basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2011

MARKS: 120

TIME: 3 hours

This question paper consists of 32 pages, 3 addenda and an information sheet.

INSTRUCTIONS AND INFORMATION

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

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 which network folder.

QUESTION 1

Delphi:	**Java:**
DamsDB.mdb	Dams.java
Question1_P.dpr	DamsDB.mdb
Question1_P.res	tblDams.txt
Question1_U.dfm	tblTowns.txt
Question1_U.pas	TestQuestion1.java
tblDams.txt	
tblTowns.txt	

QUESTION 2

Delphi:	**Java:**
uHousehold.pas	Household.java
Question2_P.dpr	TestQuestion2.java
Question2_P.res	
Question2_U.dfm	
Question2_U.pas	

QUESTION 3

Delphi:	**Java:**
Data.txt	Data.txt
Question3_P.dpr	TestQuestion3.java
Question3_P.res	
Question3_U.dfm	
Question3_U.pas	

If you received a disk (CD or stiffy) containing the files above, 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 question number 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 use the Java API files. You may NOT use any other resource material.**

9. At the end of this examination session you must hand in the disk or CD with all 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. Make printouts of the programming code of all the programming questions you have done.

11. All printing of the programming questions that you have done 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

Water is one of the most essential commodities required for the survival of human beings, plants and animals. The Department of Water Affairs is embarking on an intensive campaign to help save water. Many measures and programmes have been put in place to help bring about awareness on how to use water sparingly.

QUESTION 1: DELPHI PROGRAMMING AND DATABASE

The Department of Water Affairs has created a database named **DamsDB** containing information on all the dams in the country and the towns to which they supply water. An incomplete program has been developed to process queries on the data in the **DamsDB** database. Your task will be to complete this program.

The database named **DamsDB**, as well as an incomplete Delphi project named **Question1_P.dpr**, are saved in the folder named **Question1_Delphi**.

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

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

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

Do the following:

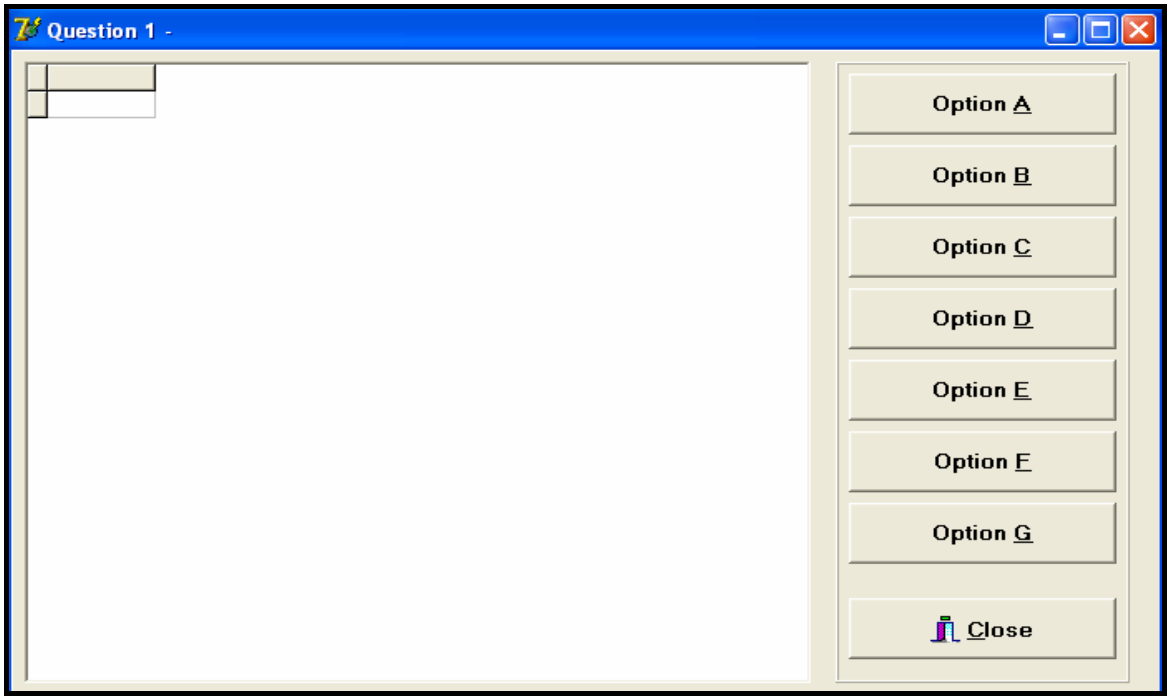
- Rename the folder **Question1_Delphi** as **Question1_X**, where X should be replaced with your examination number.
- Open Delphi and then open the file **Question1_P.dpr** in the folder **Question1_X**. The program displays eight buttons as well as a DBGrid that will be used as an output component (see example on the 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 main unit as **Question1_UXXXX** (where XXXX must be replaced with 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 with the last FOUR digits of your examination number).
- The program should be able to connect to the database named **DamsDB**. When you do QUESTION 1.1 (on the next page) and you find that the connectivity is not in place, use the steps supplied in **ADDENDUM C** to establish connectivity with the database.

NOTE: If your program cannot connect to the database, make sure that the database file **DamsDB** is in the same folder as your program. If this is not the case, copy the database file **DamsDB** into 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 program.

NOTE: If you still cannot establish connectivity with the database 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 unit named Question1_UXXXX.

When you execute the program, the interface below will be displayed. An error will be displayed when you click the buttons, due to the incomplete SQL statements.



Do the following:

Complete the SQL statements in **Question1_UXXXX.pas** for each button, as indicated in QUESTIONS 1.1 to 1.7 below. The code to execute the SQL statements and to display the results in the DBGrid has been given to you. You only need to complete the SQL statements and some input statements, as required in the **Question1_UXXXX** unit.

1.1 The Department of Water Affairs wants a list of all the dams in the country, sorted according to the height of the dam walls from the lowest to the highest. Complete the code for the **Option A** button by formulating an SQL statement to display **all the details** of dams stored in the **tblDams** table, sorted as required.

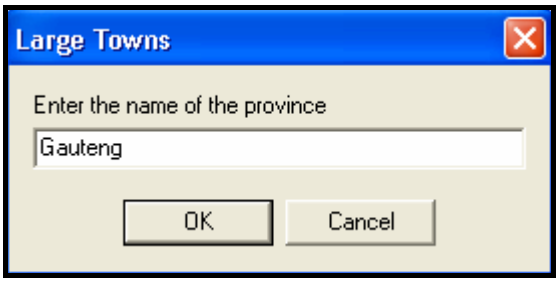
Example of the output for the first seven records:

DamID	DamName	River	YearCompleted	DamLevel	Capacity	HeightOfWall
83	Lake Mzingazi Dam	Mzingazi River	1942	19678	37000	8.2
146	Vaal Barrage	Vaal River	1922	48897	56712	10.3
34	Douglas Weir	Vaal River	1977	5910	16700	10.6
152	Voëlvlei Dam	Voëlvlei River	1971	131302	158600	10.6
41	Emmarentia Dam	Braamfontein Spruit	1912	151	250	11.4
68	Klipdrif Dam	Loop Spruit	1918	4629	13300	12.2
148	Vaalharts Storage Weir	Vaal River	1936	24105	48700	12.5

: (3)

1.2 One of the main concerns is large urban towns. The Department wants a list of all towns in a particular province that have a population exceeding 100 000. Complete the code for the **Option B** button by asking the user to enter the name of the province. Formulate an SQL statement to display the **TownName** and **Population** of all the towns that have a population exceeding 100 000 in the designated province.

Example of the input and output of all the towns in **Gauteng** with a population exceeding 100 000:



TownName	Population
Johannesburg	1975500
Tshwane	1473800
Vanderbijlpark	338000

(6)

1.3 An audit of the dams is taking place and additional information (not stored in the table) is required. You must write a query to display the age of each dam, as well as the current water level of each dam, as a percentage of its capacity. The age of a dam is calculated by subtracting **YearCompleted** from the current year. Call this field **Age**. The current water level of a dam as a percentage of its capacity can be calculated using the fields **DamLevel** and **Capacity**. Call this field **Percentage** and round it down to ONE decimal place. Complete the code for the **Option C** button by formulating an SQL statement to display the **DamID**, **DamName** and the two calculated fields.

Example of the output for the first seven records:

DamID	DamName	Age	Percentage
1	Albasini Dam	59	71.1
2	Albert Falls Dam	35	49.4
3	Allemanskraal Dam	51	73.7
4	Alphen Dam	21	33.9
5	Armenia Dam	57	54.3
6	Beervlei Dam	54	58.8
7	Berg River Dam	4	88.4

: (7)

1.4 The Department of Water Affairs considers any town with water restrictions to be a 'critical town' and wants to know how many critical towns there are in each province. Complete the code for the **Option D** button by formulating an SQL statement that will display the **Province** and a **calculated field** for **the total number of critical towns** in that province. Name the calculated field **CriticalTowns**.

Example of the output:

Province	CriticalTowns
Eastern Cape	11
Free State	6
Gauteng	3
KwaZulu-Natal	11
Limpopo	7
Mpumalanga	11
North West	5
Northern Cape	3
Western Cape	15

(5)

1.5 Due to the fact that the Vaal River flows through a number of provinces, the Department needs to know which provinces would be affected, should the Vaal River be contaminated by pollution. A province is supplied with water by the Vaal River if the dam that supplies a town in the province with water, receives water from the Vaal River. Complete the code for the **Option E** button by formulating an SQL statement to display the names of all the provinces that are supplied with water by the Vaal River. The name of each province should appear in the list only ONCE.

Example of the output:

Province
Free State
Gauteng
Mpumalanga
North West
Northern Cape

(7)

1.6 Some analysts have indicated that the **North West** province will experience severe droughts in the coming years. They have recommended that water restrictions be imposed on all towns in this province, which means they will all become critical towns. Complete the code for the **Option F** button by formulating an SQL statement that will **update** the records of all towns in the North West province to show which towns have water restrictions.

Example of the output (on the next page):

HINT: Run **Option D** to verify that the records have been updated. There should be 13 critical towns in the North West province after Option F has been executed. (4)

1.7 The risk of flooding has been assessed and it is recommended that all dams with a dam wall height of less than 11,50 metres may not be used any longer. Complete the code for the **Option G** button by formulating an SQL statement to delete the record of all dams from the **tblDams** table that have a dam wall height (**HeightOfWall)** of less than 11,50 metres.

Example of the output:

HINT: Run **Option A** to verify that the records have been deleted. (3)

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

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

The local municipality wants to make people more aware of how much water they are using on a daily basis. They want software for personal use by households to encourage them to use water sparingly.

A program, that has been partially developed, consists of an object class (unit), which describes the attributes and behaviours of a household regarding their water usage, and an application class (main unit). The program has to process data regarding the daily water consumption of the household over a period of one week and display information they can use to monitor their water usage.

Do the following:

- Rename the folder **Question2_Delphi** as **Question2_X** (where X must be replaced with your examination number).
- Open Delphi and then open the file **Question2_P.dpr** in the folder **Question2_X**.
- Go to 'File/Save As …' and save the main unit as **Question2_UXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the unit **uHousehold.pas**.
- Go to 'File/Save As …' and save the unit as **uHouseholdXXXX.pas** (where XXXX must be replaced with 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 with the last FOUR digits of your examination number).

You are required to correct and complete the given program by doing the following:

2.1 The unit named **uHouseholdXXXX.pas** contains attributes and methods that describe the water usage of a single household. Modify code in the given methods and write some additional methods, as described below.

2.1.1 The constructor receives the following information of a household as parameters:

- An account number
- The number of members in the household
- An array containing seven integer values representing the daily water usage of the household, measured in litres, over a period of one week

Initialise the account number field, the number of members field and the array, using the parameters received by the constructor. (3)

2.1.2 Write a method called **calculateTotal** to calculate and return the total amount of water used by the household during one week. Use the values assigned to the array to calculate the total. (4)

2.1.3 Write a method called **calculateAve** to calculate and return the average water usage of the household per day. Use the method **calculateTotal** in the calculation. (2)

2.1.4 Write a method called **determineHighDay** that will calculate and return the day of the week when the most water was used by the household. The value to be returned must be a number. (4)

2.1.5 The method called **determineHighRisk** will return a Boolean value indicating whether the household is a high-risk household or not, in terms of their water usage. The method receives a parameter indicating the acceptable limit of water usage for a household per day.

A household is a high-risk household in terms of water usage, if:

- The average water usage of the household per day is more than the daily limit

OR

- More than two of the daily water-usage figures by the household in one week exceed the daily limit

Complete the method to return the correct Boolean value based on the criteria explained above. (9)

2.1.6 You have been provided with a method called **toString** that constructs and returns a string containing the account number and the number of members in the household.

Add code to the method so that the string will include headings, labels and the contents of the array in the following format:

Account number: xxxxx
Number of members: x
Daily water usage

Days:	1	2	3	etc.
Water used:	xxx	xxx	xxx	etc.

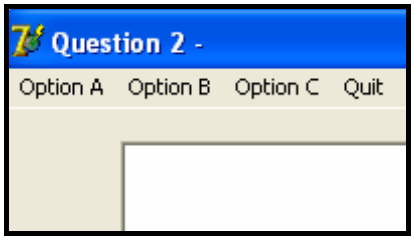
Example of the output when the string returned by the **toString** method is displayed:

Account number: AC-23245
Number of members: 4
Daily water usage

Days:	1	2	3	4	5	6	7
Water used:	481	438	454	353	421	396	432

(6)

2.2 In the **Question2_UXXXX.pas** file (the main unit) you have been provided with a menu component that will display the following options when you execute the program:



Do the following:

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

2.2.1 Use the account number, the number of members in the household and the array containing the water-usage values of the household for seven days, as given in the program, to create a type **THousehold** object. (2)

2.2.2 **Menu Option A**

When the user selects this menu option, the program must invoke the relevant methods to display the account number, number of members in the household, the water usage for each of the seven days of the week, the total water usage and the average water usage per day as shown below.

Example of the output:

```
Account number: AC-23245
Number of members: 4
Daily water usage
Days:          1    2    3    4    5    6    7
Water used:    481  438  454  353  421  396  432

Total water usage: 2975 litres
Average water usage per day: 425.0 litres
```

(4)

2.2.3 **Menu Option B**

When the user selects this menu option, the program must display a heading and invoke the relevant methods to display the average water usage of the household per day. The program must then display subheadings and the days on which the water usage exceeded the average water usage per day, and by how many litres it was exceeded.

Example of the output (on the next page):

```
Days and amount of water exceeding the average
==========================================
Average water usage per day: 425.0 litres
Days   Value exceeding average by (litres)
1          56.0
2          13.0
3          29.0
7          7.0
```

(6)

2.2.4 **Menu Option C**

When the user selects this menu option, the user will be asked to enter a value representing an acceptable limit of water usage for a household per day. The program must invoke the relevant methods to display the information, as shown in the sample output. Also display a suitable message indicating whether the household is a high-risk household or not.

Example of the output with an input value of 400:

```
Account number: AC-23245
Number of members: 4
Daily water usage
Days:          1     2     3     4     5     6     7
Water used:    481   438   454   353   421   396   432

The day on which the most water was used: 1

High-risk household
```

(5)

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

[45]

QUESTION 3: DELPHI – PROGRAMMING

The local Department of Water has been inundated with many calls and e-mails relating to residential and business water accounts. A call centre has been set up at the Department of Water to handle all the issues relating to these accounts.

The issues are categorised as an **account query**, a **complaint** or a **suggestion**.

The software that will be developed will separate the suggestions from the account queries and the complaints. A reference number is allocated to each account query and each complaint, using a specified format.

An account holder can make enquiries about the status of his/her account by providing the account number. The personnel at the call centre will then draw up a schedule that indicates all account queries and complaints related to the account number submitted.

You have been provided with an incomplete program in the folder named **Question3_Delphi**.

A text file called **Data.txt**, containing all the issues related to **account queries**, **complaints** and **suggestions**, has also been supplied in the same folder.

