basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2012

MARKS: 120

TIME: 3 hours

This question paper consists of 31 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:**
FundsDB.mdb	Funds.java
Question1_P.dpr	FundsDB.mdb
Question1_P.res	tblDonations.txt
Question1_U.dfm	tblStalls.txt
Question1_U.pas	TestQuestion1.java
tblDonations.txt	
tblStalls.txt	

   QUESTION 2

Delphi:	**Java:**
Data2011.txt	Data2011.txt
Question2_P.dpr	Event.java
Question2_P.res	TestQuestion2.java
Question2_U.dfm	
Question2_U.pas	
uEvent.pas	

   QUESTION 3

Delphi:	**Java:**
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. The invigilator will inform you whether you should hand in printouts of the programming code of all the questions you have done.

11. If printing is required, all printing of programming questions that you have done will take place within an hour of the completion of this question paper.

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

Fund-raising has become an integral part of an annual school programme. The community, together with the teachers and learners at Sundowns High School, put their efforts together in organising several events throughout the year to ensure that the school has sufficient funds to support the school's academic, sporting and cultural activities.

QUESTION 1: DELPHI PROGRAMMING AND DATABASE

One fundraising event the school decided to have is a curry and rice day. Mr Sweety, the fundraising coordinator at the school, has created a database named **FundsDB** to store the data for the curry and rice day.

Each class is responsible for a stall that will be set up, either on rugby fields A, B or C or on hockey fields A, B or C. Space is allocated to each stall according to the estimated number of guests that will visit the stall. Donations are collected towards the purchasing of some of the requirements (meat, rice, vegetables or other items) that are needed to prepare and serve the curry and rice.

An incomplete program has been developed to process queries on the data in the given database. Your task will be to complete this program.

The database named **FundsDB**, as well as an incomplete Delphi project named **Question1_P.dpr**, has been saved in the folder named **Question1_Delphi**.

NOTE: The design of the tables in the **FundsDB** 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 **FundsDB** 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.

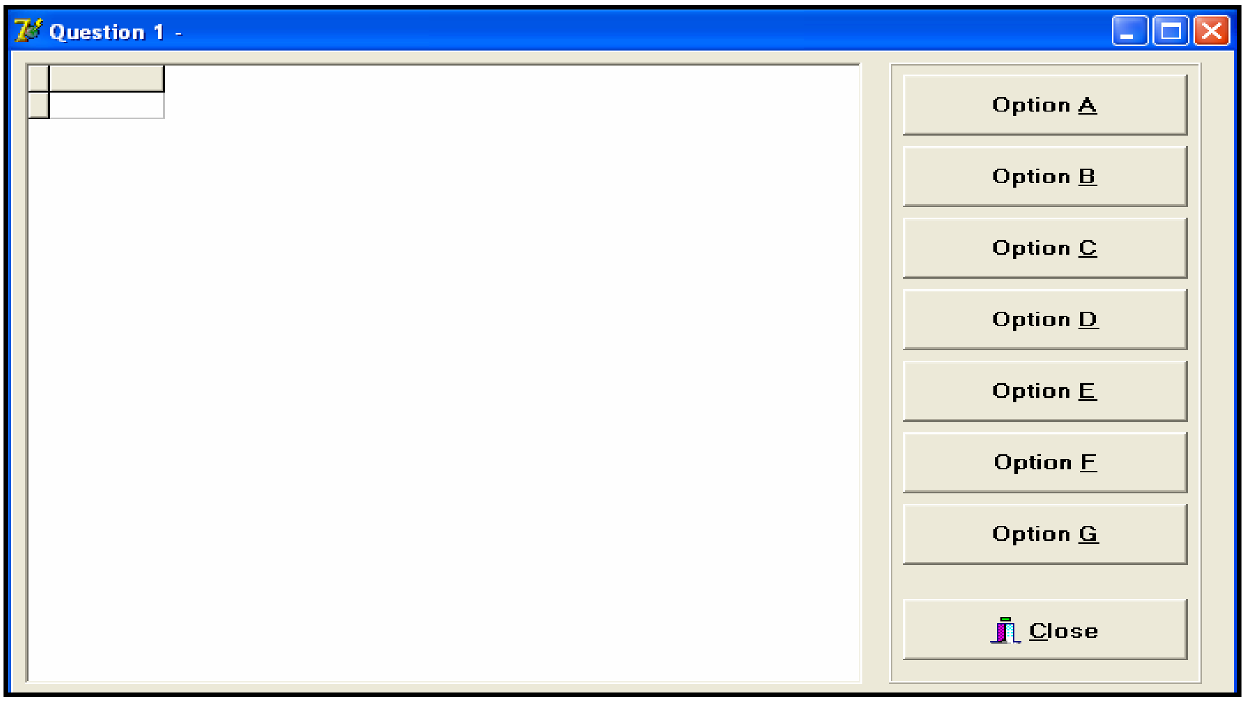
- Go to 'File/Save As …' and save the unit as **Question1_UXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Go to 'File/Save Project As …' and save the project as **Question1_PXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- The program should be able to connect to the database named **FundsDB**. 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 connection with the database.

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

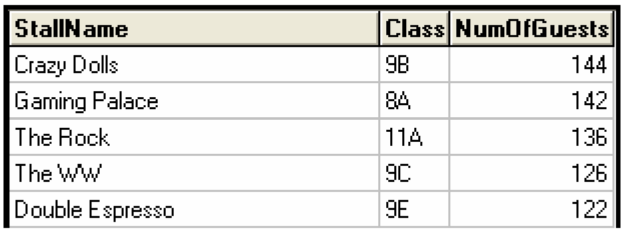


Do the following:

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 Mr Sweety wants a list of the estimated number of guests that will visit the stalls. Complete the code for the **Option A** button by formulating an SQL statement to display the **stall name**, **class** and **number of guests** for all the stalls in the **tblStalls** table. Display the results according to the number of guests in descending order.

