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:

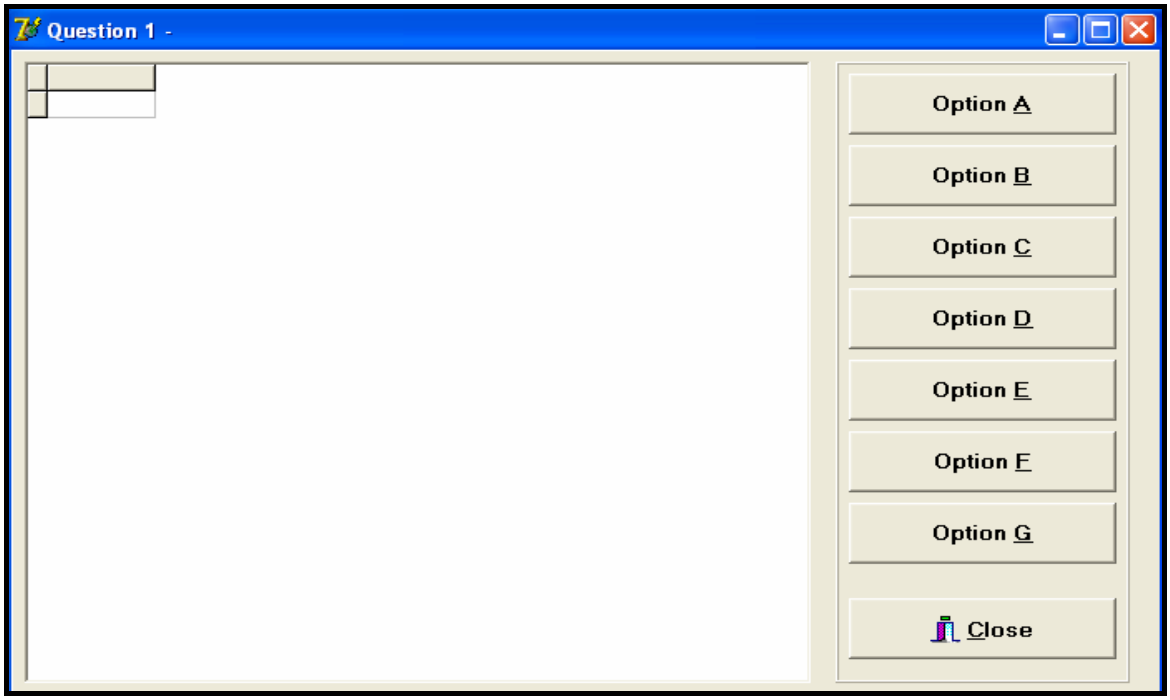
- Rename the folder **Question1_Delphi** as **Question1_X**, where X should be replaced with your examination number.
- Open Delphi and then open the file **Question1_P.dpr** in the folder **Question1_X**. The program displays eight buttons as well as a DBGrid that will be used as an output component (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.

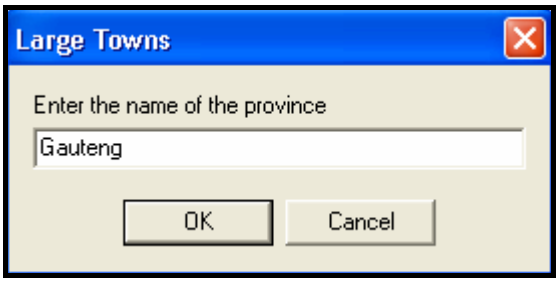
Example of the output for the first seven records:

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.

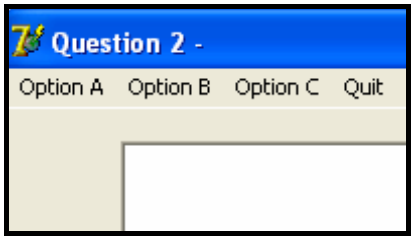
Example of the output when the string returned by the **toString** method is displayed:

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:

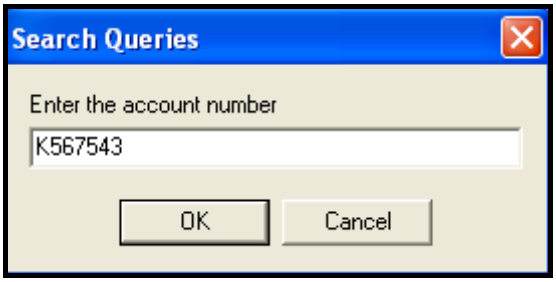
- Allow the user to enter an account number and call the **validateAccNum** subprogram to validate the account number.
- Display a suitable message if the account number is invalid and terminate the search process. Continue the search process if the account number is valid.
- All issues related to the 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.

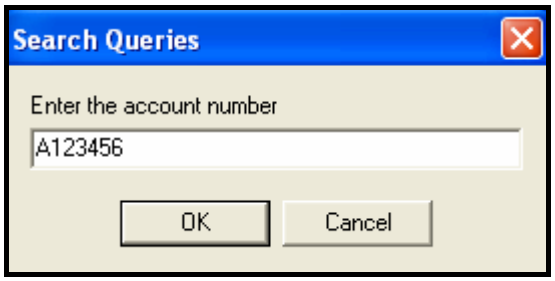
NOTE: You have to run **Option A** if **Option B** is to produce the correct output.

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:

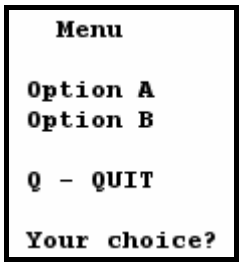
The type of issue:Account number:Date#Contents of the issue

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 **Question 1.**

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

tblTowns Table Structure

tblTowns : Table

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

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 Table Sample Data

tblTowns : Table

TownID	TownName	Province	WaterRestrictions	Population	DamID
1	Willowmore	Eastern Cape	False	7100	6
2	Queenstown	Eastern Cape	False	55800	14
3	East London	Eastern Cape	False	423500	18
4	Kareedouw	Eastern Cape	False	3400	23
5	Tsomo	Eastern Cape	False	10600	107
6	Stutterheim	Eastern Cape	False	20800	162
7	Kirkwood	Eastern Cape	True	10200	29
8	Indwe	Eastern Cape	True	12300	33
9	Uitenhage	Eastern Cape	True	198800	52
10	Hazelmere	Eastern Cape	True	3700	55
11	Humansdorp	Eastern Cape	True	15600	58
12	Cradock	Eastern Cape	True	30200	73
13	Qamata	Eastern Cape	True	6700	88
14	Umtata	Eastern Cape	True	79000	102
15	Graaff Reinet	Eastern Cape	True	35400	111
16	Cofimvaba	Eastern Cape	True	3500	144
17	Whittlesea	Eastern Cape	True	12100	154
18	Kroonstad	Free State	False	90800	9
19	Van Stadensrus	Free State	False	4500	38
20	Theunissen	Free State	False	21300	42

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 own database with the name **DamsDB** that includes a table named **tblDams** and another table named **tblTowns** in the folder called **Question1_Delphi** or **Question1_Java**.
- Change the data types and the sizes of the fields in the two tables according to the specifications given below.

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

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 thousands of litres
Capacity	Number	Long Integer	The total capacity of the dam in thousands of litres
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 **tblTowns** table are defined as 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 imposed on the town
Population	Number	Long Integer	The population of the town
DamID	Number	Integer	The ID of the dam that supplies the town's water

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)*

_____

_____

_____

_____

_____