Each line of text in the file has the following format:

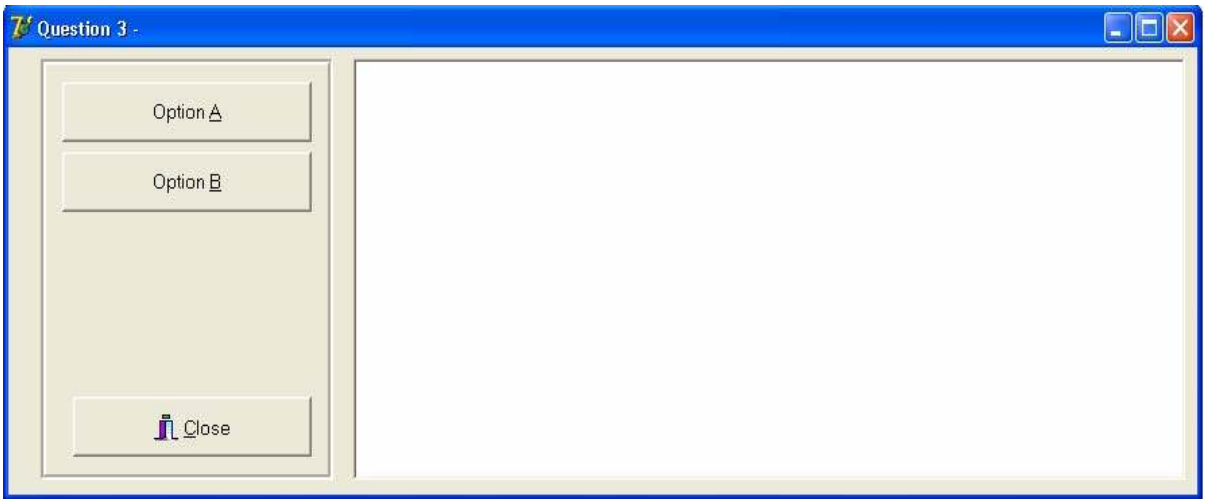
The type of issue:Account number:Date#Contents of the issue

Example of the contents of the text file:

```
Complaint:P8120876:03/03/2011#Unresolved water issues after five visits to municipality
Suggestion:F543199:10/04/2011#Water should be supplied every hour if normal water supply is suspended
Account:F999765:30/04/2011#Statement of account for April has not been supplied
Complaint:H654321:04/05/2011#Rate of billing is inconsistent from month to month
Account:T564321:14/06/2011#Paid an extra R150 for May 2011, balance has not been carried over for June
Suggestion:J345556:21/06/2011#E-mail statements instead of posting to account holders
Complaint:K567543:03/07/2011#Overflow of water for June 2011, awaiting payment from water insurance
Account:G654321:01/08/2011#Statement not received for July 2011
Complaint:B654321:10/08/2011#Water shortages occur too often in Peacevale
Complaint:K567543:15/08/2011#Still awaiting payment from water insurance
Account:Y6754332:29/08/2011#Overcharged for August 2011, statement does not match usage
Account:K567543:01/09/2011#A second payment for August 2011 is not reflected on September statement
Account:K765434:23/09/2011#August payment not reflected on account for September 2011
Complaint:K567543:01/10/2011#Incorrect reading for September 2011 - premises were locked the entire month
Suggestion:H876543:05/10/2011#Review new tariffs for 2012 before implementation
```


Do the following:

- Rename the folder named **Question3_Delphi** as **Question3_X** (where X should be replaced with your examination number).
- Open the Delphi program in this folder.
- Save the main unit as ('File/Save As') **Question3_UXXXX** and the project as ('File/Save Project As') **Question3_PXXXX** inside the folder (where XXXX should be replaced with the last FOUR digits of your examination number).
- Add your examination number to the right of 'Question 3 –' in the caption of the form.
- Execute the program. A menu with the following options will be displayed:



Do the following:

3.1 An empty text file must be created to store all the suggestions made by account holders. Write a procedure called **createSuggestionsFile** to create an empty text file and use **Suggestions** as the name of the text file. (2)

3.2 Only account queries, complaints and suggestions with valid account numbers will be processed. Write a subprogram called **validateAccNum** to receive an account number as a string parameter and return a Boolean value indicating whether the account number is valid or not. A valid account number must satisfy the following criteria:

- The account number must have only SEVEN characters.
- The account number must start with a letter. (6)

3.3 When the user clicks on **Option A**, the data from the **Data.txt** text file with valid account numbers only must be used to write the suggestions to the **Suggestions.txt** text file and create reference numbers for all the account queries and all the complaints.

Complete the code for **Option A** that will do the following:

For each line of text that is read from the **Data.txt** text file, do the following:

Validate the account numbers using the **validateAccNum** subprogram.

If the account number is invalid, the line of text must be ignored.
If the account number is valid, the following must be done:

- If it is a suggestion, write the account number, the date and the contents of the suggestion to the text file called **Suggestions.txt** in the following format:

  Account number**:**date#contents of suggestion

- If it is not a suggestion:
  - Create a **reference number** containing the following information in the format shown below:

    - A letter indicating the type of issue (A for 'Account query' or C for 'Complaint')
    - A number indicating the sequence of this specific account query or complaint

      Example: The first account query that is read from the **Data.txt** file will be 1,
      the second account query will be 2,
      the third account query will be 3, etc.

      The first complaint that is read from the **Data.txt** file will be 1,
      the second complaint will be 2,
      the third complaint will be 3, etc.

    - A hyphen
    - The account number of the issue
    - A hyphen
    - The date of the issue

  - Display the reference number.
  - Store the reference number as well as the content of the account query or complaint so that it can be used in Option B.

NOTE: The contents of each account query and complaint will be required when account holders request the status of their accounts in Option B.

Example of the output:

```
Reference Numbers
==================
A1-F999765-30/04/2011
C1-H654321-04/05/2011
A2-T564321-14/06/2011
C2-K567543-03/07/2011
A3-G654321-01/08/2011
C3-B654321-10/08/2011
C4-K567543-15/08/2011
A4-K567543-01/09/2011
A5-K765434-23/09/2011
C5-K567543-01/10/2011
```

(24)

3.4 An account holder can query the status of his/her account by submitting a valid account number. All account queries and complaints related to this account number must be displayed.

Complete the code for **Option B** as follows:

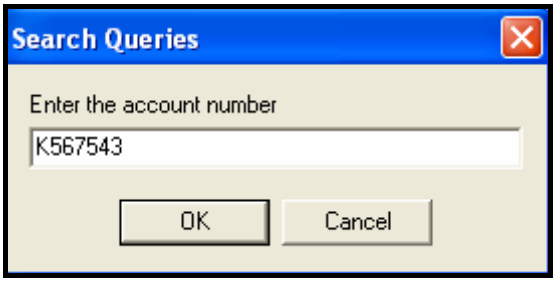
- Allow the user to enter an account number and call the **validateAccNum** subprogram to validate the account number.
- Display a suitable message if the account number is invalid and terminate the search process. Continue the search process if the account number is valid.
- All issues related to the account number entered must be displayed in the following format:

  Reference number of the issue <tab> Description of the issue

- Display a suitable message if there are no issues reported for the account number entered.

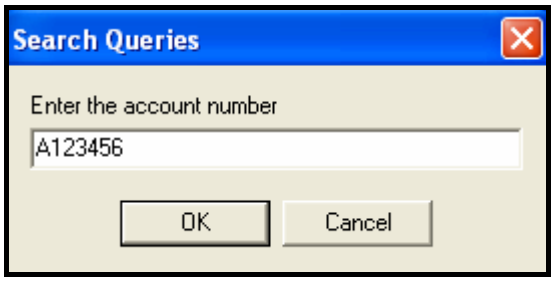
NOTE: You have to run **Option A** if **Option B** is to produce the correct output.

Example of the output using the account number **K567543** as input (on the next page):




```
C2-K567543-03/07/2011  Overflow of water for June 2011, awaiting payment from water insurance
C4-K567543-15/08/2011  Still awaiting payment from water insurance
A4-K567543-01/09/2011  A second payment for August 2011 is not reflected on September statement
C5-K567543-01/10/2011  Incorrect reading for September 2011 - premises were locked the entire month
```

Example of the output using the account number **A123456** as input:




```
No issues have been reported for account number: A123456
```

(8)

- Enter your examination number as a comment in the first line of the main unit **Question3_UXXXX**.
- Save the main unit and the project ('File/Save All').
- A printout of the code for the unit **Question3_UXXXX** will be required. **[40]**

TOTAL SECTION A: 120

SECTION B

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

SCENARIO

Water is one of the most essential commodities required for the survival of human beings, plants and animals. The Department of Water Affairs is embarking on an intensive campaign to help save water. Many measures and programmes have been put in place to help bring about awareness on how to use water sparingly.

QUESTION 1: JAVA PROGRAMMING AND DATABASE

The Department of Water Affairs has created a database named **DamsDB** containing information on all the dams in the country and the towns to which they supply water. An incomplete program has been developed to process queries on the data in the **DamsDB** database. Your task will be to complete this program.

The database named **DamsDB**, as well as an incomplete Java program, are saved in the folder named **Question1_Java**. The folder contains a test class named **TestQuestion1.java** and an object class named **Dams.java** which will display the results of the queries.

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

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

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

Do the following:

- Rename the folder **Question1_Java** as **Question1_X**, where X should be replaced with your examination number.
- Rename the **TestQuestion1.java** file in the folder **Question1_X** as **TestQuestion1XXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the incomplete program **TestQuestion1XXXX.java** in the **Question1_X** folder.
- Change the name of the class to **TestQuestion1XXXX** (where XXXX must be replaced with the last FOUR digits of your examination number). Save the file.

The connectivity code as well as the code to display the results have already been written as part of the given code in the file named **Dams.java**.

NOTE: If your program cannot connect to the database, make sure that the database file **DamsDB** is in the same folder as your program. If this is not the case, copy the database file **DamsDB** into 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 program.

NOTE: If you still cannot establish connectivity with the database 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 file named TestQuestion1XXXX.java.

When you compile and execute the **TestQuestion1XXXX.java** file, the menu below will be displayed. 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

Your choice?
```

Do the following:

Complete the SQL statements in the **TestQuestion1XXXX.java** file for each menu option, as indicated in QUESTIONS 1.1 to 1.7 below. The code to pass the SQL statements to the relevant method in the **Dams.java** file has been given to you. You only need to complete the SQL statements and some input statements, as required in the **TestQuestion1XXXX.java** file.

1.1 The Department of Water Affairs wants a list of all the dams in the country, sorted according to the height of the dam walls from the lowest to the highest. Complete the code for **Option A** by formulating an SQL statement to display **all the details** of the dams stored in the **tblDams** table, sorted as required.

Example of the output for the first seven records:

DamID	DamName	River	YearCompleted	DamLevel	Capacity	HeightOfWall
83	Lake Mzingazi Dam	Mzingazi River	1942	19678	37000	8.2
146	Vaal Barrage	Vaal River	1922	48897	56712	10.3
34	Douglas Weir	Vaal River	1977	5910	16700	10.6
152	Voëlvlei Dam	Voëlvlei River	1971	131302	158600	10.6
41	Emmarentia Dam	Braamfontein Spruit	1912	151	250	11.4
68	Klipdrif Dam	Loop Spruit	1918	4629	13300	12.2
148	Vaalharts Storage Weir	Vaal River	1936	24105	48700	12.5

: (3)

1.2 One of the main concerns is large urban towns. The Department wants a list of all towns in a particular province that have a population exceeding 100 000. Complete the code for **Option B** by asking the user to enter the name of the province. Formulate an SQL statement to display the **TownName** and **Population** of all the towns that have a population exceeding 100 000 in the designated province.

Example of the input and output of all the towns in **Gauteng** with a population exceeding 100 000:

```
Enter the name of the province: Gauteng

TownName          Population
==========================
Johannesburg         1975500
Tshwane              1473800
Vanderbijlpark        338000
```

(6)

1.3 An audit of the dams is taking place and additional information (not stored in the table) is required. You must write a query to display the age of each dam, as well as the current water level of each dam, as a percentage of its capacity. The age of a dam is calculated by subtracting **YearCompleted** from the current year. Call this field **Age**. The current water level of a dam as a percentage of its capacity can be calculated using the fields **DamLevel** and **Capacity**. Call this field **Percentage** and round it down to ONE decimal place. Complete the code for **Option C** by formulating an SQL statement to display the **DamID**, **DamName** and the two calculated fields.

Example of the output for the first seven records:

DamID	DamName	Age	Percentage
1	Albasini Dam	59	71.1
2	Albert Falls Dam	35	49.4
3	Allemanskraal Dam	51	73.7
4	Alphen Dam	21	33.9
5	Armenia Dam	57	54.3
6	Beervlei Dam	54	58.8
7	Berg River Dam	4	88.4
:			

(7)

1.4 The Department of Water Affairs considers any town with water restrictions to be a 'critical town' and wants to know how many critical towns there are in each province. Complete the code for **Option D** by formulating an SQL statement that will display the **Province** and a **calculated field** for **the total number of critical towns** in that province. Name the calculated field **CriticalTowns**.

Example of the output:

Province	CriticalTowns
Eastern Cape	11
Free State	6
Gauteng	3
KwaZulu-Natal	11
Limpopo	7
Mpumalanga	11
North West	5
Northern Cape	3
Western Cape	15

(5)

1.5 Due to the fact that the Vaal River flows through a number of provinces, the Department needs to know which provinces would be affected, should the Vaal River be contaminated by pollution. A province is supplied with water by the Vaal River if the dam that supplies a town in the province with water, receives water from the Vaal River. Complete the code for **Option E** by formulating an SQL statement to display the names of all the provinces that are supplied with water by the Vaal River. The name of each province should appear in the list only ONCE.

Example of the output:

Province
Free State
Gauteng
Mpumalanga
North West
Northern Cape

(7)

1.6 Some analysts have indicated that the **North West** province will experience severe droughts in the coming years. They have recommended that water restrictions be imposed on all towns in this province, which means they will all become critical towns. Complete the code for **Option F** by formulating an SQL statement that will **update** the records of all towns in the North West province to show which towns have water restrictions.

Example of the output:

```
Records Processed Successfully
```

HINT: Run **Option D** to verify that the records have been updated. There should be 13 critical towns in the North West province after Option F has been executed. (4)

1.7 The risk of flooding has been assessed and it is recommended that all dams with a dam wall height of less than 11,50 metres, may not be used any longer. Complete the code for **Option G** by formulating an SQL statement to delete the record of all dams from the **tblDams** table that have a dam wall height (**HeightOfWall**) of less than 11,50 metres.

Example of the output:

```
Records Processed Successfully
```

HINT: Run **Option A** to verify that the records have been deleted. (3)

- Enter your examination number as a comment in the first line of the file named **TestQuestion1XXXX.java** containing the SQL statements.
- Save the **TestQuestion1XXXX.java** file.
- A printout for the code of the **TestQuestion1XXXX.java** file will be required. **[35]**

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING

The local municipality wants to make people more aware of how much water they are using on a daily basis. They want software for personal use by households to encourage them to use water sparingly.

A program, that has been partially developed, consists of an object class, which describes the attributes and behaviours of a household regarding their water usage, and a test class. The program has to process data regarding the daily water consumption of the household over a period of one week and display information they can use to monitor their water usage.

Do the following:

- Rename the folder **Question2_Java** as **Question2_X** (where X must be replaced with your examination number).
- Rename the **Household.java** file in the folder **Question2_X** as **HouseholdXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Rename the **TestQuestion2.java** file in the folder **Question2_X** to **TestQuestion2XXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the **HouseholdXXXX.java** file.
- Change the **class name** and the **constructor methods** to **HouseholdXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Add your examination number as a comment in the first line of the **HouseholdXXXX.java** class. Save the file.
- Open the **TestQuestion2XXXX.java** file.
- Change the **class name** to **TestQuestion2XXXX** (where XXXX must be replaced with the last FOUR digits of your examination number). Save the file.

You are required to correct and complete the given program by doing the following:

2.1 The object class named **HouseholdXXXX.java** contains attributes and methods that describe the water usage of a single household. Modify code in the given methods and write some additional methods, as described on the next page.

2.1.1 The constructor receives the following information of a household as parameters:

- An account number
- The number of members in the household
- An array containing seven integer values representing the daily water usage of the household, measured in litres, over a period of one week

Initialise the account number field, the number of members field and the array, using the parameters received by the constructor. (3)

2.1.2 Write a method called **calculateTotal()** to calculate and return the total amount of water used by the household during one week. Use the values assigned to the array to calculate the total. (4)

2.1.3 Write a method called **calculateAve()** to calculate and return the average water usage of a household per day. Use the method **calculateTotal()** in the calculation. (2)

2.1.4 Write a method called **determineHighDay()** that will calculate and return the day of the week when the most water was used by the household. The value to be returned must be a number. (4)

2.1.5 The method called **determineHighRisk(…)** will return a Boolean value indicating whether the household is a high-risk household or not, in terms of their water usage. The method receives a parameter indicating the acceptable limit of water usage for a household per day.

A household is a high-risk household in terms of water usage, if:

- The average water usage of the household per day is more than the daily limit

OR

- More than two of the daily water-usage figures by the household in one week exceed the daily limit

Complete the method to return the correct Boolean value based on the criteria explained above. (9)

2.1.6 You have been provided with a method called **toString()** that constructs and returns a string containing the account number and the number of members in the household.

Add code to the method so that the string will include headings, labels and the contents of the array in the following format:

Account number: xxxxx
Number of members: x
Daily water usage

Days:	1	2	3	etc.
Water used:	xxx	xxx	xxx	etc.

Example of the output when the string returned by the **toString()** method is displayed:

```
Account number: AC-23245
Number of members: 4
Daily water usage
Days:          1      2      3      4      5      6      7
Water used:    481    438    454    353    421    396    432
```

(6)

2.2 In the **TestQuestion2XXXX.java** file (the test class) you have been provided with code to display the following menu when you execute the program:

```
  Menu

Option A
Option B
Option C

Q - QUIT

Your choice?
```

Do the following:

- Add your examination number as a comment in the first line of the **TestQuestion2XXXX.java** class.
- Write code in the given **TestQuestion2XXXX.java** class to do the following:

2.2.1 Use the account number, the number of members in the household and the array containing the water-usage values of the household for seven days, as given in the program, to create a **Household** object. (2)

2.2.2 **Menu Option A**

When the user selects this menu option, the program must invoke the relevant methods to display the account number, number of members in the household, the water usage for each of the seven days of the week, the total water usage and the average water usage per day, as shown on the next page.

Example of the output:

```
Account number: AC-23245
Number of members: 4
Daily water usage
Days:           1       2       3       4       5       6       7
Water used:     481     438     454     353     421     396     432

Total water usage: 2975 litres
Average water usage: 425.0 litres
```

(4)

2.2.3 Menu Option B

When the user selects this menu option, the program must display a heading and invoke the relevant methods to display the average water usage of the household per day. The program must then display subheadings and the days on which the water usage exceeded the average water usage per day and by how many litres it was exceeded.

Example of the output:

```
Days and amount of water exceeding the average
==============================================
Average water usage per day: 425.0 litres
Days    Value exceeding average by (litres)
1          56.0
2          13.0
3          29.0
7          7.0
```

(6)

2.2.4 Menu Option C

When the user selects this menu option, the user will be asked to enter a value representing an acceptable limit of water usage for a household per day. The program must invoke the relevant methods to display the information, as shown in the sample output. Also display a suitable message indicating whether the household is a high-risk household or not.

Example of the output with an input value of 400:

```
Account number: AC-23245
Number of members: 4
Daily water usage
Days:           1       2       3       4       5       6       7
Water used:     481     438     454     353     421     396     432

The day on which the most water was used: 1
High-risk household
```

(5)

- Make sure that your examination number is entered as a comment in the first line of the test class **Question2XXXX.java**, as well as the object class **HouseholdXXXX.java**.
- Save all the files (File/Save All).
- Printouts of the code for the classes **Question2XXXX.java** and **HouseholdXXXX.java** will be required. **[45]**

QUESTION 3: JAVA – PROGRAMMING

The local Department of Water has been inundated with many calls and e-mails relating to residential and business water accounts. A call centre has been set up at the Department of Water to handle all the issues relating to these accounts.

The issues are categorised as an **account query**, a **complaint** or a **suggestion**.

The software that will be developed will separate the suggestions from the account queries and the complaint. A reference number is allocated to each account query and each complaint, using a specified format.

An account holder can make enquiries about the status of his/her account by providing the account number. The personnel at the call centre will then draw up a schedule that indicates all account queries and complaints related to the account number submitted.

You have been provided with an incomplete program in the folder named **Question3_Java**.

A text file called **Data.txt**, containing all the issues related to **account queries**, **complaints** and **suggestions**, has also been supplied in the same folder.

Each line of text in the file has the following format:

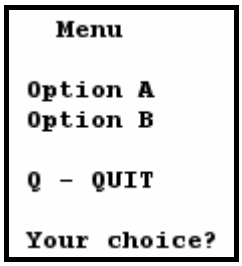
The type of issue:Account number:Date#Contents of the issue

Example of the contents of the text file:

```
Complaint:P8120876:03/03/2011#Unresolved water issues after five visits to municipality
Suggestion:F543199:10/04/2011#Water should be supplied every hour if normal water supply is suspended
Account:F999765:30/04/2011#Statement of account for April has not been supplied
Complaint:H654321:04/05/2011#Rate of billing is inconsistent from month to month
Account:T564321:14/06/2011#Paid an extra R150 for May 2011, balance has not been carried over for June
Suggestion:J345556:21/06/2011#E-mail statements instead of posting to account holders
Complaint:K567543:03/07/2011#Overflow of water for June 2011, awaiting payment from water insurance
Account:G654321:01/08/2011#Statement not received for July 2011
Complaint:B654321:10/08/2011#Water shortages occur too often in Peacevale
Complaint:K567543:15/08/2011#Still awaiting payment from water insurance
Account:Y6754332:29/08/2011#Overcharged for August 2011, statement does not match usage
Account:K567543:01/09/2011#A second payment for August 2011 is not reflected on September statement
Account:K765434:23/09/2011#August payment not reflected on account for September 2011
Complaint:K567543:01/10/2011#Incorrect reading for September 2011 - premises were locked the entire month
Suggestion:H876543:05/10/2011#Review new tariffs for 2012 before implementation
```

Do the following:

- Rename the folder named **Question3_Java** as **Question3_X** (where X should be replaced with your examination number).
- Rename the file **TestQuestion3.java** in this folder as **TestQuestion3XXXX.java** (where XXXX should be replaced with the last FOUR digits of your examination number).
- Open the file (incomplete program) **TestQuestion3XXXX.java**.
- Add your examination number as a comment in the first line of the program.
- Change the class name to **TestQuestion3XXXX** (where XXXX must be replaced with the last FOUR digits of your examination number). Save the file.
- Execute the program. A menu with the following options will be displayed:



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

Do the following:

3.1 An empty text file must be created to store all the suggestions made by account holders. Write a method called **createSuggestionsFile** to create an empty text file and use **Suggestions.txt** as the name of the text file. (2)

3.2 Only account queries, complaints and suggestions with valid account numbers will be processed. Write a method called **validateAccNum** to receive an account number as a string parameter and return a Boolean value indicating whether the account number is valid or not. A valid account number must satisfy the following criteria:

- The account number must have only SEVEN characters.
- The account number must start with a letter. (6)

3.3 When the user chooses **Option A**, the data from the **Data.txt** text file with valid account numbers only must be used to write the suggestions to the **Suggestions.txt** text file and create reference numbers for all the account queries and all the complaints.

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

For each line of text that is read from the **Data.txt** text file, do the following:

Validate the account numbers using the method **validateAccNum**.

If the account number is invalid, the line of text must be ignored.
If the account number is valid, the following must be done:

- If it is a suggestion, write the account number, the date and the contents of the suggestion to the text file called **Suggestions.txt** in the following format:

  Account number**:**date#contents of suggestion

- If it is not a suggestion:
  - Create a **reference number** containing the following information in the format shown below:

    - A letter indicating the type of issue (A for 'Account query' or C for 'Complaint')
    - A number indicating the sequence of this specific account query or complaint

      Example: The first account query that is read from the **Data.txt** file will be 1,
      the second account query will be 2,
      the third account query will be 3, etc.

      The first complaint that is read from the **Data.txt** file will be 1,
      the second complaint will be 2,
      the third complaint will be 3, etc.

    - A hyphen
    - The account number of the issue
    - A hyphen
    - The date of the issue

  - Display the reference number.
  - Store the reference number as well as the content of the account query or complaint so that it can be used in Option B.

NOTE: The contents of each account query and complaint will be required when account holders request the status of their accounts in Option B.

Example of the output:

```
Reference Numbers
=================
A1-F999765-30/04/2011
C1-H654321-04/05/2011
A2-T564321-14/06/2011
C2-K567543-03/07/2011
A3-G654321-01/08/2011
C3-B654321-10/08/2011
C4-K567543-15/08/2011
A4-K567543-01/09/2011
A5-K765434-23/09/2011
C5-K567543-01/10/2011
```

(24)

3.4 An account holder can query the status of his/her account by submitting a valid account number. All account queries and complaint related to this account number must be displayed.

Complete the code for **Option B** as follows:

- Allow the user to enter an account number and call the **validateAccNum** method to validate the account number.
- Display a suitable message if the account number is invalid and terminate the search process. Continue the search process if the account number is valid.
- All issues related to the account number entered must be displayed in the following format:

  Reference number of the issue <tab> Description of the issue

- Display a suitable message if there are no issues reported for the account number entered.

NOTE: You have to run **Option A** if **Option B** is to produce the correct output.

Example of the output using the account number **K567543** as input:

```
Enter the account number to query
K567543

C2-K567543-03/07/2011   Overflow of water for June 2011, awaiting payment from water insurance
C4-K567543-15/08/2011   Still awaiting payment from water insurance
A4-K567543-01/09/2011   A second payment for August 2011 is not reflected on September statement
C5-K567543-01/10/2011   Incorrect reading for September 2011 - premises were locked the entire month
```

Example of the output using the account number **A123456** as input:

```
Enter the account number to query
A123456

No issues have been reported for account number:A123456
```

(8)

- Enter your examination number as a comment in the first line of the test class **TestQuestion3XXXX.java**, as well as any other class(es) you have created with code.
- Save the class(es).
- A printout of the code for the test class **TestQuestion3XXXX.java**, as well as any other class(es) you have created, will be required.

[40]

TOTAL SECTION B: 120
GRAND TOTAL: 120

ADDENDUM A: Table Description Sheet

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

tblDams Table Structure

tblDams : Table

	Field Name	Data Type	Description
🔑	DamID	Number	Unique number assigned to the dam
	DamName	Text	The name of the dam
	River	Text	River that supplies the dam with water
	YearCompleted	Number	The year that the dam was completed
	DamLevel	Number	The current level of water in the dam in thousands of litres
	Capacity	Number	The total capacity of the dam in thousands of litres
	HeightOfWall	Number	The height of the dam wall in metres

tblDams Table Sample Data

tblDams : Table

DamID	DamName	River	YearCompleted	DamLevel	Capacity	HeightOfWall
1	Albasini Dam	Levubu River	1952	20053	28200	34.55
2	Albert Falls Dam	Umgeni River	1976	142339	288100	33.52
3	Allemanskraal Dam	Sand River	1960	128666	174500	38.20
4	Alphen Dam	Bonte River	1990	237	700	18.70
5	Armenia Dam	Leeu River	1954	7056	13000	22.60
6	Beervlei Dam	Groot River	1957	50460	85800	31.60
7	Berg River Dam	Berg River	2007	112394	127100	68.40
8	Bivane Dam	Bivane River	2000	47764	115200	72.30
9	Bloemhoek Dam	Jordaan Spruit	1976	8867	26400	28.80
10	Bloemhof Dam	Vaal River	1970	875662	1240200	33.70
11	BlydeRivierpoort Dam	Blyde River	1974	53441	54400	71.50
12	Boegoeberg Dam	Orange River	1929	12121	19800	12.50
13	Bon Accord Dam	Apies River	1925	3447	4400	18.90
14	Bongolo Dam	Komani River	1908	3402	7015	24.10
15	Boschmanskop No 1 Dam	Woes-Alleen River	1995	7253	14400	22.70
16	Boskop Dam	Mooi River	1959	16950	20840	33.30
17	Bospoort Dam	Hex River	1933	8866	15800	28.60
18	Bridle Drift Dam	Buffels River	1969	66835	101600	55.40
19	Bronkhorstspruit Dam	Bronkhorst Spruit	1950	47748	57400	35.10
20	Buffeljags Dam	Buffeljags River	1967	4001	4800	24.70

tblTowns

tblTowns : Table

	Field Name	Data Type	
🔑	TownID	Number	Unique
	TownName	Text	Name o
	Province	Text	The pro
	WaterRestrictions	Yes/No	Whethe
	Population	Number	The po
	DamID	Number	The ID

tblTowns Ta

tblTowns : Table

TownID	TownName	Pro
1	Willowmore	Easter
2	Queenstown	Easter
3	East London	Easter
4	Kareedouw	Easter
5	Tsomo	Easter
6	Stutterheim	Easter
7	Kirkwood	Easter
8	Indwe	Easter
9	Uitenhage	Easter
10	Hazelmere	Easter
11	Humansdorp	Easter
12	Cradock	Easter
13	Qamata	Easter
14	Umtata	Easter
15	Graaff Reinet	Easter
16	Cofimvaba	Easter
17	Whittlesea	Easter
18	Kroonstad	Free S
19	Van Stadensrus	Free S
20	Theunissen	Free S

ADDENDUM B: Instructions to create the database DamsDB.mdb

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

- Use the two text files named **tblDams** and **tblTowns** that have been supplied. Create your **DamsDB** that includes a table named **tblDams** and another table named **tblTowns** in the fold **Question1_Java**.
- Change the data types and the sizes of the fields in the two tables according to the specifications

The **tblDams** table stores data on the dams in the country. The fields in the **tblDams** table are defined

Field Name	Type	Size	Description
DamID	Number	Integer	Unique number assigned to the dam
DamName	Text	30	The name of the dam
River	Text	25	River that supplies the dam with water
YearCompleted	Number	Integer	The year the dam was completed
DamLevel	Number	Long Integer	The current level of water in the dam in th
Capacity	Number	Long Integer	The total capacity of the dam in thousand
HeightOfWall	Number	Double	The height of the dam wall in metres

See ADDENDUM A for an example of the data in the **tblDams** table.

The **tblTowns** table stores data on the towns supplied with water from the dams. The fields in the follows:

Field Name	Type	Size	Description
TownID	Number	Integer	Unique number assigned to the town
TownName	Text	25	Name of the town
Province	Text	25	The province the town is in
WaterRestrictions	Yes/No		Whether water restrictions have been imp
Population	Number	Long Integer	The population of the town
DamID	Number	Integer	The ID of the dam that supplies the town'

See ADDENDUM A for an example of the data in the **tblTowns** table.

ADDENDUM C: Instructions to connect to the database in Delphi

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

- Click on the ADOQuery component named **qryRec**.
- Click on the Ellipsis 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 Ellipsis button (three dots) that allows you to browse and look for the **DamsDB** file. You will find this file in the **Question1_Delphi** folder. Once you have found it, select the **DamsDB** 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.

NSC

INFORMATION TECHNOLOGY P1

NOVEMBER 2011

INFORMATION SHEET *(to be completed by the candidate)*

120

NAME OF PROVINCE _____

CENTRE NUMBER _____

EXAMINATION NUMBER _____

WORKSTATION NUMBER _____

DATE OF EXAMINATION _____

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

Delphi	Java

Question number	Saved (*tick ✓*)	Maximum mark	Mark achieved	Marker initial/ code
1		35		
2		45		
3		40		
	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 2011

MEMORANDUM

MARKS: 120

This memorandum consists of 30 pages.

GENERAL INFORMATION:

- **Pages 2–11 contain the Delphi memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 12–22 contain the Java memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 23–30 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
  TfrmRec = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnA: TButton;
    btnB: TButton;
    btnC: TButton;
    btnD: TButton;
    btnE: TButton;
    btnF: TButton;
    btnG: TButton;
    BitBtn1: TBitBtn;
    qryRec: TADOQuery;
    tblRecAg: TDataSource;
    grdRec: TDBGrid;

    procedure btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);
    procedure btnCClick(Sender: TObject);
    procedure btnDClick(Sender: TObject);
    procedure btnEClick(Sender: TObject);
    procedure btnFClick(Sender: TObject);
    procedure btnGClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmRec: TfrmRec;
implementation

{$R *.dfm}
```

See ADDENDUM A for alternatives and marking guidelines

```
procedure TfrmRec.btnAClick(Sender: TObject);
begin
  qryRec.Active := False;                                    //QUESTION 1.1
  qryRec.SQL.Text := 'SELECT *✓ FROM tblDams✓ ORDER BY HeightOfWall✓ ASC';
  qryRec.Active := True;
end;                                                                  (3)
//=============================================================================
procedure TfrmRec.btnBClick(Sender: TObject);                //QUESTION 1.2
var
  pr : String;
begin
  qryRec.Active := False;
  pr := InputBox('Large Towns', 'Enter the name of the province', ''); ✓
  qryRec.SQL.Text := 'SELECT TownName, Population✓ FROM tblTowns WHERE✓
       Population > 100000✓ AND✓ Province = "' + pr + '"'✓;
    qryRec.Active := True;
end;                                                                  (6)
//=============================================================================
procedure TfrmRec.btnCClick(Sender: TObject);                //QUESTION 1.3
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT DamID, DamName✓, (YEAR(NOW())✓ - YearCompleted✓)
       AS Age✓, ROUND (DamLevel / Capacity * 100✓, 1✓) AS Percentage✓ FROM
       tblDams';
  qryRec.Active := True;
end;                                                                  (7)
//=============================================================================
procedure TfrmRec.btnDClick(Sender: TObject);                //QUESTION 1.4
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM
       tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY Province✓';
  qryRec.Active := True;
end;                                                                  (5)
//=============================================================================
procedure TfrmRec.btnEClick(Sender: TObject);                //QUESTION 1.5
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓ WHERE
     tblTowns.DamID✓ = tblDams.DamID✓ AND River✓ = "Vaal River"✓';
  qryRec.Active := True;
end;                                                                  (7)
//=============================================================================
procedure TfrmRec.btnFClick(Sender: TObject);                //QUESTION 1.6
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'UPDATE tblTowns✓ SET✓ WaterRestrictions = True✓ WHERE
      Province = "North West"✓';
  qryRec.ExecSQL;
  MessageDlg('Records Processed Successfully',mtInformation,[mbok],0);
end;                                                                  (4)
//=============================================================================
procedure TfrmRec.btnGClick(Sender: TObject);                //QUESTION 1.7
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓';
  MessageDlg('Records Processed Successfully',mtInformation,[mbok],0);
  qryRec.ExecSQL;
end;                                                                  (3)
//=============================================================================
end.                                                                 [35]
```

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

unit uHouseholdXXXX;

```
interface
uses SysUtils;
   type
       arrType = array[1..7] of integer;
       THousehold = class (TObject)
         private
            fAccount        :string;
            fMembers        :integer;
            fArrWaterUse    :arrType;
         public
          constructor create(aAccount : string; aMembers :integer;arrWaterUse :
                                                              arrType );
          function calculateTotal:integer;
          function calculateAve:double;
          function determineHighDay:integer;
          function determineHighRisk(dayLimit:real):boolean;
          function toString:string;
       end;

implementation
//=========================================================================
```

// Q 2.1.1 (3)

```
constructor THousehold.create(aAccount : string; aMembers:integer;
                                        arrWaterUse:arrType);
begin
  fAccount  := aAccount; ✓
  fMembers  := aMembers; ✓
  fArrWaterUse := arrWaterUse; ✓
end;
```

Q 2.1.1
(3) Assign parameters to private fields

```
Accept a loop to assign the array
Subtract only 1 mark if the assignment statements are reversed, e.g.
aAccount := fAccount
```

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

// Q 2.1.2 (4)

```
Ignore any errors in definition (declaration) of method - no marks
Total (or return value) can be double or int
```

```
function THousehold.calculateTotal:integer;
var
  iTotal, k :integer;

begin
  iTotal := 0; ✓
  for k := 1 to length(fArrWaterUse) do ✓
     iTotal := iTotal + fArrWaterUse[k];
      // or inc(iTotal, fArrWaterUse[k]);
  result := iTotal; ✓
end;
```

Q 2.1.2
(1) Initialise total
(1) for loop
(1) Add array element to total
(1) return total (use result or function name)

Accept: iTotal as an instance/global variable.
Accept: loop to <=7 or < 8
Accept: adding individual elements – no loop
Accept: not using a variable iTotal – add up and assign to result- all in one statement

Award 4 marks if method/code done correctly but in the main unit

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

// Q 2.1.3 (2)

```
function THousehold.calculateAve:double; ✓
begin
        result := calculateTotal / 7; ✓
end;
```

Q 2.1.3
(1) Data type of return value real (or double)
(1) Correct calculation

Accept the use of iTotal only if calculateTotal has been called (can be called in main unit.
Accept if values are added here to get a total.
Accept integer as a return type.