Example of the output for the first five records:

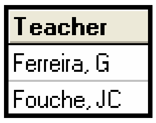


StallName	Class	NumOfGuests
Crazy Dolls	9B	144
Gaming Palace	8A	142
The Rock	11A	136
The WW	9C	126
Double Espresso	9E	122

: (3)

1.2 Miss Shelby, the deputy principal, needs to contact all the teachers who are responsible for the stalls on the C-rugby field with an expected number of 100 or more guests. Complete the code for the **Option B** button by formulating an SQL statement to display the **names of all the teachers** who are responsible for these stalls.

Example of the output:

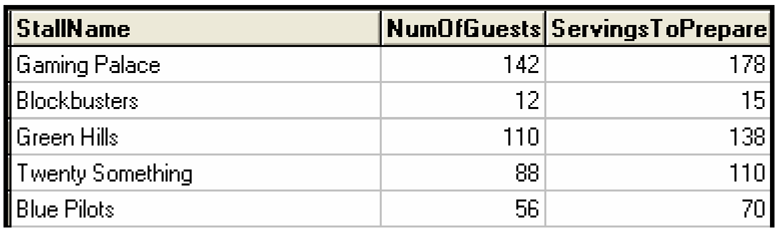


Teacher
Ferreira, G
Fouche, JC

(4)

1.3 Mr Sweety needs to calculate the estimated number of servings to prepare for each stall using a figure of 1,25 servings per expected guest. Complete the code for the **Option C** button by formulating an SQL statement to display the **name of the stall**, the **number of guests** expected and **the number of servings to prepare** for each stall in the **tblStalls** table. The **number of servings to prepare** is a calculated field which needs to be rounded off to the nearest whole number. Use **ServingsToPrepare** as the field name.

Example of the output for the first five records:

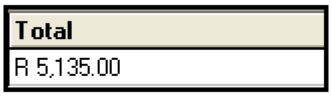


StallName	NumOfGuests	ServingsToPrepare
Gaming Palace	142	178
Blockbusters	12	15
Green Hills	110	138
Twenty Something	88	110
Blue Pilots	56	70

: (5)

1.4 Complete the code for the **Option D** button by formulating an SQL statement to display the total amount of donations promised, but not yet received, for a particular item. Allow the user to enter the name of the item from the keyboard. Use **Total** as the calculated field name.

Example of the output, if **Rice** is the item entered by the user:

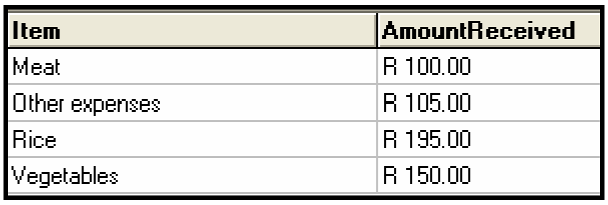


Total
R 5,135.00

(5)

1.5 Mr Du Plessis (the teacher for Grade 12B) wants to know what the total amount received is, in terms of donations, for each of the various items for the Grade 12B class. Complete the code for the **Option E** button by formulating an SQL statement to display the total amount received for each item for the stall operated by Grade 12B. Use **AmountReceived** as the new field name.

Example of the output:

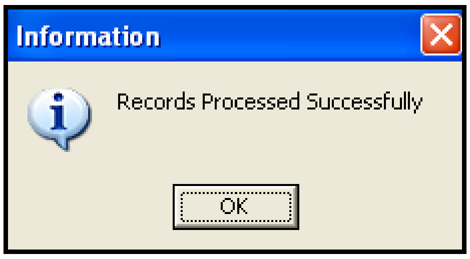


Item	AmountReceived
Meat	R 100.00
Other expenses	R 105.00
Rice	R 195.00
Vegetables	R 150.00

(7)

1.6 Mr Sweety has been informed that the estimated number of guests for all the A-classes (that is Grades 8A, 9A, 10A, 11A and 12A) has increased by 5% according to the ticket sales. Complete the code for the **Option F** button by formulating an SQL statement that will **update** the **tblDonations** table accordingly.

Example of the output:



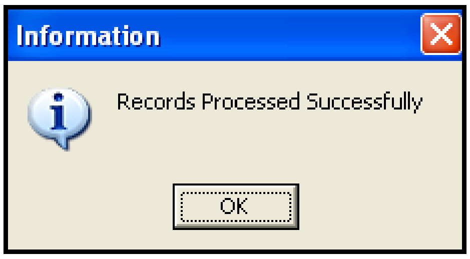
Information

Records Processed Successfully

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

1.7 One of the parents of a Grade 12B learner donated R200,00 towards 'other expenses' to the class. Grade 12B operates the stall with ID number HC77 (**StallID**). Complete the code for the **Option G** button by formulating an SQL statement that will add the donation to the records.

Example of the output:



```
Records Processed Successfully
```

HINT: Select **Option E** to verify that the record has been updated. (4)

- Enter your examination number as a comment in the first line of the file named **Question1_UXXXX.pas** containing the SQL statements.
- Save the unit **Question1_UXXXX** and the project **Question1_PXXXX** (File/Save All).
- A printout of the code for the **Question1_UXXXX.pas** file will possibly be required (see Instruction 10 on page 3).

