any TWO: ✓✓ Intel, AMD, Motorola, IBM, Transmeta (2)

(b) Basic set of instructions that the CPU/computer ✓can recognize and execute. ✓ (2)

(c) Consists of four cores/processors✓ combined on one chip✓ (2)

6.5.2 (a) North bridge✓ (1)

(b) Transfers data and instructions between the CPU and RAM✓ (1)
OR
Connects the CPU and RAM

6.5.3 (a) Reorganises the files ✓so that file segments are placed together/in continuous clusters on the hard drive✓ (2)

(b) Can support more than 2 peripheral devices✓ (2)
Fast data transfer rates ✓

6.6 6.6.1 Any TWO: ✓✓ (2)

- Longer passwords are better than shorter passwords
- Do not choose obvious passwords such as your surname
- Use a variety of different characters
- Use a variety of uppercase and lowercase letters

6.6.2 (a) Biometrics✓ (1)

(b) Any TWO: ✓✓ (2)
Face, handprint, eye, voice, DNA

6.6.3 (a) A set of rules governing communications✓between two or more computers/nodes/points✓ (2)

(b) Encryption✓ (1)

TOTAL SECTION E: 45
GRAND TOTAL: 180