Award 2 marks if method/code done correctly but in the main unit

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

// Q 2.1.4 (8/2 = 4) (rounded up)

```
function THousehold.determineHighDay:integer; ✓
var
  iHighDay,  iHighAmount, k  :integer;
begin
  iHighDay := 1; ✓
  iHighAmount := fArrWaterUse[1]; ✓
  for k := 2 to 7 do✓
  begin
    if (fArrWaterUse [k] > iHighAmount) ✓ then
     begin
       iHighDay := k; ✓
       iHighAmount := fArrWaterUse[k]; ✓
     end;
     result := iHighDay; ✓
  end;
 end;
```

Q 2.1.4
(1) Return type integer
(1) Initialise iHighDay
(1) Initialise iHighAmount
(1) For loop
(1) if statement
(1) change iHighDay
(1) change iHighAmount
(1) return iHighDay

Accept sorting the amounts, also returned the correct day (full marks)
Accept correct variations of finding highest e.g. start with 0 as highest instead of first element.
Sorting done correctly but correct day not found and returned – 3 out of 4 marks

Award 4 marks if method done correctly but in the main unit

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

// Q 2.1.5 (9)

```
function  THousehold.determineHighRisk(dayLimit:real):boolean;
var
  rAve          :real;
  iCount, k     :integer;
begin
       rAve := calculateAve;
       iCount := 0; ✓
       for k := 1 to length(fArrWaterUse) do✓
       begin
         if(fArrWaterUse[k] > dayLimit) then✓
                  inc(iCount); ✓
       end;
       if ((rAve > dayLimit) ✓  OR✓ (iCount > 2)) ✓ then
        result := true✓
```

Q 2.1.5
(1) Initialise iCount
(1) Loop
(1) if array element > dayLimit
(1) increment count
(3) if rAve > dayLimit or iCount > 2
(1) return true
(1) else return false

```
       else
        result := false; ✓
end;
```

```
Accept variables as global
Do not deduct a mark for input of dayLimit
Accept: if ((calculateAve > dayLimit) OR (iCount > 2))
Accept: a single statement that returns a Boolean value
Result = ✓ (rAve > dayLimit)✓ OR ✓(iCount > 2) ✓✓
Accept: Initialising a Boolean variable, return the Boolean variable
```

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

// Q 2.1.6 (6)

1 mark for each piece of information = 5 marks
1 mark for adding all the information in one string

```
function THousehold.toString:string;
var
   sObjStr:  string;
   k:integer;
begin
   sObjStr := 'Account number : ' + fAccount + #13 + 'Number of members : ' +
            IntToStr(fMembers) + #13;
   sObjStr := sObjStr + 'Daily water usage' + #13 ✓+ 'Days:          ' + #9;
     for k := 1 to 7 do
              sObjStr := sObjStr + intToStr(k) ✓ + #9;

     sObjStr := sObjStr + #13 + 'Water used:'✓ + #9;
     for k := 1 to length(fArrWaterUse) do✓
       sObjStr := sObjStr + IntToStr(fArrWaterUse[k])✓+
                                                    #9;
                              // Join strings✓
    result :=  sObjStr;
end;
```

Q 2.1.6
(1) Headings + new line (#13 or #10)
(1) Day numbers
(1) Heading
(2) Values from array
(1) Strings concatenated

```
Accept separate array entries instead of the loop.
Accept any correct form of joining all correct information
```

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

unit Question2XXXX_U;

```
interface

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

type
  TfrmHousehold = class(TForm)
    MainMenu1: TMainMenu;
    OptionA: TMenuItem;
    OptionB: TMenuItem;
    redOutput: TRichEdit;
    OptionC: TMenuItem;
    Quit: TMenuItem;
    procedure FormActivate(Sender: TObject);
    procedure QuitClick(Sender: TObject);
    procedure OptionAClick(Sender: TObject);
    procedure OptionBClick(Sender: TObject);
    procedure OptionCClick(Sender: TObject);

  private
```

```
  public
    { Public declarations }
  end;

var
  frmHousehold: TfrmHousehold;

implementation
uses
  uHouseholdXXXX;
/=============================================================================
```

// Q 2.2.1 (2)

```
var
  Household   :THousehold; ✓
  sAccount    :string;
  iMembers    :integer;
  arrWaterUse :arrType = (481, 438, 454, 353, 421, 396, 432);
{$R *.dfm}

procedure TfrmHousehold.FormActivate(Sender: TObject);
begin
     sAccount := 'AC-23245';
     iMembers := 4;
     Household := THousehold.create(sAccount, iMembers, arrWaterUse); ✓
end;
```

Q 2.2.1
(2) Declare object variable

Deduct 1 mark for no parameters.

```
procedure TfrmHousehold.QuitClick(Sender: TObject);
begin
  Application.Terminate;
end;
//=============================================================================
```

// Q 2.2.2 (4)

```
procedure TfrmHousehold.OptionAClick(Sender: TObject);
begin
   redOutput.Clear;
   redOutput.Lines.Add(Household.toString); ✓
   redOutput.Lines.Add('');
   redOutput.Lines.Add('Total water usage: '✓ +
           IntToStr(Household.calculateTotal) ✓+' litres');
   redOutput.Lines.Add('Average water usage per day: ' +
         FloatToStrF(Household.calculateAve, ✓ ffFixed,8,1) + ' litres');
end;
```

Q 2.2.2
(1) Call the toString method of the object
(1) Display label
(1) Call calculateTotal method
(1) Call calculateAverage method

Do not be strict in the wording of the labels and formatting of values

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

// Q 2.2.3 (6)

```
procedure TfrmQuestion2.mnuOptionBClick(Sender: TObject);
var
  rAve :real;
  k :integer;
begin
     redOutput.Clear;
     rAve := Household.calculateAve; ✓
     redOutput.Lines.Add('Days and amount of water exceeding the average');
     redOutput.Lines.Add('=======================================');
```

Q 2.2.3
(1) Call calculateAve method
(1) Display average
(1) Loop
(1) if
(2) Display number & difference

```
    redOutput.Lines.Add('Average water usage per day: ' +
        FloatToStrF(Household.calculateAve, ffFixed, 8, 1) ✓ + ' litres');
    redOutput.Lines.Add('Days    Value exceeding average by (litres)');
    for k := 1 to length(arrWaterUse) do✓
      begin
        if (arrWaterUse[k] > rAve) then✓
          begin
            redOutput.Lines.Add(IntToStr(k) ✓ + #9 +
            FloatToStrF(arrWaterUse[k]- rAve, ✓ ffFixed,8,1));
          end;
      end;
end;
```

No marks for headings
Display average – no matter how average is obtained, mark is not for formatting
Fourth mark goes for calculation, not formatting

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

// Q 2.2.4 **(5)**

Q 2.2.4
(1) Input rDayLimit
(1) Call toString
(1) Call calculateHighDay
(1) If statement
(1) Display correct message

```
procedure TfrmQuestion2.mnuQuitClick(Sender: TObject);
var
  rDayLimit :double;
begin
  redOutput.Clear;
  rDayLimit := StrToFloat(InputBox('Water Limit',
                       'Enter the limit of water per day', '')); ✓
  redOutput.Lines.Add(Household.toString); ✓
  redOutput.Lines.Add('');
  redOutput.Lines.Add('The day on which the most water was used is: ' +
                   intToStr(household.determineHighDay)); ✓
  redOutput.Lines.Add('');
  if (Household.determineHighRisk(rDayLimit)) ✓ then
      redOutput.Lines.Add('High-risk household')
  else
      redOutput.Lines.Add('Not a high-risk household');   ✓
end;
end.
```

rDayLimit – integer or real
Second mark: For call of toString – no other way accepted to display
Third mark goes for calling method, not label. Accept with no label
Fourth mark: for calling the method as part of an if or assign to variable
Fifth mark: displaying message – mark for two messages with else or second if

[45]

QUESTION 3: DELPHI PROGRAMMING

NOTE: This is only a sample – learners may answer this question in any way they see fit. Make use of the generalised 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 btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion3: TfrmQuestion3;
  iCountRefs  : integer;
  arrRefs, arrQueries : array[1..100] of String;
implementation
{$R *.dfm}
//============================================================================
```

//QUESTION 3.1

```
procedure CreateSuggestionsFile;
var
  TFile : textfile;
begin
  AssignFile(TFile, 'Suggestions.txt');
  ReWrite(TFile);
  CloseFile(TFile);
end;
```

Q 3.1
Code was given in Afrikaans Java version.
2 marks allocated in Question 3.3

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

//QUESTION 3.2 (6)

Q 3.2
(1) Subprogram heading
(1) Initialise Boolean value
(2) if statement
(1) Change Boolean value
(1) Return Boolean value

```
function  validateAccNum(sAccNum:String): boolean; ✓
var
  bValid : boolean;
begin
      bValid := false; ✓
      if (length(sAccNum) = 7) ✓ and (sAccNum[1]in ['A'..'Z']) ✓ then
        begin
            bValid := true; ✓
        end;
      result := bValid; ✓
end;
```

```
Accept: if … else instead of initializing Boolean
Accept: Any correct code to obtain the first character
Accept: One statement in method returning Boolean, e.g.
Result := (length(sAccNum)and …….);
```

```
//============================================================================
//QUESTION 3.3                    (24) + 2
procedure TfrmQuestion3.btnAClick(Sender: TObject);
var
  inFile, sugFile : textfile;
  sLine, sAccNum, sQuery, sDate, sQueryType, sRefNum : String;
  iLoop, iComCount, iAccCount : integer;
begin
  CreateSuggestionsFile; ✓
  AssignFile(inFile, 'Data.txt'); ✓
  Reset(inFile); ✓
  AssignFile(sugFile, 'Suggestions.txt');
  Append(sugFile); ✓

  iCountRefs := 0;
  iComCount := 0;
  iAccCount := 0;
  while NOT EOF (inFile) do✓
    begin
      ReadLn(inFile, sLine); ✓

      sQueryType := Copy(sLine, 1, Pos(':', sLine) - 1); ✓
      Delete(sLine, 1, Pos(':', sLine));

      sAccNum := Copy(sLine, 1, Pos(':', sLine) - 1); ✓
      Delete(sLine, 1, Pos(':', sLine));

      sDate := Copy(sLine, 1, Pos('#', sLine) - 1); ✓
      Delete(sLine, 1, Pos('#', sLine));

      sQuery := sLine; ✓
      if (validateAccNum(sAccNum)) then✓
        begin
          if (sQueryType = 'Suggestion') then✓
             WriteLn(sugFile, sAccNum + ':' + sDate + '#' + sQuery); ✓
          else
            begin
              Inc(iCountRefs); ✓
              sRefNum := sQueryType[1]; ✓
              if (sQueryType = 'Complaint') then✓
                begin
                  Inc(iComCount); ✓
                  sRefNum := sRefNum + IntToStr(iComCount); ✓
                end
              else if (sQueryType = 'Account') then
                begin
                  Inc(iAccCount); ✓
                  sRefNum := sRefNum + IntToStr(iAccCount); ✓
                end;

              sRefNum := sRefNum + '-' + sAccNum + '-' + sDate; ✓
              arrRefs[iCountRefs] := sRefNum; ✓
              arrQueries[iCountRefs] := sQuery; ✓
            end;
        end;
       btnB.Enabled := true;
    end;

  redOutput.Lines.Clear;
  redOutput.Lines.Add('Reference Numbers');
  redOutput.Lines.Add('================');
```

Q 3.3
(1) Call createSuggestions file
(1) Open file for writing
(2) Open file Data.txt
(1) While not eof
(1) Read a line
(1) Extract type of issue
(1) Extract account Num
(1) Extract date
(1) Extract issue
(1) Call validateAccNo
(1) Check if suggestion
(1) Write suggestion to file
(1) Inside else Increase ref number counter
(1) Extract first letter of issue
(1) Check category
(2) Create ref number for complaint
(2) Create ref number for Account query
(1) Create issue reference number
(1) Store reference number in array
(1) Store query in array
(2) Display ref numbers
(1) Close Suggestions file

```
  for iLoop := 1 to iCountRefs do✓
    begin
      redOutput.Lines.Add(arrRefs[iLoop]); ✓
    end;
  CloseFile(sugFile); ✓
  CloseFile(inFile);
end;
```

Accept: Open and close Suggestion file inside loop.
While reading from file with begin and end = 1 mark, no marks with no begin and end
Accept any part of the text written to the Suggestions file.
Accept the whole word for checking purposes.

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

//QUESTION 3.4 (8)

```
procedure TfrmQuestion3.btnBClick(Sender: TObject);
var
  sAccNum : String;
  iLoop   : integer;
  bFound  : boolean;
begin
  sAccNum := Uppercase(InputBox('Search Queries',
             'Enter the account number', ''));
  redOutput.Lines.Clear;
  bFound := false; ✓
  if NOT(validateAccNum(sAccNum)) then✓
      ShowMessage('Invalid account number') ✓
  else
    begin
      for iLoop := 1 to iCountRefs do✓
        begin
          if (Pos(sAccNum, arrRefs[iLoop]) > 0) ✓  then
            begin
              redOutput.Lines.Add(arrRefs[iLoop] + #9 +
                                  arrQueries[iLoop]);✓
              bFound := true; ✓
            end;
        end;
      if bFound = false then
        begin
         redOutput.Lines.Add('No issues have been reported for
                  account number: ' + sAccNum);                  } ✓
        end;
     end; // else
 end;
```

Q 3.4
(1) Initialise Boolean variable
(1) Validate acc number
(1) Display message if invalid acc num is entered
(1) Loop
(1) Check if num entered in array
(1) Display ref num and query
(1) Change Boolean value
(1) Display message if input value not found

```
Do not subtract mark if no uppercase
Accept: Extract the account number and then compare
```

```
end.
//============================================================================
```

[40]

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 TestDams
   {
       public static void main (String[] args) throws SQLException,IOException
      {
   BufferedReader inKb = new BufferedReader (new InputStreamReader
(System.in));
      Dams DB = new Dams();
      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 tblDams✓ ORDER BY HeightOfWall✓ ASC";
                    DB.query(sql);
                    break;
                 }                                                    (3)
//=============================================================================
               case 'B':                                    //QUESTION 1.2
                  {
                    System.out.print("Enter the name of the province : ");
                    String pr = inKb.readLine();✓
                    sql = "SELECT TownName, Population✓ FROM tblTowns WHERE✓
                        Population > 100000✓ AND✓ Province = '" + pr + "'"✓;
                    DB.query(sql);
                    break;
                  }                                                   (6)
//=============================================================================
               case 'C':                                    //QUESTION 1.3
                  {
                    sql = "SELECT DamID, DamName✓, (YEAR(NOW())✓ -
                    YearCompleted✓) AS Age✓, ROUND (DamLevel / Capacity *
                    100✓, 1✓) AS Percentage✓ FROM tblDams ";
```

See ADDENDUM D for alternatives and marking guidelines

```
                    DB.query(sql);
                    break;
                  }                                                    (7)
//=============================================================================
               case 'D':                                          //QUESTION 1.4
                  {
                    sql = "SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM
                    tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY
                                    Province✓";
                    DB.query(sql);
                    break;
                  }                                                    (5)
//=============================================================================
               case 'E':                                          //QUESTION 1.5
                  {
                    sql = "SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓
                     WHERE tblTowns.DamID✓ = tblDams.DamID✓ AND River✓
                         = 'Vaal River'✓";
                    DB.query(sql);
                    break;
                  }                                                    (7)
//=============================================================================
               case 'F':   //QUESTION 1.6
                  {
                    sql = "UPDATE tblTowns✓ SET✓ WaterRestrictions = True✓
                           WHERE Province = 'North West'✓";
                    DB.query(sql);
                    break;
                  }                                                    (4)
//=============================================================================
               case 'G':                                          //QUESTION 1.7
                  {
                    sql = " DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓";
                    DB.query(sql);
                    break;
                  }                                                    (3)
//=============================================================================
            }
         }while (choice != 'Q');
         DB.disconnect();
         System.out.println("Done");
      }
   }
//=============================================================================
```

[35]

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

HouseholdXXXX.java

```
public class HouseholdXXXX
{
    private String account;
    private int members;
    private int [] arrWaterUse;

    public HouseholdXXXX()
    {

    }
```

// Q 2.1.1 (3)

```
    public HouseholdXXXX(String Account, int Members, int [] arrWater)
    {
        account = Account; ✓
        members = Members; ✓
        arrWaterUse = arrWater; ✓
    }
```

Q 2.1.1
(3) Assign parameter values to private fields

```
Accept a loop to assign the arrays
Subtract only 1 mark if the assignment statements are reversed, e.g.
arrWater := arrWaterUse
```

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

// Q 2.1.2 (4)

```
Ignore any errors in definition (declaration) of method - no marks
Return type can be double or int
```

```
public int calculateTotal()
    {
        int total = 0; ✓
        for (int k = 0 ; k < arrWaterUse.length; k++)✓
        {
            total = total + arrWaterUse[k]; ✓
           // or total += arrWaterUse[k];
        }
        return total; ✓
    }
```

Q 2.1.2
(1) Initialise total
(1) for loop
(1) Add array element to total
(1) return total

```
Accept: total as an instance / global variable.
Accept: loop to <=6 or < 7
Accept: adding individual elements – no loop
Accept: not using a variable total – add up and return in one statement
```

```
Award 4 marks if method/code done correctly but in the test/driver class
```