[33]

QUESTION 2: DELPHI OBJECT-ORIENTED PROGRAMMING

The Grade 11 learners have to raise money to host the Grade 12 farewell function at the end of the year. Software has been partially developed to assist with managing the income and expenses of the different events held to raise funds.

The given program in the **Question2_Delphi** folder consists of a class unit to describe a fundraising-event object and a main unit to create an array of fundraising-event objects. The program uses the array of objects to display specific required information.

The program is designed to cater for storing the name of an event, the name of the person who organised the event, the date when the event took place, as well as the income received and expenses incurred for an event.

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 unit as **Question2_UXXXX** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the unit **uEvent.pas**.
- Go to 'File/Save As …' and save the unit as **uEventXXXX.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).

2.1 The class unit named **uEventXXXX.pas** contains a **TEvent** class that describes a single fundraising event.

Note the following:

- **Use the names of the fields indicated in QUESTION 2.1.1 in this class.**
- None of the fields in this class should be accessible to the main unit (form unit).
- All the methods of this class should be accessible to the main unit (form unit).
- Code should **not be repeated** to solve a given problem. Where possible, the given methods **must** be used to do tasks required to solve problems.
- Some of the code in the given class has been inserted as comments in order to get the unit to compile. You will be required to remove the comment symbols, as indicated in the questions that follow.
- In addition to modifying the given methods, you will be required to add code for new methods, as described in the questions that follow.

You are required to correct and complete the given code in the **uEventXXXX** unit by doing the following:

2.1.1 The constructor receives the following information of an event as parameters:

- **Name**, which refers to the name of the event
- **Person**, which refers to the name of the person who organised the event
- **Date** on which the event took place
- **Income** that was generated
- **Expenses** that had to be incurred

The class variables have been initialised in the constructor using the parameter values. However, the class variables have not been declared yet.

Do the following:

- Declare the class variables of the class using the variables' names, as used in the constructor.
- Remove the comment symbols from the statements initialising the class variables in the constructor.
- Remove the comment symbols from the return statement in the given **toString** method. (5)

2.1.2 Write a method called **calculateProfit** to calculate and return the profit that was made at an event. (3)

2.1.3 Write a method called **findTerm** that will determine and return a value (1, 2, 3 or 0) indicating the term during which the event took place.

The following applies:

- Period of term 1: January, February and March
- Period of term 2: April, May and June
- Period of term 3: July, August and September
- Period of term 4: October, November and December
  No events were hosted during term 4.
- A value of 0 must be returned in the cases where a value other than 1, 2 or 3 was generated. (7)

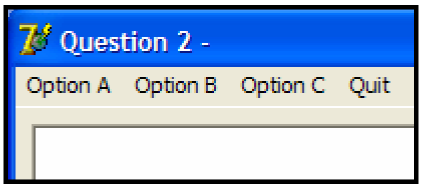
2.1.4 Write a method called **constructNameString** that will construct and return a new string containing the name of the organiser of an event in the following format:

Initials<space>Last name

Example:

If the name of the person is Mr Johnson, John Patrick, the new string must be **JP Johnson**. (10)

2.2 In the **Question2_UXXXX** unit (the main unit) you have been given code to display the following menu when you execute the program:



Open the **Question2_UXXXX** unit (the main unit).

Add your examination number as a comment in the first line of the **Question2_UXXXX** unit.

2.2.1 An array to store a maximum of twenty **TEvent**-type objects has been declared.

A text file named **Data2011.txt** containing information on all the fundraising events that took place during the year has been supplied in the **Question2_X** folder. The information on each event has been saved over three lines in the following format in the text file:

- **First line**: The name of the event followed by a colon (:) and the name of the organiser followed by the date of the event. Note that the name of the organiser and the date of the event are separated by the word 'on'.
- **Second line**: The income as an amount
- **Third line**: The expenses as an amount

Example of the content of the text file:

```
Fun Walk:Mr Jackson,Harold James on 2011/05/12
2500
500
Raffle#1:Mrs Freeman,Jane on 2011/02/24
545
0
Bazaar:Mrs Green,Lee-Ann Suzanne on 2011/04/26
780
300
Raffle#2:Mr Nthumba,Eric on 2011/03/11
375
0
LAN Gaming Competition:Mrs Xaba,Mary Elizabeth on 2011/08/01
2400
600
Raffle#3:Mr Wilson,Jeremy Peter Wayne on 2011/07/25
425
0
Pancake Sale:Mrs Moodley,Harriot Charlene on 2011/06/02
565
150
Battle of the Bands:Mr Anderson,Brandon Westley on 2011/02/18
8900
3000
Talent Competition:Mrs Mohammed,Judith Diane Sheryl on 2011/07/15
9500
2000
```

NOTE: The program must be able to read the information of an unknown number of events from the text file.

Write code in the **OnActivate event handler** of the form to read the information from the text file **Data2011.txt** as follows:

Test if the text file exists. Display a suitable message if the file does not exist and terminate the program.

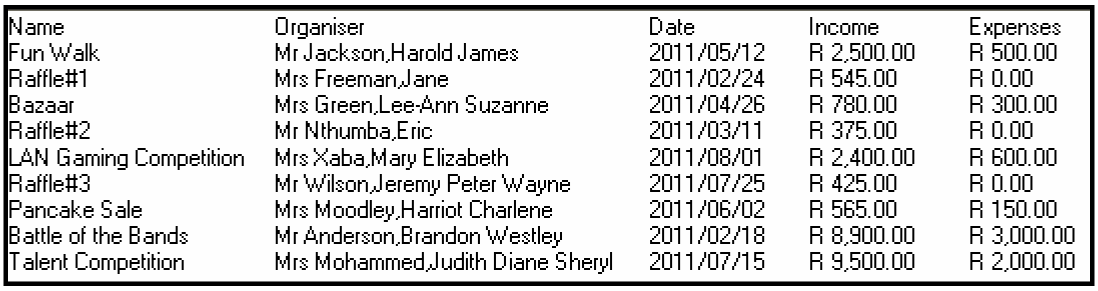
If the file does exist, use a loop to read the information from the text file according to the following steps:

- Read the first line from the file and separate the text into the name of the event, the name of the organiser and the date of the event.
- Read the next two lines from the text file as the income and the expenses of the event.
- Use this information to create a new **TEvent** object and place the object into the array.
- Use a counter to keep track of how many objects have been placed in the array. (10)

2.2.2 **Menu Option A**

When the user clicks on this option, the program must display all the information on all the events stored in the array.

Complete the code to display the output as follows:



```
Name                     Organiser                         Date        Income       Expenses
Fun Walk                 Mr Jackson,Harold James           2011/05/12  R 2,500.00   R 500.00
Raffle#1                 Mrs Freeman,Jane                  2011/02/24  R 545.00     R 0.00
Bazaar                   Mrs Green,Lee-Ann Suzanne         2011/04/26  R 780.00     R 300.00
Raffle#2                 Mr Nthumba,Eric                   2011/03/11  R 375.00     R 0.00
LAN Gaming Competition   Mrs Xaba,Mary Elizabeth           2011/08/01  R 2,400.00   R 600.00
Raffle#3                 Mr Wilson,Jeremy Peter Wayne      2011/07/25  R 425.00     R 0.00
Pancake Sale             Mrs Moodley,Harriot Charlene      2011/06/02  R 565.00     R 150.00
Battle of the Bands      Mr Anderson,Brandon Westley       2011/02/18  R 8,900.00   R 3,000.00
Talent Competition       Mrs Mohammed,Judith Diane Sheryl  2011/07/15  R 9,500.00   R 2,000.00
```

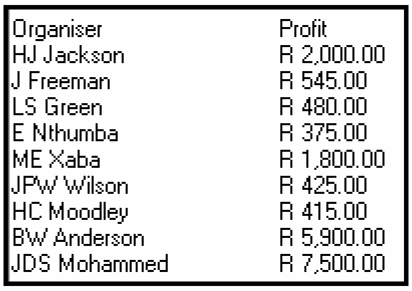
(2)

2.2.3 **Menu Option B**

When the user clicks on this option, the program must display a heading and call the relevant methods to display the name of the organiser and the profit that was made during each event. The name of the organiser must be displayed in the following format:

Initials<space>Last name

Example of the output (on the next page):



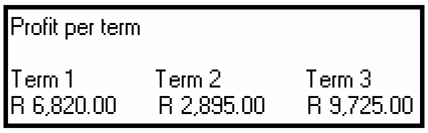
```
Organiser             Profit
HJ Jackson            R 2,000.00
J Freeman             R 545.00
LS Green              R 480.00
E Nthumba             R 375.00
ME Xaba               R 1,800.00
JPW Wilson            R 425.00
HC Moodley            R 415.00
BW Anderson           R 5,900.00
JDS Mohammed          R 7,500.00
```

(4)

2.2.4 **Menu Option C**

When the user clicks on this option, the program must use the **findTerm** and the **calculateProfit** methods to determine and display the total profit that was made during each term.

Example of the output:



```
Profit per term

Term 1        Term 2        Term 3
R 6,820.00    R 2,895.00    R 9,725.00
```

(8)

- 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 class unit **uEventXXXX.pas**.
- Save all the files (File/Save All).
- Printouts of the code for the units **Question2_UXXXX.pas** and **uEventXXXX.pas** may be required (see Instruction 10 on page 3).

[49]

QUESTION 3: DELPHI PROGRAMMING

A treasure hunt is one of the activities used as a fund-raiser during an annual market-day event at the school.

Participants will be required to pay an entry fee and select five different numbers from 1 to 20. Points are awarded for each number selected. Depending on the total number of points a participant has, different prizes are awarded.

Rules of the game:

- The numbers from 1 to 20 are used in the treasure hunt for participants to select from.
- A number of points are hidden behind each number.
- The participant must enter five different numbers in the range of 1 to 20.
- The hidden number of points awarded to each number that was entered will be added together to determine the total number of points for the participant.
- The participant is not allowed to enter the same number more than once. Every time a previously entered number is selected again, five points will be deducted from the total number of points for the participant.
- Every time a number outside the range of 1 to 20 is entered, five points will be deducted from the total number of points for the participant.
- A prize will be awarded to the participant depending on the total number of points that was scored.

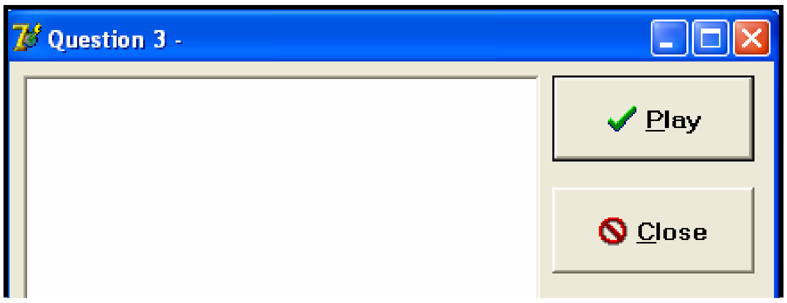
NOTE: This is a problem-solving question. The following applies:

- You have to develop your own solution according to the specifications in the question paper.
- Good programming principles should be followed, for example descriptive variable names, indentation, et cetera.
- A modular programming approach should be followed. A maximum of **four marks** could be deducted if your solution is not modular, that is it has no subprograms with parameter passing.
- The program only has to work for ONE participant.