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

// Q 2.1.3 (2)

```
    public double✓ calculateAve()
    {
        return calculateTotal()/ 7.0; ✓

    }
```

Q 2.1.3
(1) Data type of return value double/int
(1) Correct calculation

Accept the use of total only if calculateTotal() has been called (can be called in test / driver class).
Accept if values are added here to get a total.
Accept int as a return type - accept / 7 instead of /7.0

Award 2 marks if method/code done correctly but in the test/driver class

//==========

// Q 2.1.4 (8/4=4) (rounded up)

```
public int determineHighDay()✓
  {
     int highDay = 1; ✓
     int highAmount = arrWaterUse[0]; ✓
     for (int k = 1;k < 7; k++)✓
     {
        if (arrWaterUse[k] > highAmount) ✓
        {
           highDay = k+1; ✓
           highAmount = arrWaterUse[k]; ✓
        }
     }
     return highDay; ✓
  }
```

Q 2.1.4
(1) Return type int
(1) Initialise highDay
(1) Initialise highAmount
(1) for loop
(1) if statement
(1) Change highDay
(1) Change highAmount
(1) return highDay

Accept sorting the amounts, also returned the correct day(full marks)
Accept correct variations of finding highest e.g. start with 0 as highest instead of first element.
Sorting done correctly but correct day not found and returned – 3 out of 4 mark)

Award 4 marks if method done correctly but in the test/driver class

//==========

// Q 2.1.5 (9)

```
public boolean determineHighRisk(double dayLimit)
 {
    double ave = calculateAve();

    int count = 0; ✓
    for (int k = 0; k < arrWaterUse.length;k++)✓
    {
       if(arrWaterUse[k] > dayLimit) ✓
       {
          count++;✓
       }
    }
    if (ave > dayLimit✓ || ✓count > 2) ✓

       return true; ✓
    else
       return false; ✓

 }
```

Q 2.1.5
(1) Initialise count
(1) Loop
(1) if array element > dayLimit
(1) Increment count
(3) if ave > dayLimit or counter > 2
(1) return true
(1) else return false

```
Accept variables as global/instance
Do not deduct a mark for input of dayLimit
Accept: if (calculateAve() > dayLimit || count > 2)
Accept: a single statement that returns a Boolean value
return ✓ (ave > dayLimit✓ || ✓count > 2) ✓✓
Accept: Initialising a Boolean variable, return the Boolean variable
```

//==========

// Q 2.1.6 (6)

1 mark for each piece of information = 5 marks
1 mark for adding all the information in one string

```
public String toString()
    {
        String objStr = "Account number: " + account + "\n";
        objStr = objStr + "Number of members: " + members + "\n";
        objStr = objStr + "Daily water usage" +"\n" ✓+ "Days:     " + "\t";
        for (int k = 1 ; k <= 7;k++)
        {
           objStr = objStr + k✓ + "\t";
        }
        objStr = objStr + "\n" + "Water used:"✓+ "\t";
        for (int k =0  ; k < arrWaterUse.length;k++)✓
        {
           objStr = objStr + (arrWaterUse[k] ✓ + "\t");
         }
                              // Add strings✓
        return objStr;
    }
```

Q 2.1.6
(1) Headings + new line
(1) Day numbers
(1) Heading
(2) Values from array
(1) Strings concatenated

Accept correct use of formatter to construct the string (Java)
Accept separate array entries instead of the loop.
Accept any correct form of joining all correct information

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

TestQuestion2XXXX

```
import java.util.Scanner;

public class TestQuestion2XXXX
{
//=============================================================================
```

// Q 2.2.1 (2)

```
public static void main(String args[]) throws Exception
{
  String accountNumber = "AC-23245";
  int members = 4;
  int [] arrWaterUse =  {481, 438, 454, 353, 421, 396, 432};

  HouseholdXXXX household ✓= new HouseholdXXXX (accountNumber, members,
         arrWaterUse); ✓
```

Q 2.2.1
(2) Declare object variable

Deduct 1 mark for no parameters.

```
//=============================================================================
   Scanner input = new Scanner(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("       Option C ");
      System.out.println(" ");
      System.out.println("       Q - QUIT");
```

```
      System.out.println(" ");
      System.out.print("       Your choice? ");
      ch = input.nextLine().toUpperCase().charAt(0);

      switch (ch)
      {
//============================================================================
   // Q 2.2.2                    (4)

      case 'A':
      {
        System.out.println();
        System.out.println(household.toString());✓
        System.out.println(" ");
        System.out.println("Total water usage: " ✓+
                           household.calculateTotal()✓ + " litres");
        System.out.printf("%s%6.1f%s","Average water usage:",
                           household.calculateAve()✓, " litres\n");
        break;
      }
```

Q 2.2.2
(1) Call the toString method of the object
(1) Display label
(1) Call calculateTotal method
(1) Call calculateAverage method

```
Accept: Call to the toString method as: System.out.println(household)
Do not be strict on the wording of labels or formatting of values
```

```
//============================================================================
// Q 2.2.3                    (6)
    case 'B':
    {
      System.out.println();
      double ave = household.calculateAve();✓
      System.out.println("Days and amount of water exceeding the average ");
      System.out.println("=========================================
      System.out.printf("%s%.1f%s","Average water usage per
            day:",household.calculateAve()✓," litres\n");
      System.out.println("Days    Value exceeding average by (litres)");

      for (int k = 0 ; k < arrWaterUse.length;k++)✓
       {
        if (arrWaterUse[k] > ave) ✓
        {
            System.out.printf("%d%s%.1f%s",(k+1)✓,
             "\t\t" , (arrWaterUse[k]- ave, ✓ "\n"));
        }
     }
     System.out.println(" ");
     break;
    }
```

Q 2.2.3
(1) Call calculateAve() method
(1) Display average
(1) Loop
(1) if
(2) Display number & difference

```
No marks for headings
Display average – no matter how average is obtained, mark not for
formatting
Fourth mark goes for calculation, not formatting
```

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

// Q 2.2.4 **(5)**

```
    case 'C':
     {
       System.out.println("Enter the limit of water per day");
       double dayLimit = input.nextDouble();✓
       System.out.println(household.toString());✓
       System.out.println(" ");
       System.out.println("The day on which the most water was
       used: " + household.determineHighDay());✓

       if (household.determineHighRisk(dayLimit)) ✓
       System.out.println("High-risk household");
       else
       System.out.println("Not a high-risk household"); ✓

        break;
        }
```

Q 2.2.4
(1) Input dayLimit
(1) Call toString
(1) Call calculateHighDay()
(1) If statement
(1) Display correct message

```
dayLimit – integer or real
Second mark: For call of toString – no other way accepted to display
Third mark goes for calling method, not label. Accept with no label
Fourth mark: for calling the method as part of an if or assign statement
Fifth mark: displaying message – mark for two messages with else or second
if
```

```
       case 'Q':
        {
          System.exit(0);
        } // case
      } // switch
    } // while
  } // main
} // class

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

[45]

QUESTION 3: JAVA PROGRAMMING

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

TestQuestion3XXXX.java

Q 3.1
Code was given in Afrikaans Java version.
2 marks allocated in Ouestion 3.3

//QUESTION 3.1

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

 public class TestCallCentre
{
 public void createSuggestionsFile()
  {
    try
    {
     PrintWriter out = new PrintWriter (new FileWriter ("Suggestions.txt"));
    }

    catch(IOException e)
    {
     System.out.println("Suggestion File Error!!!" + e.getMessage());
    }
  }
```

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

//QUESTION 3.2 (6)

```
public static boolean validateAccNum(String accNo) ✓
    {
       boolean validNo = false;✓
       if (accNo.length() == 7 ✓&& Character.isLetter (accNo.charAt(0))) ✓
       {
          validNo =true; ✓
       }
       return validNo; ✓
    }
```

Q 3.2
(1) Method heading
(1) Initialise Boolean value
(2) if statement
(1) Change Boolean value
(1) Return value

```
Accept: if … else instead of initializing Boolean
Accept: Any correct code to obtain the first character
Accept: One statement in method returning Boolean, e.g.
return (accNo.length()&& …….);
```

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

//QUESTION 3.3 (24) + 2

```
String [] refNumbers = new String [100];
String [] query = new String [100];

int countRefNumbers =0;
int countComplaints = 0;
int countAccounts = 0;

public void referenceNumbers()
    {
      createSuggestionsFile();✓
      try
       {
```

```
Scanner sc = new Scanner (new FileReader✓ ("Data.txt"));✓
while (sc.hasNext())✓
{

 String line = sc.nextLine();✓
 int psnColon = line.indexOf(":");✓
 int lastPsnColon = line.lastIndexOf(":");✓
 String accNo = line.substring(psnColon+1,lastPsnColon);✓
 int psnHash = line.indexOf("#");
 String date = line.substring(lastPsnColon+1,psnHash); ✓
 String querie = line.substring(psnHash+1); ✓

 char type =line.charAt(0); ✓
 if (validateAccNum(accNo)) ✓
 {
   if (type == 'S') ✓
   {

    try
    {
     PrintWriter out = new PrintWriter(new FileWriter
                      ("Suggestions.txt",true)); ✓
     out.println(line.substring(psnColon+1)); ✓
     out.close();✓
    }
    catch(IOException e)
    {
    System.out.println("Suggestion Error!!!"
                         +e.getMessage());
    }
   }//if

   else
    {
     type = Character.toUpperCase(type);
     switch(type)✓
      {
     case 'C': countComplaints++;✓
               refNumbers[countRefNumbers]✓ =
               "C"+countComplaints+"-"+accNo+"-"+date;✓
                break;
     case 'A': countAccounts++;✓
               refNumbers[countRefNumbers]  =
               "A" ✓+countAccounts+"-"+accNo+"-"+date;
               break;
      }//switch
     query[countRefNumbers] = querie;✓
     countRefNumbers++;✓
     }//else
    }//if
    }//while
 }//try
     catch(FileNotFoundException e)
    {
      System.out.println("Error!!!"+e.getMessage());
    }
System.out.println("Reference Numbers\n================");
for (int i = 0; i<countRefNumbers;i++)✓
 {
    System.out.println(refNumbers[i]);✓
 }//for
}
```

Q 3.3
(1) Call Create Suggestions file
(2) Open file Data.txt
(1) While not eof
(1) Read a line
(1) Extract type of issue
(1) Extract account Num
(1) Extract date
(1) Extract issue
(1) Call validateAccNo
(1) Check if suggestion
(1) Open file for writing
(1) Write suggestion to file
(1) Close file
(1) Inside else Increase ref number counter
(1) Extract first letter of issue
(1) Check category
(2) Create ref number for complaint
(2) Create ref number for Account query
(1) Create issue reference number
(1) Store reference number in array
(1) Store query in array
(2) Display ref numbers

Accept: Open Suggestion file once above while, not inside loop.
While reading from file with { } = 1 mark, no marks with no { }
Accept any part of the text written to the Suggestions file.
Accept the whole word for checking purposes.
Accept using text files instead of arrays

```
//============================================================================
//QUESTION 3.4                                    (8)
   public void searchAccount()
  {
     Scanner kb = new Scanner (System.in);
     System.out.println("Enter the account number to query");

     String accNumber = kb.next();
     boolean found = false; ✓
     System.out.println();
     if !(validateAccNum(accNumber)) ✓
          System.out.println("Invalid account number
                                      entered");✓

    else
     {
      for (int i = 0; i< countRefNumbers; i++)✓
      {
        if (refNumbers[i].contains(accNumber)) ✓
        {
          System.out.println(refNumbers[i]+"\t"+
                                          query[i]);✓
          found =true; ✓
        }//if
      }//for

     if (!found)
     {
        System.out.println("No issues have been reported for account
                                 number:"+accNumber);                } ✓
     }
    } // else
  }
```