You have been supplied with an incomplete program in the **Question3_ Delphi** folder.

Do the following:

- Rename the folder named **Question3_Delphi** as **Question3_X** (where X should be replaced with your examination number).
- Open Delphi and then open the file called **Question3_P.dpr** in the folder **Question3_X**.
- Go to 'File/Save As …' and save the unit as **Question3_UXXXX** (where XXXX should be replaced with the last FOUR digits of your examination number).
- Go to 'File/Save Project As …' and save the project as **Question3_PXXXX** (where XXXX should be replaced with the last FOUR digits of your examination number).
- Add your examination number to the right of the caption of the form.
- Execute the program. The following interface will be displayed (on the next page):



:

A list of numbers from 1 to 20 and their corresponding points are stored in an array called **arrPoints**.

NOTE: The 20 elements in the array are not in numeric order.

Example of the declaration of the array **arrPoints**:

```
arrPoints : Array[1..20] of string =
     ('12:40','20:0','13:0','3:0','15:0','9:0', '19:50','10:0',
      '8:90','11:0','1:0', '5:30','16:0','14:100','4:0','17:0',
      '18:20','6:0','7:0','2:20');
```

The format of each entry in the array is as follows:

'12:40': **12** refers to the number (that the participant will select) and **40** refers to the number of points hidden behind number 12.

'20:0': **20** refers to the number (that the participant will select) and **0** refers to the number of points hidden behind number 20, et cetera.

Complete the code for the **Play** button to do the following:

3.1 Allow the participant to enter five numbers.

Create an array to store the entries entered by the user.

The following must be done for each number entered:

- Add the corresponding points for the number entered to the participant's total.
- Delete the entry from the given array. Example: If 8 is entered, the entry '8:90' must be deleted from array arrPoints.
- Store the entry in a new array in the same format as it is in the given array. Example: If 8 is entered, the entry '8:90' must be stored in the new array.
- If a number is entered more than once:
  - o Five points must be deducted from the participant's total points each time.
  - o The entry is stored as 'number ALREADY SELECTED'.
    Example: If 8 is entered a second time, the entry in the new array will be '8 ALREADY SELECTED'.

- If a number outside the range of 1 to 20 is entered:
  - o Five points must be deducted from the participant's total points each time.
  - o The entry is stored as 'number INVALID NUMBER' in the new array. Example: If 21 is entered, the entry in the new array will be '21 INVALID NUMBER'. (26)

3.2 Display the following:

- The remaining entries in the treasure hunt after the participant's choices
- The participant's choices, the points achieved and the corresponding prize

The prize awarded to the participant will depend on the number of total points scored.

Total Points	**Prize**
<= 0	No Prize
1–100	Teddy Bear
101–200	Fishing Rod
> 200	Gym Membership

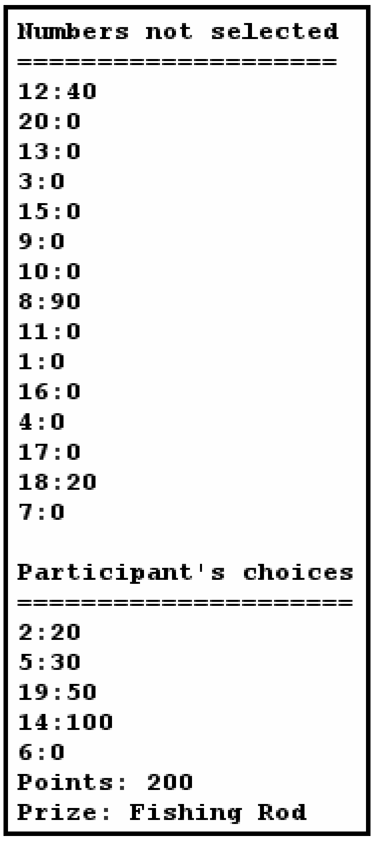
(12)

Example test runs:

Example 1:
Input:

The following numbers are entered one by one: **2**; **5**; **19**; **14**; **6**.

Output (on the next page):

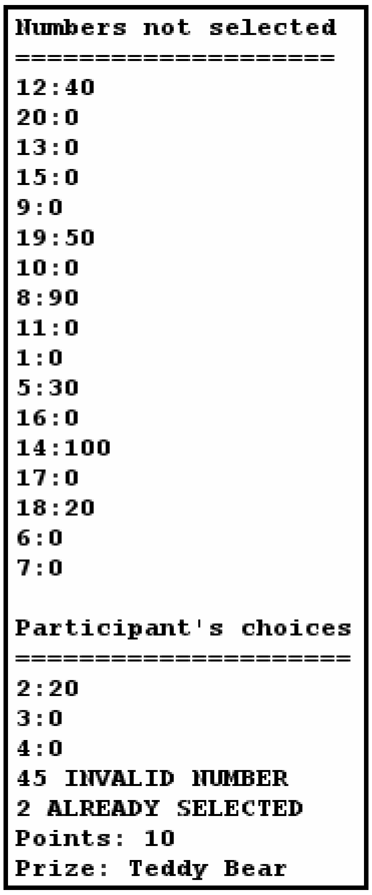


```
Numbers not selected
====================
12:40
20:0
13:0
3:0
15:0
9:0
10:0
8:90
11:0
1:0
16:0
4:0
17:0
18:20
7:0

Participant's choices
=====================
2:20
5:30
19:50
14:100
6:0
Points: 200
Prize: Fishing Rod
```

Example 2:
Input:

The following numbers are entered one by one: **2**; **3**; **4**; **45**; **2**.

Output (on the next page):



```
Numbers not selected
====================
12:40
20:0
13:0
15:0
9:0
19:50
10:0
8:90
11:0
1:0
5:30
16:0
14:100
17:0
18:20
6:0
7:0

Participant's choices
=====================
2:20
3:0
4:0
45 INVALID NUMBER
2 ALREADY SELECTED
Points: 10
Prize: Teddy Bear
```

- Make sure your examination number is entered as a comment in the first line of the main unit **Question3_UXXXX**.
- Save all the files (File/Save All).
- A printout of the code for the unit **Question3_UXXXX** may be required (see Instruction 10 on page 3).

[38]

TOTAL SECTION A: 120

SECTION B

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

SCENARIO

Fund-raising has become an integral part of an annual school programme. The community, together with the teachers and learners at Sundowns High School, put their efforts together in organising several events throughout the year to ensure that the school has sufficient funds to support the school's academic, sporting and cultural activities.

QUESTION 1: JAVA PROGRAMMING AND DATABASE

One fundraising event the school decided to have is a curry and rice day. Mr Sweety, the fundraising coordinator at the school, has created a database named **FundsDB** to store the data for the curry and rice day.

Each class is responsible for a stall that will be set up, either on rugby fields A, B or C or on hockey fields A, B or C. Space is allocated to each stall according to the estimated number of guests that will visit the stall. Donations are collected towards the purchasing of some of the requirements (meat, rice, vegetables or other items) that are needed to prepare and serve the curry and rice.

An incomplete program has been developed to process queries on the data in the given database. Your task will be to complete this program.

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

NOTE: The design of the tables in the **FundsDB** 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 **FundsDB** 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.
- Open the incomplete program **TestQuestion1.java** in the **Question1_X** folder.
- Change the name of the class to **TestQuestion1XXXX** (where XXXX should be replaced with the last FOUR digits of your examination number). Save the file.
- Save the class as **TestQuestion1XXXX.java** (where XXXX should be replaced with the last FOUR digits of your examination number).

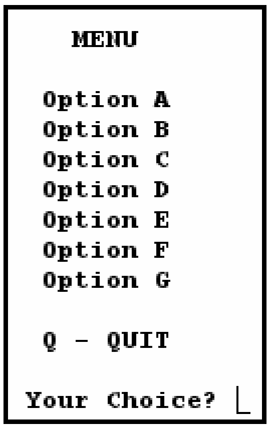
The connectivity code as well as the code to display the results has already been written as part of the given code in the file named **Funds.java**.

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



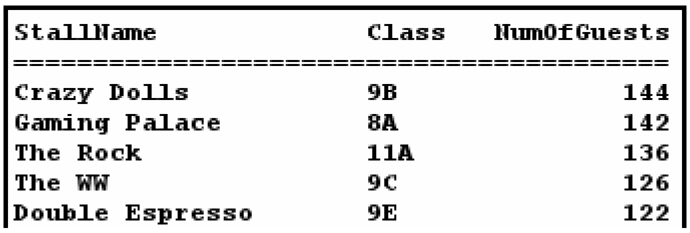
```
   MENU

  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 methods in the **Funds.java** file has been given to you. You need only complete the SQL statements and some input statements as required in the **TestQuestion1XXXX.java** file.

1.1 Mr Sweety wants a list of the estimated number of guests that will visit the stalls. Complete the code for **Option A** by formulating an SQL statement to display the **stall name**, **class** and **number of guests** for all the stalls in the **tblStalls** table. Display the results according to the number of guests in descending order.

Example of the output for the first five records:

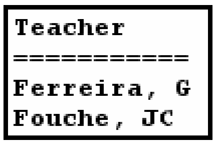


```
StallName              Class   NumOfGuests
==========================================
Crazy Dolls            9B              144
Gaming Palace          8A              142
The Rock               11A             136
The WW                 9C              126
Double Espresso        9E              122
```

: (3)

1.2 Miss Shelby, the deputy principal, needs to contact all the teachers who are responsible for the stalls on the C-rugby field with an expected number of 100 or more guests. Complete the code for **Option B** by formulating an SQL statement to display the **names of all the teachers** who are responsible for these stalls.

Example of the output:

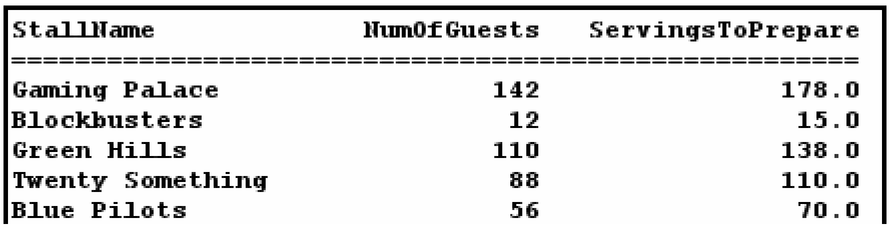


```
Teacher
===========
Ferreira, G
Fouche, JC
```

(4)

1.3 Mr Sweety needs to calculate the estimated number of servings to prepare for each stall using a figure of 1,25 servings per expected guest. Complete the code for **Option C** by formulating an SQL statement to display the **name of the stall**, the **number of guests** expected and **the number of servings to prepare** for each stall in the **tblStalls** table. The **number of servings to prepare** is a calculated field which needs to be rounded off to the nearest whole number. Use **ServingsToPrepare** as the field name.

Example of the output for the first five records:



```
StallName              NumOfGuests   ServingsToPrepare
======================================================
Gaming Palace                  142               178.0
Blockbusters                    12                15.0
Green Hills                    110               138.0
Twenty Something                88               110.0
Blue Pilots                     56                70.0
```

: (5)

1.4 Complete the code for **Option D** by formulating an SQL statement to display the total amount of donations promised, but not yet received, for a particular item. Allow the user to enter the name of the item from the keyboard. Use **Total** as the calculated field name.

Example of the output, if **Rice** is the item entered by the user:

```
Total
=======
R 5135.00
```

(5)

1.5 Mr Du Plessis (the teacher for Grade 12B) wants to know what the total amount received is, in terms of donations, for each of the various items for the Grade 12B class. Complete the code for **Option E** by formulating an SQL statement to display the total amount received for each item for the stall operated by Grade 12B. Use **AmountReceived** as the new field name.

Example of the output:

```
Item              AmountReceived
================================
Meat              R 100.00
Other expenses    R 105.00
Rice              R 195.00
Vegetables        R 150.00
```

(7)

1.6 Mr Sweety has been informed that the estimated number of guests for all the A-classes (that is Grades 8A, 9A, 10A, 11A and 12A) has increased by 5% according to the ticket sales. Complete the code for **Option F** by formulating an SQL statement that will **update** the **tblDonations** table accordingly.

Example of the output:

```
Records Processed Successfully
```

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

1.7 One of the parents of a Grade 12B learner donated R200,00 towards 'other expenses' to the class. Grade 12B operates the stall with ID number HC77 (**StallID**). Complete the code for **Option G** by formulating an SQL statement that will add the donation to the records.

Example of the output:

```
Records Processed Successfully
```

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

- 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 of the code for the **TestQuestion1XXXX.java** file will possibly be required (see Instruction 10 on page 3).

[33]

QUESTION 2: JAVA OBJECT-ORIENTED PROGRAMMING

The Grade 11 learners have to raise money to host the Grade 12 farewell function at the end of the year. Software has been partially developed to assist with managing the income and expenses of the different events held to raise funds.

The given program in the **Question2_Java** folder consists of an object class to describe a fundraising-event object and a test class to create an array of fundraising-event objects. The program uses the array of objects to display specific required information.

The program is designed to cater for storing the name of an event, the name of the person who organised the event, the date when the event took place, as well as the income received and expenses incurred for an event.

Do the following:

- Rename the folder **Question2_Java** as **Question2_X** (where X must be replaced with your examination number).
- Rename the **Event.java** file in the folder **Question2_X** as **EventXXXX.java** (where XXXX must be replaced with the last FOUR digits of your examination number).
- Open the **EventXXXX.java** file.
- Change the **class name** and the name of the **constructor method** to **EventXXXX** (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 **EventXXXX.java** class. Save the file.
- Rename the **TestQuestion2.java** file in the folder **Question2_X** as **TestQuestion2XXXX.java** (where XXXX must be replaced with the last FOUR digits of your examination number).
- 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.

2.1 The object class named **EventXXXX.java** will describe a single fundraising event.

Note the following:

- **Use the names of the fields indicated in QUESTION 2.1.1 in this class.**
- None of the fields in this class should be accessible to the test (application) class.
- All the methods of this class should be accessible to the test (application) class.
- Code should **not be repeated** to solve a given problem. Where possible, the given methods **must** be used to do tasks required to solve problems.
- Some of the code in the given class has been inserted as comments in order to get the class to compile. You will be required to remove the comment symbols, as indicated in the questions that follow.

- In addition to modifying the given methods, you will be required to add code for new methods, as described in the questions that follow.

You are required to correct and complete the given code in the **EventXXXX.java** class by doing the following:

2.1.1 The constructor receives the following information of an event as parameters:

- **Name**, which refers to the name of the event
- **Person**, which refers to the name of the person who organised the event
- **Date** on which the event took place
- **Income** that was generated
- **Expenses** that had to be incurred

The class variables have been initialised in the constructor using the parameter values. However, the class variables have not been declared yet.

Do the following:

- Declare the class variables of the class using the variables' names, as used in the constructor.
- Remove the comment symbols from the statements initialising the class variables in the constructor.
- Remove the comment symbols from the return statement in the given **toString()** method. (5)

2.1.2 Write a method called **calculateProfit()** to calculate and return the profit that was made at an event. (3)

2.1.3 Write a method called **findTerm()** that will determine and return a value (1, 2, 3 or 0) indicating the term during which the event took place.

The following applies:

- Period of term 1: January, February and March
- Period of term 2: April, May and June
- Period of term 3: July, August and September
- Period of term 4: October, November and December
  No events were hosted during term 4.
- A value of 0 must be returned in the cases where a value other than 1, 2 or 3 was generated. (7)

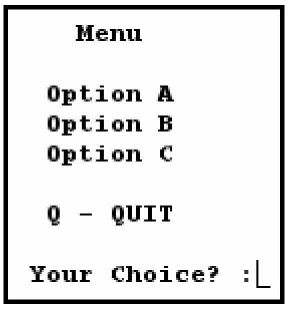
2.1.4 Write a method called **constructNameString()** that will construct and return a new string containing the name of the organiser of an event in the following format:

Initials<space>Last name

Example:

If the name of the person is Mr Johnson, John Patrick, the new string must be **JP Johnson**. (10)

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