Q 3.4
(1) Initialise Boolean variable
(1) Validate accNumber
(1) Loop
(1) Check if num entered matches ref num in array
(1) Display ref num and query
(1) Change Boolean to true
(1) Display message if input value not found
(1) Display message if invalid acc num is

```
Accept: if(refNumbers[i].indexOf(accNumber)>-1)
Accept: Extract the account number and then compare
```

```
//============================================================================
public static void main (String [] args)
     {
        TestCallCentre obj = new TestCallCentre();
        Scanner input = new Scanner(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 = input.nextLine().toUpperCase().charAt(0);
             boolean optionA = false;
            if (ch == 'A')
            {
               obj. referenceNumbers();
               optionA = true;
            }
            if ( ch == 'B')
            {
           if (!(optionA)
            {
               System.out.println("\n\nFirst choose Option A ");
             }
           else
            {
               obj.searchAccount();
            }

          }
          if (ch == 'Q')
          {
             System.exit(0);
          }
         } // while
      }

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

[40]

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

In general:
Subtract only 1 mark for a common error made throughout all SQL's.
If no mark allocated in memo but a mistake was made, subtract a maximum of one mark

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	SELECT *✓ FROM tblDams✓ ORDER BY HeightOfWall✓ ASC	**3**	
1.2	Input Province✓ SELECT TownName, Population✓ FROM tblTowns WHERE✓ Population > 100000✓ AND✓ Province = "' + pr + '"'✓ **Accept:** Province LIKE **Last mark: allow for a quoted string "100000" incorrect, must not be quoted** **Order of selected fields not important**	**6**	
1.3	SELECT DamID, DamName✓, YEAR(NOW())✓ - YearCompleted✓) AS Age✓, ROUND (DamLevel / Capacity * 100✓, 1✓) AS Percentage✓ FROM tblDams **Note**: SELECT DamID, DamName ….. FROM tblDams – **(one concept, 1 mark).** **New field names(all questions)-do not penalise if not exactly same text as suggested in question.** **Accept:** YEAR(DATE()) **or** 2011 **Accept:** format(DamLevel / Capacity * 100,'0.0') **Accept: correct use of int to round down to 1 dec** Int((DamLevel / Capacity * 100)*10)/10	**7**	
1.4	SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY Province✓ **Accept:** WaterRestrictions = YES **or** NO **Accept:** Count**(Any field from table instead of *)** **Accept:** WHERE WaterRestrictions **(without = true)** **GROUP BY has to be at the end.**	**5**	
1.5	sql = "SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓ WHERE tblTowns.DamID✓ = tblDams.DamID✓ AND River✓ = "Vaal River"✓ **Accept:** GROUP BY Province **at the end of the SQL statement instead of** DISTINCT Province **Accept: Inner join to join tables:** …FROM tblDams INNER JOIN tblTowns ON tblDams.DamID = tblTowns.DamID….. **Accept:** LIKE 'Vaal%' **Note: Subtract 1 mark for syntax error e.g. leaving out the table names or the dot, etc.** **Accept use of aliasses e.g. tblTowns A, tblDams B**	**7**	

1.6	UPDATE tblTowns✓ SET✓ WaterRestrictions = True✓ WHERE Province = "North West"✓ **Accept:** Province LIKE **Accept:** WaterRestrictions = YES **or** NO **North West must be spelt correctly, quoted**	**4**	
1.7	DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓ **Accept:** DELETE *	**3**	
	TOTAL:	**35**	

ADDENDUM B

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 4 - 8)

CENTRE NUMBER: ……	EXAMINATION NUMBER: ……

QUESTION 2: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Constructor:** (3) Assign parameter values to private fields	**3**	
2.1.2	**calculateTotal:** (1) Initialise total (1) for loop (1) Add array element to total (1) return total	**4**	
2.1.3	**calculateAve:** (1) Data type of return value real (or double) (1) Correct calculation	**2**	
2.1.4	**determineHighDay:** (1) Return type int (1) Initialise iHighDay (1) Initialise iHighAmount (1) For loop (1) if statement (1) change iHighDay (1) Change iHighAmount (1) return iHighDay	**8/2=4 (rounded up)**	
2.1.5	**determineHighRisk:** (1) Initialise count (1) Loop (1) if array element > dayLimit (1) increment count (3) if ave > dayLimit or count > 2 (1) return true (1) else return false	**9**	
2.1.6	**toString:** (1)Headings + new line (1)Day numbers (1)Heading (1)Values from array (1)Strings concatenated	**6**	
2.2			
2.2.1	(2) Declare a single object variable	**2**	
2.2.2	(1) Call the toString method of the object (1) Display label (1) Call calculateTotal method (1) Call calculateAverage method	**4**	
2.2.3	(1) Call calculateAve method (1) Display average (1) Loop (1) if (2) Display number & difference	**6**	
2.2.4	(1) Input dayLimit (1) Call toString (1) Call calculateHighDay (1) If statement (1) Display correct message	**5**	
	TOTAL:	**45**	

ADDENDUM C

QUESTION 3: DELPHI PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 9 - 13)

CENTRE NUMBER: ……..	EXAMINATION NUMBER: ……..		
QUESTION 3: DELPHI – MARKING GRID			
QUESTION	**ASPECT**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	Code was given in Afrikaans Java version 2 marks re-allocated in Question 3.3		
3.2	(1) Sub-program heading (1) Initialise Boolean value (2) if statement (1) Change Boolean value (1) Return Boolean value	**6**	
3.3	**Option A:** (1) Call method to create Suggestion file (2) Open Data file to read from (1) Open Suggestion file to write to (1) While not eof (1) Read a line (1) Extract type of issue (1) Extract account Num (1) Extract date (1) Extract issue (1) Call validateAccNo (1) Check if suggestion (1) Write suggestion to file (1) Inside else Increase ref number counter (1) Extract first letter of issue (1) Check category (2) Create ref number for complaint (2) Create ref number for Account query (1) Create issue reference number (1) Store reference number in array (1) Store query in array (2) Display ref numbers (1) Close Suggestion file	**24 + 2**	
3.4	**Option B:** (1) Initialise Boolean variable (1) Validate acc number (1) Display message if invalid acc num is entered (1) Inside for loop (1) Check if num entered in array (1) Display ref num and query (1) Change Boolean value (1) Display message if input value not found	**8**	
	TOTAL:	**40**	

ADDENDUM D

QUESTION 1: JAVA – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	EXAMINATION NUMBER: ……

QUESTION 1: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
In general: **Subtract only 1 mark for a common error made throughout all SQL's.** **If no mark allocated in memo but a mistake was made, subtract a maximum of one mark**			
1.1	SELECT *✓ FROM tblDams✓ ORDER BY HeightOfWall✓ ASC	**3**	
1.2	Input Province✓ SELECT TownName, Population✓ FROM tblTowns WHERE✓ Population > 100000✓ AND✓ Province = '" + pr + "'"✓ **Accept:** Province LIKE **Last mark: allow for a quoted string** **100000 must not be quoted** **Order of selected fields not important**	**6**	
1.3	SELECT DamID, DamName✓, YEAR(NOW())✓ - YearCompleted✓) AS Age✓, ROUND (DamLevel / Capacity * 100✓, 1✓) AS Percentage✓ FROM tblDams **Note**: SELECT DamID, DamName ….. FROM tblDams – **(one concept, 1 mark).** **New field names(all questions)-do not penalise if not exactly same text as suggested in question.** **Accept:** YEAR(DATE()) **or** 2011 **or** YEAR(NOW) **Accept:** FORMAT (DamLevel / Capacity * 100, '0.0') **Accept: correct use of int to round down to 1 dec** Int((DamLevel / Capacity * 100)*10)/10	**7**	
1.4	SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY Province✓ **Accept:** WaterRestrictions = YES **or** NO **Accept:** Count**(Any field from table instead of *)** **Accept:** WHERE WaterRestrictions **(without = true)** GROUP BY **has to be at the end.**	**5**	
1.5	SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓ WHERE tblTowns.DamID✓ = tblDams.DamID✓ AND River✓ = 'Vaal River'✓ **Accept:** GROUP BY Province **at the end of the SQL statement instead of** DISTINCT Province **Accept: Inner join to join tables:** …FROM tblDams INNER JOIN tblTowns ON tblDams.DamID = tblTowns.DamID….. **Accept:** LIKE 'Vaal%' **Note: Subtract 1 mark for error e.g. leaving out the table names or the dot, etc.** **Accept use of aliases e.g. tblTowns A, tblDams B**	**7**	

1.6	UPDATE tblTowns ✓ SET✓ WaterRestrictions = True✓ WHERE Province = 'North West'✓ **Accept:** Province LIKE **Accept: WaterRestrictions = YES or** NO North West must be spelt correctly, quoted	**4**	
1.7	DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓ **Accept:** DELETE *	**3**	
	TOTAL:	**35**	

ADDENDUM E

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 14 - 18)

CENTRE NUMBER: ……	EXAMINATION NUMBER: ……

QUESTION 2: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Constructor:** (3) Assign parameters to private fields	**3**	
2.1.2	**calculateTotal:** (1) Initialise total (1) for loop (1) Add array element to total (1) return total	**4**	
2.1.3	**calculateAve:** (1) Data type of return value is real (or double) (1) Correct calculation	**2**	
2.1.4	**determineHighDay:** (1) Return type int (1) Initialise highDay (1) Initialise highAmount (1) For loop (1) if statement (1) change highDay (1) change highAmount (1) return highDay	**8/2=4 (rounded up)**	
2.1.5	**determineHighRisk:** (1) Initialise count (1) Loop (1) if array element > dayLimit (1) increment count (3) if ave > dayLimit or counter > 2 (1) return true (1) else return false	**9**	
2.1.6	**toString:** (2)Headings + new line (1) Day numbers (2)Heading (1)Values from array (2)Strings concatenated	**6**	
2.2			
2.2.1	(2) Declare a single object variable	**2**	
2.2.2	(1) Call the toString method of the object (1) Display label (1) Call calculateTotal method (1) Call calculateAverage method	**4**	
2.2.3	(1) Call calculateAve method (1) Display average (1) Loop (1) if (2) Display number & difference	**6**	
2.2.4	(1) Input dayLimit (1) Call toString (1) Call calculateHighDay (1) If statement (1) Display correct message	**5**	
	TOTAL:	**45**	

ADDENDUM F

QUESTION 3: JAVA – PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 19 - 22)

CENTRE NUMBER: ……..	EXAMINATION NUMBER: …….

QUESTION 3: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
3.1	Code was given in Afrikaans Java version 2 marks re-allocated in Question 3.3		
3.2	(1) Method heading (1) Initialise Boolean value (2) if statement (1) Change Boolean value (1) Return Boolean value	**6**	
3.3	**Option A:** (1) Call method to create Suggestions file (2) Open Data file to read from (1) While more text to read (1) Read a line (1) Extract type of issue (1) Extract account Num (1) Extract date (1) Extract issue (1) Call validateAccNo (1) Check if suggestion (1) Open file for writing (1) Write suggestion to file (1) Close file (1) Inside else Increase ref number counter (1) Extract first letter of issue (1) Check category (2) Create ref number for complaint (2) Create ref number for Account query (1) Create issue reference number (1) Store reference number in array (1) Store query in array (2) Display ref numbers	**24 + 2**	
3.4	**Option B:** (1) Initialise Boolean variable (1) Validate accNumber (1) Inside loop (1) Check if num entered matches ref num in array (1) Display ref num and query (1) Change Boolean to true (1) Display message if input value not found (1) Display message if invalid acc num is entered	**8**	
	TOTAL:	**40**	

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P2

NOVEMBER 2011

MARKS: 180

TIME: 3 hours

This question paper consists of 19 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 software development	(49)
SECTION E: Integrated scenario	(43)

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 Copyrighted software provided to a user at no cost is known as …

A shareware.
B freeware.
C wrap ware.
D adware. (1)

1.2 The maximum amount of data that can be transmitted over an electronic communication channel during a given period of time is known as …

A bandwidth.
B a connection.
C protocol.
D frequency. (1)

1.3 A … is a recorded audio file stored on a website that can be downloaded to a computer or a portable media player.

A blog
B wiki
C podcast
D portal (1)

1.4 The procedure followed to identify the identity of the sender of an e-mail is known as a digital …

A handshake.
B signature.
C certificate.
D fingerprint. (1)

1.5 … is the name of a well-known and widely used open-source operating system used on desktop computers.

A Windows Vista
B Linux
C Unix
D Symbian (1)

1.6 An operating-system process that sends documents to be printed to a buffer instead of directly to the printer is called …

A spoofing.
B buffering.
C spooling.
D formatting. (1)

1.7 Which ONE of the following software items does NOT allow a user to perform maintenance-type tasks on a computer?

A Defragmenter
B Uninstaller
C Windows Explorer
D Compiler (1)

1.8 In object-oriented programming, which ONE of the following will NOT be regarded as a suitable object to create in a program for a library?

A Book
B Librarian
C Price of a book
D Member of the library (1)

1.9 Which ONE of the following files will probably contain some music?

A abc.pdf
B abc.wav
C abc.doc
D abc.jpeg (1)

1.10 The component that uses battery power to store the configuration information of a computer is known as the …

A BIOS.
B RAM.
C CMOS.
D OS. (1)

TOTAL SECTION A: 10

SCENARIO

Yellowdale Secondary School held an ICT expo week. Schools and leading IT industry specialists were invited to create an exhibit or to give a presentation at the expo.

One of the themes for the learners was: *Design a modern computer that will improve the overall performance of your school's IT laboratory (lab).*

Each of the participants was assigned a stall in the school hall equipped with the required hardware to display or exhibit their presentations.

You are a Grade 12 IT learner and were appointed as a member of the judging panel, in order to advise the panel on the various aspects of ICT.

SECTION B: HARDWARE AND SOFTWARE

QUESTION 2

2.1 The statement above the first stall read:

'For any electronic computing device to be functional, it requires a motherboard with at least a CPU!'

This group of learners presented the design below for their modern motherboard. (Refer to DIAGRAM 1.)

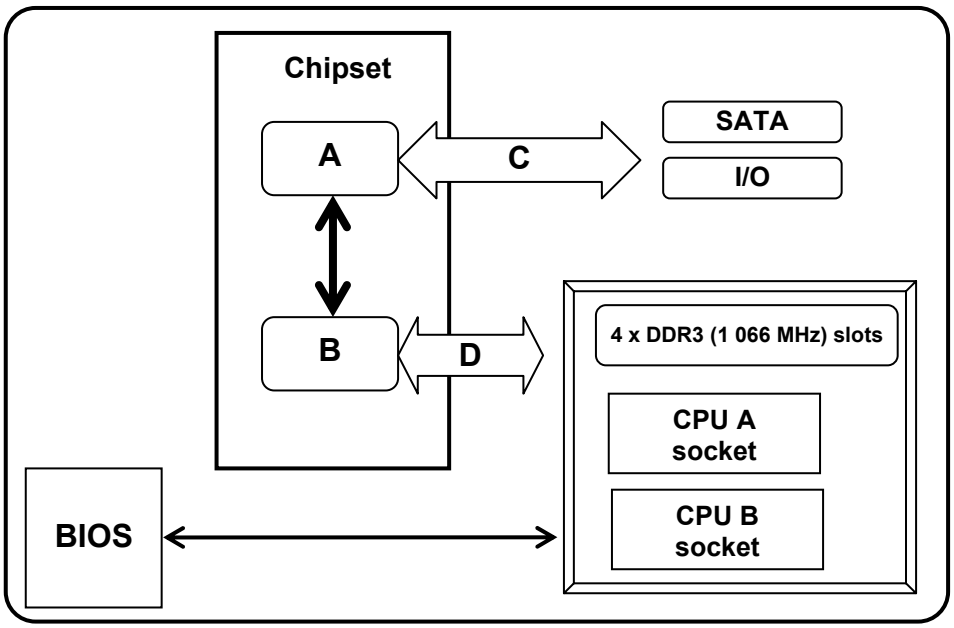


DIAGRAM 1: DESIGN OF A MODERN MOTHERBOARD

2.1.1 Why does a motherboard need a chipset? (2)

2.1.2 The basic design principles for chipsets were followed. Identify the TWO parts of the chipset by writing down the names of component A and component B. (2)

2.1.3 Structures C and D are different bus types.

(a) Explain what a *bus* is in terms of computer hardware. (2)

(b) Which bus (C or D) is known as the system bus? (1)

(c) Name and briefly explain the function of any TWO system buses. (4)

2.1.4 State THREE advantages of using a USB port. (3)

2.1.5 Cache memory, the ALU and registers are three components of the CPU.

(a) What is the function of the ALU? (1)

(b) Does any change in the size of the registers influence the CPU's performance? Motivate your answer. (2)

2.1.6 How does overclocking affect the speed of the CPU? (1)

2.1.7 A group member states that the CPU (Intel Pentium 2.0 GHz) can be replaced by any newer version of the CPU that the user wants. Is this possible? Motivate your answer. (2)

2.1.8 One of the factors that has an influence on the performance of a processor is the design of its instruction set.

(a) Explain what the *instruction set* of a processor is. (1)

(b) Give ONE example of an extended instruction set. (1)

2.1.9 Two CPUs on the motherboard will generate a lot of heat. Name TWO ways to prevent the CPUs from overheating. (2)

2.2 The learners from another stall emphasised some important aspects about secondary storage.

2.2.1 Users should be aware that metadata exists for each file.

(a) Explain what *metadata* is. (1)

(b) Give ONE example of metadata that will be provided with a FAT32 file system. (1)

2.2.2 DIAGRAMS 2a and 2b below show the properties of a zipped file saved on an NTFS file system.

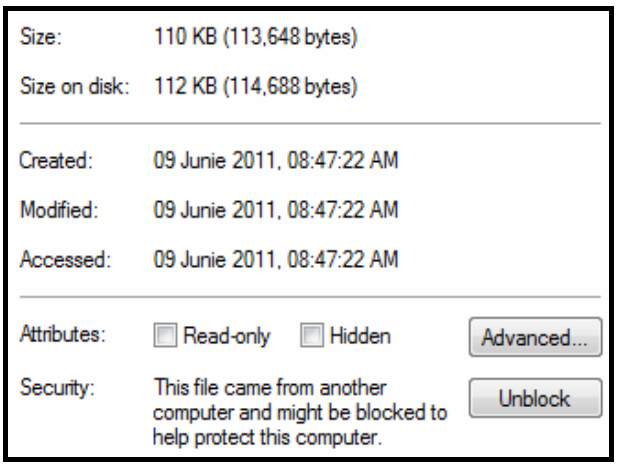


DIAGRAM 2a

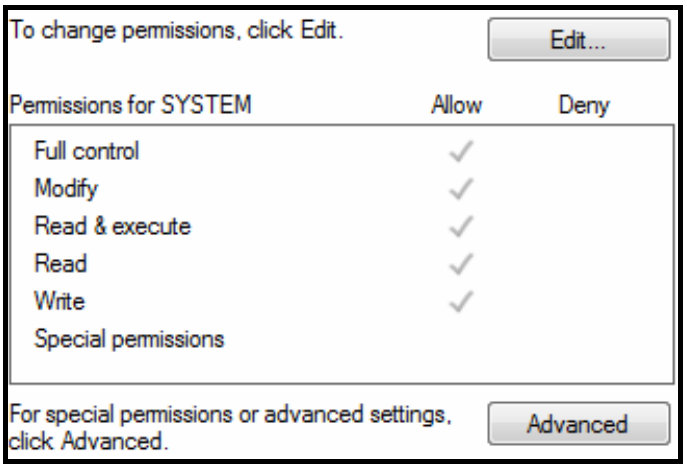


DIAGRAM 2b

(a) State TWO advantages of using an NTFS file system over using a FAT32 file system. (2)

(b) Using DIAGRAM 2a, briefly explain why there is a difference in the size of the file (110 kB) and the size of the file on the hard drive (112 kB). (2)

(c) Changing the 'Hidden' attribute of a file to 'true' will not always prevent other users from modifying the file.

Suggest TWO other ways in which one can prevent other users from modifying a file. (2)

2.2.3 DIAGRAM 3 below is a representation of the management console of the hard drive of a computer.

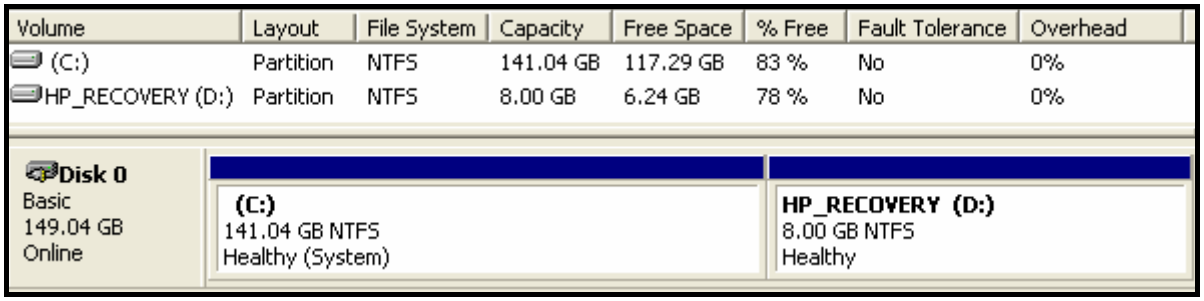


DIAGRAM 3: MANAGEMENT OF DISK 0

(a) Explain what is meant by the term *partition*. (1)

(b) State TWO advantages of having the hard drive partitioned. (2)

(c) Partitioning should take place before the logical formatting of the hard drive is done. Explain the purpose of the logical formatting of a hard drive. (2)

2.2.4 Hard drives store a huge amount of data. For some users, like banks, their data is critical and therefore RAID technology is often used.

(a) What is *RAID technology*? (1)

(b) Briefly explain how RAID Level 5 works. (2)

2.3 The computers in the IT lab are currently networked, as shown in DIAGRAM 4 below. Another group of learners recommends the use of a star topology for the IT lab's network.

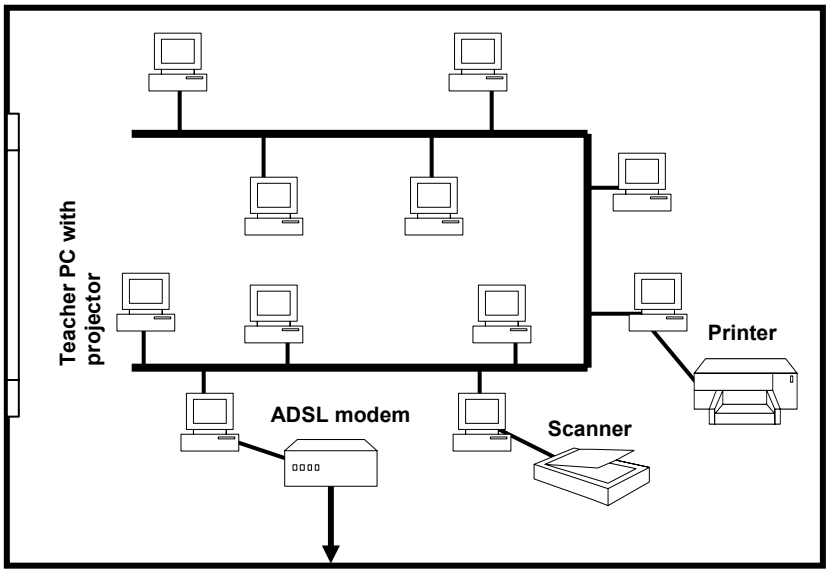


DIAGRAM 4: CURRENT NETWORK LAYOUT OF IT LAB

2.3.1 What does the term *topology* refer to in relation to computer networks? (1)

2.3.2 Identify the network topology currently used (DIAGRAM 4). (1)

2.3.3 In order to change the current network topology to a star topology, a switch and additional cables are required.

(a) What type of cables will be required? (1)

(b) State ONE disadvantage of using the type of cable named in QUESTION 2.3.3(a). (1)

(c) State ONE advantage and ONE disadvantage of using a star topology. (2)

2.3.4 It is recommended that a network administrator should be appointed. State THREE functions of a network administrator. (3)

2.3.5 One of the members proposed the use of thin clients.

(a) What is meant by a *thin-client computer network*? (1)

(b) State TWO disadvantages of a thin-client computer network. (2)

2.4 One of the groups indicated that the software requirements of the IT lab needed to be addressed.

2.4.1 Is a device driver part of system software or application software? (1)

2.4.2 The school acquired a computer with a 32-bit bus system. The principal wants to install a 64-bit multithreading operating system on this computer.

(a) What is meant by the term *multithreading*? (2)

(b) Give ONE reason why is it not advisable to install the new 64-bit operating system on the computer acquired by the school. (1)

2.4.3 'Why pay for software?'

The statement above is one of the slogans of the stall.

State TWO disadvantages of using open-source software in the network environment of the school. (2)

TOTAL SECTION B: 58

SECTION C: APPLICATIONS AND IMPLICATIONS

QUESTION 3: e-COMMUNICATION

One of the groups of learners was requested to man an e-communication stall at the expo where visitors can get information and advice on e-communication issues.

3.1 The use of the Internet is an essential part of life today. Prospective Internet users often have questions on how to get connected to the Internet.

3.1.1 Users are often advised to get an ADSL (asymmetric digital subscriber line) connection rather than a dial-up connection. Briefly explain what an *ADSL connection* is. (3)

3.1.2 A 3G connection is another popular way of connecting to the Internet.

(a) Do you need a modem for a 3G connection? Explain your answer. (1)

(b) What type of technology is used with a 3G connection? (1)

3.1.3 It is possible for someone else to use your Internet connection if you have a wireless connection. How can you detect that this is taking place? (1)

3.2 E-mail is one of the most popular uses of the Internet, but users are often frustrated by the lack of netiquette. What is *netiquette*? Give an example as part of your answer. (2)

3.3 Businesses cannot be competitive today without doing business electronically (e-commerce). Security is always an issue when conducting business electronically.

3.3.1 Briefly explain how the secure socket layer (SSL) can ensure that private electronic communication takes place. (2)

3.3.2 All web pages do not use SSL. Why not? (1)
[11]

QUESTION 4: SOCIAL AND ETHICAL ISSUES

The learners manning the e-communication stall have to make computer users aware of the responsible use of computers when communicating via electronic media.

4.1 Many companies monitor or intercept employees' e-mail messages. Do you think this practice is acceptable? Briefly motivate your answer. (2)

4.2 Computer users often have to spend long hours working on their computers.

4.2.1 Name TWO possible effects that working long hours on a computer could have on a person's health. (2)

4.2.2 Computer users should be aware of ergonomics. Explain what *ergonomics* is. (2)

4.3 It is estimated that about 1 billion computers were discarded during 2010. E-waste, such as old computers, contains toxic elements.

Suggest TWO ways in which e-waste can be dealt with more effectively in order to avoid pollution. (2)

4.4 In several recent high-profile cases, major news sources have published purposefully altered photos. Explain why this practice can be regarded as fraud. (1)
[9]

TOTAL SECTION C: 20

SECTION D: PROGRAMMING AND SOFTWARE DEVELOPMENT

QUESTION 5: ALGORITHMS AND PLANNING

The organisers of the expo have recruited a team of software developers, analysts and engineers to coordinate academic games and activities to encourage local learners to improve their knowledge of software design concepts. Learners from different schools have been invited to interact with these experts and to participate in these activities.

5.1 The first activity is a matching exercise that learners have to complete. The learner that completes the exercise first, with all the answers correct, will be rewarded with a special prize.

Choose a description from COLUMN B that matches a term in COLUMN A. Write only the letter (A–E) next to the question number (5.1.1–5.1.5) in the ANSWER BOOK, for example 5.1.6 F.

COLUMN A		COLUMN B	
5.1.1	Data type	A	an object keeping its fields/ data private
5.1.2	Data structure	B	specifies the area in which a variable can be used
5.1.3	Scope of a variable	C	determines the possible values that can be stored in a variable
5.1.4	Encapsulation	D	specifies whether a field is private or public
5.1.5	Access modifier	E	organises and stores related data

(5 x 1) (5)

5.2 The software developers emphasise the importance of modular programming in an object-oriented environment.

5.2.1 Explain what *modular programming* is. (2)

5.2.2 State THREE advantages of modular programming. (3)

5.2.3 Would you refer to a constructor as being a method? Motivate your answer. (3)

5.2.4 Name ONE significant difference between a *private method* and a *public method*. (2)

5.2.5 Name ONE significant difference between an *accessor method* and a *mutator method*. (2)

5.3 As one of the activities, the learners have been divided into groups where each group will be given a topic related to programming. The learners in each group will need to formulate questions related to that specific topic. Your group has been given the topic of error handling and debugging.

5.3.1 Name the THREE common types of errors in programming. (3)

5.3.2 Explain the concept *exception handling*. (2)

5.3.3 Give TWO reasons why validation of data entered into a program is necessary. (2)

5.3.4 Normal test data often does not reveal possible incorrect outputs of a program.

Name TWO types of test data that will assist in ensuring correct outputs. (2)

5.3.5 Name TWO good programming practices that may be followed to make the debugging of a program easier. (2)

5.4 The IT expo has a variety of stalls to keep the community entertained for the duration of the expo. A database has been designed to store information on the different stalls at the expo. The database contains two tables, namely **tblCategories** and **tblStalls**, which are related. Below are screen shots of the content of the two tables in the database.

tblCategories

CategoryName
Books & Resources
Entertainment
Food
Hardware
Software

tblStalls

StallID	StallOwner	StallCategory	Indoors
1	Maxie Kenton	Food	☑
2	Henry Potgieter	Hardware	☑
3	Penny Swartkop	Food	☐
4	Brandon Naidoo	Entertainment	☐
5	Thabisile Nkosi	Food	☑
6	Olivier Moses	Books & Resources	☑

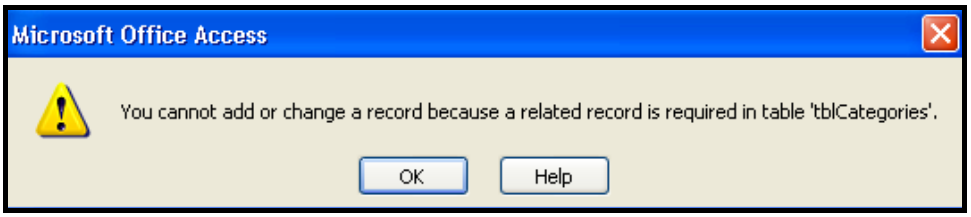
5.4.1 Name the type of relationship between the **tblCategories** and **tblStalls** tables. (1)

5.4.2 Which field in the **tblStalls** table would you suggest as a suitable foreign key? (1)

5.4.3 The data capturer is attempting to enter a new record into the **tblStalls** table. The data to be captured are as follows:

Field Name	Content
StallOwner	Kensley Jenkins
StallCategory	Books
Indoors	Yes

The following error message is displayed after the record has been captured:



(a) Explain the reason for this error. (2)

(b) There are a number of ways of preventing the capturing of incorrect data in this example. Name TWO ways. (2)

(c) Name the term that enforces the input of correct data when using multiple tables with relationships. (1)

5.4.4 All categories of stalls are not represented at the expo. For example, there are no stalls representing the software category. The expo director wants a list of all the stall categories that are represented.

The list should appear as follows:

	StallCategory
	Books & Resources
	Entertainment
	Food
▶	Hardware

Indicate whether the following statement is TRUE or FALSE. Write only 'true' or 'false'.

Any ONE of the SQL statements below may be used to display the list given above.

SQL1→ SELECT DISTINCT StallCategory FROM tblStalls

SQL2→ SELECT StallCategory FROM tblStalls GROUP BY StallCategory (2)

5.4.5 State ONE significant advantage of storing data in a database rather than in a text file. (1)

5.5 Computer experts emphasise the importance of the planning stages in programming. One way to plan effectively is to write out an algorithm.

5.5.1 Explain what an *algorithm* is. (1)

5.5.2 Explain why an algorithm designed in pseudocode should not be programming language specific. (2)

5.5.3 A certain university is awarding bursaries to Grade 12 learners to pursue a career in Information Technology.

The learners must adhere to the following criteria:

1. They must be younger than 19 years of age.
2. They must have a minimum of 4 distinctions in Grade 12.
3. One of the distinctions must be in either Mathematics or Information Technology.

NOTE: A mark of 80 or more is required for a distinction.

Study the given segment of pseudocode used to test the conditions stated in the criteria.

IF age < 19 AND mathsMark >= 80 (a) … infoTechMark >= 80 (b) … totalDistinctions (c) … 4

Write only the correct answer next to the letters (a), (b) and (c) in your ANSWER BOOK to complete the statement above correctly. (3)

5.5.4 The expo director is interested in how well the expo was received. The algorithm on the next page was written to count the number of people that rated the expo as either of a high standard or of a low standard.

Any ticket holders at the expo can enter their ticket numbers and a rating. The number 1 is used for a high-standard rating and the number 0 for a low-standard rating. The algorithm will display the total number of high ratings and low ratings separately. Input is terminated when a value of 0 is entered for the ticket number.

Study the algorithm below.

Line Number	Description
1	countHighRatings ← 0
2	countLowRatings ← 0
3	Input ticket number
4	While ticket number not equal to 0
5	Start While loop
6	Input Rating
7	If rating equals 1
8	countHighRatings ← countHighRatings + 1
9	otherwise
10	countLowRatings ← countLowRatings + 1
11	End While Loop
12	Display countHighRatings
13	Display countLowRatings

(a) Explain why a conditional loop is necessary for this application. (1)

(b) Give a reason why the algorithm given above will be caught in an infinite loop. (2)

(c) A single statement needs to be inserted to ensure a successful output.

(i) Write out the single statement that is required. (1)

(ii) Write down the line numbers between which the required statement should be inserted. (1)

TOTAL SECTION D: 49

SECTION E: INTEGRATED SCENARIO

QUESTION 6

6.1 The following heading appears above one of the stalls:

'Why amateur radio endures in a world of tweets'

The excerpt below appears on a poster.

> Somehow it makes little sense that amateur radio continues to thrive in the age of Twitter, Facebook and iPhones. It attracts 60% more users than thirty years ago.
>
> In an article published in *Wired UK*, David Rowan writes:
>
> *'For a start, there is the thrill in establishing a magical person-to-person long-distance radio conversation that no commodified Internet communication can compete with.'*
>
> There are also radio amateurs who are techno buffs; they want to touch anything that is technologically new. Radio amateurs were amongst the first to use voice over Internet protocol, linking radio, the Internet and computers into a global communications network, called EchoLink.
>
> [Adapted from *EngineerIT*, March 2011]

6.1.1 Sending an e-mail or a tweet to another person in another country requires the use of different software.

(a) Name a software application that is required to send an e-mail. (1)

(b) State TWO differences between an *e-mail* and a *tweet*, other than the software required. (2)

6.1.2 An iPhone is classified as a smart phone.

List THREE distinctive features of a mobile phone for it to be classified as a smart phone. (3)

6.1.3 Facebook and Skype are some of the buzzwords used when discussing social networking.

(a) Explain the term *social networking*. (2)

(b) Briefly explain how the use of Facebook can impact negatively on the social life of a teenager. State TWO facts. (2)

(c) What is a *communication protocol*? (2)

(d) What type of communication protocol is used by Skype? (1)

6.2 Linking computers in a global communications network impacts on different sectors of the economy differently.

6.2.1 State THREE advantages of a global communications network for the education sector. (3)

6.2.2 List THREE career options that arose as a result of connecting computers in a global network. (3)

6.2.3 The use of computers for criminal intent is on the increase. The terms below are used to describe people that are a threat to computer systems.

For each term:

- Explain the term. What is it?
- Give a brief description of how such a person can cause damage to another person.

(a) Cyber terrorist (2)

(b) Cyber extortionist (2)

(c) Script kiddie (2)

(d) Wardriver (2)

6.2.4 The concepts (a) and (b) below are associated with threats to a computer system.

For each concept, choose TWO security measures from the list below that will protect the computer system against that specific threat. You may use each security measure only ONCE.

Possible security measures available:

- Backup
- Product activation
- Encryption
- Antivirus
- Biometrics
- UPS
- User names and passwords
- Audit trail
- Authentication

(a) Unauthorised access to confidential information (2)

(b) Back doors (2)

6.2.5 Honeypots help to secure a company's network. Explain how this is accomplished. (2)

6.3 An illegal recording of someone else's work is called piracy.

6.3.1 Name TWO sectors of the economy on which piracy impacts negatively. (2)

6.3.2 Name TWO ways in which an original DVD can be distinguished from a pirated copy. (2)

6.3.3 Bootlegging is one of the main categories of piracy.

(a) Explain the term *bootlegging*. (1)

(b) Give ONE example of a bootleg. (1)

6.4 The use of the Internet created an information overloaded society. Many of the websites on the Internet do not supply the users with reliable information.

6.4.1 Name TWO ways in which we can ensure that the information on a certain web page is trustworthy. (2)

6.4.2 Name TWO ways in which you, as an Internet user, can verify the information provided on the web page. (2)

TOTAL SECTION E: 43
GRAND TOTAL: 180

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P2

NOVEMBER 2011

MEMORANDUM

MARKS: 180

This memorandum consists of 18 pages.

SECTION A: MULTIPLE-CHOICE QUESTIONS

QUESTION 1

1.1	B ✓ (freeware) or A (shareware)	(1)
1.2	A ✓ (bandwidth)	(1)
1.3	C ✓ (podcast)	(1)
1.4	B ✓ (signature) OR C (certificate)	(1)
1.5	B ✓ (Linux)	(1)
1.6	C ✓ (spooling) OR B (buffering)	(1)
1.7	D ✓ (Compiler)	(1)
1.8	C ✓ (Price of a book)	(1)
1.9	B ✓ (abc.wav)	(1)
1.10	C ✓ (CMOS)	(1)

TOTAL SECTION A: 10

SECTION B: HARDWARE AND SOFTWARE

QUESTION 2

2.1 2.1.1 The management of the data transfer ✓ between all the components of the motherboard ✓ is regulated by the chipset.
Accept: 1 mark – to indicate communication or control, speed of data
1 mark - to indicate the components (2)

2.1.2 A = south bridge ✓
B = north bridge ✓ (2)

2.1.3 (a) Bus is a set of wires used to transfer the data/instructions✓ between two components on the motherboard.✓
1 mark – the medium
1 mark – the components (2)

(b) D ✓ (1)

(c) (Any TWO busses – (name✓ and description✓)(2x2))
- **Address bus** → transfers the address of the data/instruction from the CPU to the RAM/one-way communication between CPU and RAM
- **Control bus** → transfers a control signal (read/write) to the memory/one-way communication between CPU and RAM
- **Data bus** → transfers data in a two-way communication between the RAM and CPU

Accept: USB, FSB, PCI, PCI Express, internal bus, BSB (2 marks for the buses and 2 marks for the explanation) (4)

2.1.4 (Any THREE advantages ✓✓✓)
- USB can be used to connect different devices e.g. flash drives, external hard drives.
- USB is hot-pluggable / placed externally
- Plug-and-play applies to USB.
- USB supplies a small amount of electric power to the connected devices.
- USB can make use of daisy chaining connected devices.
- USB provides faster data transfer rates / high speed transfer.
- A popular device, most used device

DO NOT Accept: No drivers required. (3)

2.1.5 (a) The ALU (arithmetic logic unit) is responsible for the processing of logic equations and whole number (integer) calculations. ✓
NB: Any 1 fact
Logical Arithmetical operations
Numerical calculations

(1)

(b) YES ✓ (any correct reason ✓)
- The size of the register dictates the largest number (2^{32} vs. 2^{64}) the CPU can store correctly.
- The size of the register also dictates the maximum size of RAM the CPU can work with.
- Can store more data
- Increases the efficiency/performance

NOTE: If answered NO – no marks ($^0/_2$)
DO NOT Accept: Increases processor speed (2)

2.1.6 The CPU will run at a faster speed than it is designed for. ✓
Accept: faster (1)

2.1.7 NO✓ (any correct reason ✓)
- Newer CPU's might not fit on this motherboard because of different pin configurations, number of cores, etc.
- If you are going to replace the CPU, it will have to be with the same type otherwise you would need to buy a new motherboard.
- Make sure the current heat sink and CPU fan fit on the new CPU otherwise they will need to be changed as well.

Accept: Yes, with a correct motivation

(2)

2.1.8 (a) An instruction set is the basic set of instructions that the CPU recognises and executes.✓
Accept: The computer instead of CPU
Do not accept: A basic set of instructions (1)

(b) (Any valid example of extended instruction set✓)
- MMX
- SSE

Accept: CISC, 3D-Now (1)

2.1.9 (Any TWO valid ways to prevent CPU overheating ✓✓)
- Use a heat sink on the CPU's / thermal paste
- Add additional fans to the computer box.
- Improve circulation by using a bigger box.
- Water cooling / nitrogen cooling / Circulation of liquid
- Soft Cooling, using software to cool the system (2)

2.2 2.2.1 (a) 'Data about data' – metadata refers to additional or extended data on a file structure ✓ **OR** The design and specification of data structures. (1)

(b) (Any ONE example ✓)
Fat32,Filename, extension, file attributes, create time / date, last accessed, last modified, size

Accept: Track number/Genre/Camera model/Date picture was taken/Database design. (1)

2.2.2 (a) (Any TWO advantages of NTFS ✓✓)
- Support additional security and management properties
- Allows the network administrator to set permissions on the files, e.g. who is allowed to read/write/modify the file
- Adds more file attributes to the file
- Provide better security for system files
- NTFS has no restriction on the number of files
- NTFS has no restriction on the size of a file (whereas FAT32 can only handle files up to 2 GB) (2)

(b)
- Size on disc (112 kB) → The cluster size used✓ by the file system is fixed (cluster consists of set number of sectors).
- File size (110 kB) → The file didn't use up the whole cluster when it was saved ✓ – therefore it is smaller than the disc size.
- Accept: Slack
- Any reasonable answer (2)

(c) (Any TWO ways to prevent file modification✓✓)
- Change the *Read only* attribute to true.
- Remove the *full control* file permission on the file.
- Remove the *modify* file permission on the file.
- Change the file permission to *deny full control/modify*.

Accept: Encryption / Passwords (2)

2.2.3 (a) Partition – to split one hard drive into two or more logical drives each with their own boot up sector. ✓
Accept: Creating virtual drives
Split 1 drive into 2 or more drives (1)

(b) (Any TWO advantages of partition ✓✓)
- Allows the installation of two different operating systems on one physical hard drive
- Allows user to split data and operating system files on the two logical hard drives / Easier backups created
- Allows the user to have two different file systems (FAT32 and NTFS) on the two logical drives

Accept: Improves security / protection / prevention of virus spreading
Formatting can take place on only one hard drive (2)

(c) The file system is placed on the disk. ✓✓
The file system is used by the operating system as an index to keep track of what is saved and where is has been saved on the hard disk.

(**OR** any similar explanation) (2)

2.2.4 (a) RAID is a way to spread data over many discs to make data recovery easier.✓
Accept: many hard drives are seen as a single drive
RAID – Redundant Array of Independent/ Inexpensive Disks. (1)

(b) RAID Level 5 →
- Data is broken into/split into blocks and distributed between disks ✓
- The RAID controller generates and stores a 'parity' block which is distributed across the disks. / Parity bit is stored✓

OR
Only one mark for: Striping with parity. (2)

2.3 2.3.1 Topology is the physical arrangement of the computers/devices in the network ✓
Accept: Layout (1)

2.3.2 Bus ✓ (1)

2.3.3 (a) (Any ONE ✓)
- UTP
- Fibre optic (1)

(b) **UTP** (Any ONE disadvantage ✓)
- Susceptible to EMR or eavesdropping / Crosstalk
- Signal strength can decrease rapidly with cable length
- Accept: Limited bandwidth/slower than fibre optic

OR

Fibre optic (Any ONE disadvantage)
- High cost to install
- Difficult to install – needs network technician (1)

(c) (Any ONE **ADVANTAGE** of star topology ✓)
- Easy to add new computers
- Easy to troubleshoot network
- Accept: Computers are still operational if 1 workstation malfunctions.
  Faster than a bus topology.
  Improves security.

(Any ONE **DISADVANTAGE** of star topology ✓)
- If the central hub/switch fails the entire network is down.
- A lot of cabling is required to set up the network. (2)

2.3.4 (Any THREE functions – network administrator✓✓✓)
- Creates user accounts
- Deletes user accounts
- Maintenance of user accounts
- Set up security
- Maintain data integrity
- Install/Uninstall programs
- Making of backups/Implementing a backup policy
- Development of network
- Network documentation
- Accept: Any reasonable explanation (3)

2.3.5 (a) Thin-client computer network is a network where application software, data and processing occurs on the server rather than on the client PC.✓
Accept: The client/computer is dependent on the server. (1)

(b) (Any TWO disadvantages of a thin-client computer network ✓✓)
- A powerful server needs to be bought.
- Use of multimedia is restricted.
- Less flexible on certain operating systems – the programs need computers with their own resources to execute and will not run on thin clients.
- Accept: The computers cannot run as standalone computers
- Accept: Slower than fat clients
- Accept: Clients can only use software from the server. (2)

2.4 2.4.1 System software ✓ (1)

2.4.2 (a) Multithreading → When a program is broken up into different threads✓, which run independently✓, seemingly at the same time.
Accept: Example only – 1 mark (2)

(b) (Any valid reason) ✓
- The bus size of the computer (32-bit) and the bus size the OS (64-bit) is capable of addressing is different.
- The 64-bit OS will not function.

Accept: Problem obtaining 64 bit drivers / software
Accept: any valid explanation (1)

2.4.3 (Any TWO disadvantages of open-source software in a school network✓✓)
- Risk of no support / Nobody is responsible if something goes wrong
- Higher skills required to use some of the software
- Different versions of the same software are available on the Internet
- Source code of software is available – the learners may obtain access to sensitive information/data in a school network
- Security
- Reliability
- Does not support all software
- Compatibility problems (2)

TOTAL SECTION B: 58

SECTION C: APPLICATIONS AND IMPLICATIONS

QUESTION 3: e-COMMUNICATION

3.1 3.1.1 An ADSL line is a ***permanent digital connection*** ✓ that is a single line that is split using multiplexing so that ***both voice and data*** ✓ can be used on ***the same line***. ✓

Accept: (Any 3 of the following)
- High Speed Connection
- Digital connection
- Full time/permanent connection
- Allows simultaneous voice/data
- Fixed monthly rate (3)

3.1.2 (a) (Yes/No → mark allocated for valid explanation✓)

NO → You need a 3G card for your laptop/PC or any 3G capable electronic device e.g. cellphone/PDA.

OR

YES → A modem is required to connect to the Internet – the cellphone/3G card acts as the modem. (1)

(b) Cellphone technology✓/Mobile technology (1)
Accept: Radio waves, wireless
Do not accept: Wifi, WiMax, HSDPA, GPRS

3.1.3 (Any ONE ✓)
- Your Internet connection is slower than normal
- You may notice indicator lights on your wireless router flashing rapidly while you are not connected to your wireless network.
- There is a facility to monitor the computers currently connected to your wireless network – you will notice a computer that does not belong to you but is connected to the network.

Accept: The CAP gets used up too quickly, excessive data used. (1)

3.2 Netiquette → Rules describing acceptable behaviour when sending e-mails or any form of electronic communication✓
(Any acceptable example ✓)
- You should not advertise products using e-mail.
- Do not send lengthy e-mails.
- Do not attach very large documents to your e-mail.
- Avoid spelling errors.
- Do not take part in flame war.
- Do not forward spam or hoaxes
- Do not use only capital letters, etc.
- Do not accept: general comments, must be linked to electronic communication. (2)

3.3 3.3.1 SSL provides encryption of all data ✓that passes between a client and an Internet server. Once the client has a digital certificate✓, the Web browser communicates securely with the client.
Accept: Use of private key / public key
Encryption (2)

3.3.2 (Any ONE ✓)
SSL is:
- More processor intensive
- Bandwidth intensive

Accept: Cost implications
Not always required (1)
[11]

QUESTION 4: SOCIAL AND ETHICAL ISSUES

4.1 YES/NO ✓ and any valid reason ✓

Yes → Any acceptable motivation such as the company pays for the e-mail service and has the right to monitor the use of the e-mail service.

No → Intercepting the e-mails violates the privacy of the employee.

(**OR** any other acceptable answer) (2)

4.2 4.2.1 (Any TWO ✓✓)
- Vision can be affected – computer-vision syndrome – tired, blurry, burning, itchy eyes
- Computer-related repetitive strain injury (RSI)
- Carpal-tunnel syndrome – inflammation of the nerves that connect the forearm to the palm or the wrist

Accept: Anti-social
Overweight and unfit
Back/joint related problems (2)

4.2.2 Ergonomics – the science of incorporating comfort, efficiency and safety into the design ✓ in relation to the human body✓
Accept: User friendly devices – 1 mark (2)

4.3 (Any TWO valid suggestions on e-waste ✓✓)
- Recycle old computers.
- Put legislation in place that enforces a more aggressive approach towards e-waste.
- Force computer companies to be responsible for collecting and recycling their products. (2)

Accept: Extending the use/re – use of computers
Accept: Temporary storage for later disposal

4.4 You are deceiving the public. ✓ (1)
[9]

TOTAL SECTION C: 20

SECTION D: PROGRAMMING AND SOFTWARE DEVELOPMENT

QUESTION 5: ALGORITHMS AND PLANNING

5.1 5.1.1 C ✓

5.1.2 E ✓

5.1.3 B ✓

5.1.4 A ✓

5.1.5 D ✓ (5)

5.2 5.2.1 Modular programming – Dividing a program✓ into separate subprograms ✓called methods, procedures or functions. (2)

5.2.2 (Any THREE advantages of modular programming ✓✓✓)
- Easier to debug/maintain.
- Modules/functions/procedures can be used repeatedly.
- Modules/functions/procedures can be used in other classes/forms.
- No repeating/duplication of code segments – make use of functions/procedures.

Accept: A team of people can work on a module (3)

5.2.3 Yes✓, it is a segment of code or a module. ✓ The code segment can be used / called repeatedly✓ (3)
Also Accept: A constructor is a special method because it can create an instance of a class; it can receive parameters

5.2.4 A private method can be accessed only in the class/unit in which it is declared ✓ whereas a public method can be accessed in other classes/units as well. ✓ (2)

5.2.5 An accessor method is used to return the value of a field/data✓ and a mutator method is used to change the value of a field/data. ✓
Accept: State, content instead of values.
Accessor have a return value and mutators are generally void methods (2)

5.3 5.3.1 Syntax✓, Logical✓ and Runtime✓
Also accept: 1 example of each type of error (3)

5.3.2 Exception handling is a programming construct or computer hardware mechanism designed to handle the occurrence of special conditions or problems that change the normal flow of program execution ✓in order to prevent the program from crashing. ✓
OR
Exception handling is used when abnormal or uncommon errors need to be caught and dealt with to prevent the program from crashing as a result of the error.
Accept: Contains/prevents/catches expected errors

(**OR** any other correct explanation)
Do not accept: A description of validation (2)

5.3.3 (Any TWO reasons for validation ✓✓)
- To ensure the capturing of valid/accurate data
- To prevent the program from crashing
- To prevent errors made by the users
- To ensure a correct output
- To ensure a correct format/range

Do not accept: Examples (2)

5.3.4 Extreme✓, erroneous✓ data/abnormal data/missing data
Accept: An appropriate example to explain each type of error (2)

5.3.5 (Any TWO ✓✓)
- Indentation
- Commenting code
- Modular programming (2)

Accept: Use appropriate, good variable names
Avoid the use of global variables
The use of a debugger
The use of appropriate white spaces / blank lines

5.4 5.4.1 One-to-many relationship✓ (1)

5.4.2 StallCategory✓ (1)

5.4.3 (a) The StallCategory must match one of the categories in tblCategories. ✓✓ (2)
Accept: There is no field called Books in tblCategories
Defies referential integrity
Add a Book category in tblCategories

(b) (Any TWO ✓✓)
Use a
- Combo box / Drop down list
- Validation rule
- Lookup wizard

Accept: Use of radio buttons (2)

(c) Referential Integrity✓ (1)

5.4.4 TRUE (**OR** True) ✓✓ (2)

5.4.5 (Any ONE advantage – using a database vs a text file ✓)
- Can perform queries on data
- Reports and input forms can be created in database
- Easier to validate the format and the range of data

Accept: The use of separate tables.
More organised and easier to read.
Allows features like relationships, queries, reports, forms (1)

5.5 5.5.1 A step-by-step approach to solving a problem.✓ (1)
Accept: A solution/way/plan to solve a problem

5.5.2 The algorithm must be able to be interpreted and understood by programmers using any programming language.✓✓
Accept: It is a plan and not the actual solution. (2)

5.5.3 (a) OR✓ Accept || (1)

(b) AND✓ Accept && (1)

(c) >=✓ (1)
Accept ≥

5.5.4 (a) The number of people participating in the rating is unknown. ✓
Accept: Any reasonable statement that implies an unknown number of times. (1)

(b) (Any ONE of the following reasons ✓✓)
- The value of the ticket number is not being changed inside the loop.
- The ticket number remains the same throughout the algorithm, resulting in Step 4 always being true. (2)

(c) (i) Input ticket number ✓

(ii) (Any ONE of the following places ✓)
- Between Line 10 and Line 11 (inside the while-loop)

(2)

TOTAL SECTION D: 49

SECTION E: INTEGRATED SCENARIO

QUESTION 6

6.1 6.1.1 (a) Any e-mail software ✓
e.g. MS Outlook, GMail, yahoo, Outlook Express, Internet Explorer
Accept: open-source e-mail clients (1)

(b) (Any TWO valid differences ✓✓)
- A tweet is restricted to 140 characters whereas the size of an e-mail message is not restricted.
- A tweet is public by default whereas an e-mail is sent to a specific user.
- A tweet must be shorter
- A tweet cannot have an attachment
- A tweet identifies users to follow
- A email is a more formal form of communicating (2)

6.1.2 (Any THREE properties of a smart phone ✓✓✓)
- Allows you to display maps
- Listen to music – portable media player
- Receives and sends e-mail
- Share photos and videos – built-in camera
- Provides PDA capabilities such as running applications
- GPS capabilities
- Touch Screen
- Has a more powerful/advanced operating system than a normal phone
- Has more memory
- Allows for Internet access
- Instant messaging (3)

6.1.3 (a) Social networking refers to virtual communities that communicate over the Internet. ✓✓
Tool, application or website where relationships or connectedness is explicitly defined.
Also Accept: People and friends instead of a virtual community
Any indication of social communication (2)

(b) (Any TWO valid negative impacts ✓✓)
- Learners don't learn social skills
- No interpersonal relationships
- Internet bullying
- Identity theft
- Distribution of pornography

Accept: School work is neglected
Antisocial
(2)

(c) Communication protocol is a set of rules controlling the communication ✓ between two or more devices using a communication medium. ✓ (2)

(d) VoIP ✓ (1)

6.2 6.2.1 (Any THREE valid advantages for education sector ✓✓✓)

- Not restricted to a classroom situation – making long-distance video conferencing possible
- Access to research material is easier
- Can easily communicate with other 'experts' in the same field of study
- Examinations can be conducted in various centres across the world at the same time, e.g. Microsoft certifications (3)

6.2.2 (Any THREE valid careers in networking ✓✓✓)

- Network administrators
- Network technician
- Website designers
- Webmasters
- Database administrators
- Graphical designers
- Hackers
- Security specialists, etc (3)

6.2.3. All candidates will be awarded 8 marks for this question.

6.2.3 (a) Cyber terrorist

- Someone who uses the Internet/network to destroy or damage computers for political / religious reasons. ✓
- The extensive damage might destroy a nation's air-traffic control system, electricity-generating companies or a telecommunications infrastructure. ✓ (2)

(b) Cyber extortionist

- Someone who uses e-mail as a vehicle for extortion. ✓
- These perpetrators send a company a threatening e-mail message indicating they will expose confidential information, exploit a security flaw, or launch an attack that will compromise the company's network if they are not paid a sum of money. ✓ (2)

(c) Script kiddie

- Has the same intent as a cracker but does not have the technical skills and knowledge. ✓
- Script kiddies are often teenagers who use prewritten hacking and cracking programs to break into computers. ✓ (2)

(d) Wardriver
- Intrusion technique in which an individual attempts to detect wireless networks via their mobile devices while driving a vehicle through areas they suspect have a wireless network. ✓
- The damage caused to other people range from using Internet data cap; having access to confidential material, etc.✓ (2)

6.2.4 (a) (Any TWO from the list provided ✓✓)
(Unauthorised access to confidential information)
- Biometrics
- User names and passwords
- Encryption (2)

(b) All candidates to be awarded 2 marks for this question.

(Any TWO from the list provided ✓✓)
(Back doors)
- Backup✓
- Audit trail ✓
- Authentication (2)

6.2.5 A honeypot is a vulnerable computer that is set up to entice an intruder to break into it. ✓✓

NOTE: Any of the following for only ONE mark:
- Honeypots are used by companies so they can analyse an attack.
- These computers, which appear real to the intruder, actually are separated safely from the company's network.

Accept: A trap set up to allow users unauthorised access. (2)

6.3 6.3.1 (Any TWO valid sectors ✓✓)
- Moviemakers
- Musicians
- Publishers

Accept: Entertainment / software / books (2)

6.3.2 (Any TWO ways to identify real DVD ✓✓)
- The IFPI SID code – it stands for International Federation of the Phonographic Industry Source Identity Code
- The copyright information with the copyright symbol ©
- The studio's symbol which tells you who made or released the original movie/music
- The zone indicator identifying where the DVD is meant to be sold, e.g. South Africa is zone 2
- The hologram insignia

Accept: Purchase from a legitimate retailer
Label must be genuine
There is a clear colour deviation on the back of the disk
The image quality of copies is inferior (2)

6.3.3 (a) Bootlegging is the process of illegally recording a live event / broadcast and then selling this recording without obtaining the copyright holder's permission. ✓
Accept: Examples (1)

(b) (Any valid example ✓)
- Filming a movie while it is being shown in a cinema
- Video-taping a live concert that you attend and then selling the video (1)

6.4 6.4.1 (Any TWO valid ways to trust a web page ✓✓)
- Check the author of web page
- Check who the publisher of the information is
- Check the reliability of the information (2)

6.4.2 (Any TWO ways to verify information on web page ✓✓)
- Check if the data/date is current
- Is there any cross references?
- Is there more than one web page with the same type of information provided?
- Accept all answers from 6.4.1 (2)

TOTAL SECTION E: 43
GRAND TOTAL: 180

NB: Do not mark the following: 6.2.3 and 6.2.4(b)