```
   Menu

 Option A
 Option B
 Option C

 Q - QUIT

Your Choice? :
```

Open the **TestQuestion2XXXX.java** file (the test class).

Add your examination number as a comment in the first line of the **TestQuestion2XXXX.java** class.

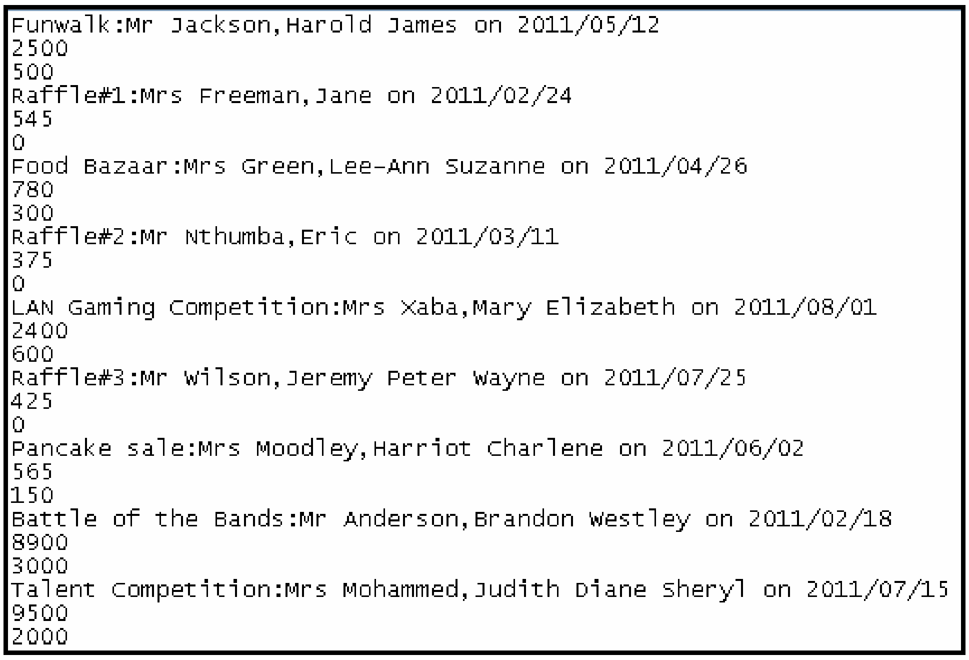
2.2.1 An array to save a maximum of twenty **Event**-type object has been declared.

NOTE: Replace XXXX in the declaration statement of the array with the last four digits of your examination number.

A text file named **Data2011.txt** containing information on all the fundraising events that took place during the year has been supplied in the **Question2_X** folder. The information on each event has been saved over three lines in the following format in the text file:

- **First line**: The name of the event followed by a colon (:) and the name of the organiser followed by the date of the event. Note that the name of the organiser and the date of the event are separated by the word 'on'.
- **Second line**: The income as an amount
- **Third line**: The expenses as an amount

Example of the content of the text file (on the next page):



```
Funwalk:Mr Jackson,Harold James on 2011/05/12
2500
500
Raffle#1:Mrs Freeman,Jane on 2011/02/24
545
0
Food Bazaar:Mrs Green,Lee-Ann Suzanne on 2011/04/26
780
300
Raffle#2:Mr Nthumba,Eric on 2011/03/11
375
0
LAN Gaming Competition:Mrs Xaba,Mary Elizabeth on 2011/08/01
2400
600
Raffle#3:Mr Wilson,Jeremy Peter Wayne on 2011/07/25
425
0
Pancake sale:Mrs Moodley,Harriot Charlene on 2011/06/02
565
150
Battle of the Bands:Mr Anderson,Brandon Westley on 2011/02/18
8900
3000
Talent Competition:Mrs Mohammed,Judith Diane Sheryl on 2011/07/15
9500
2000
```

NOTE: The program must be able to read the information of an unknown number of events from the text file.

Write code to read the information from the text file **Data2011.txt** as follows:

Test if the text file exists. Display a suitable message if the file does not exist and terminate the program.

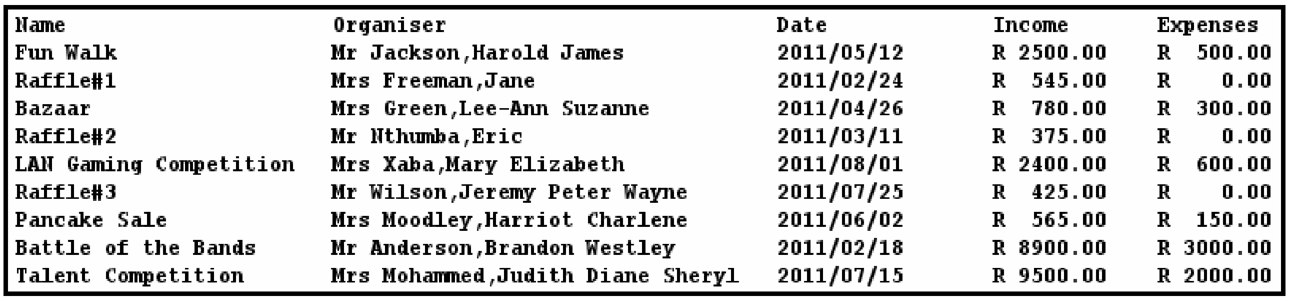
If the file does exist, use a loop to read the information from the text file according to the following steps:

- Read the first line from the file and separate the text into the name of the event, the name of the organiser and the date of the event.
- Read the next two lines from the text file as the income and the expenses of the event.
- Use this information to create a new **EventXXXX** object and place the object into the array.
- Use a counter to keep track of how many objects have been placed in the array. (10)

2.2.2 **Menu Option A**

When the user selects this menu option, the program must display all the information on all the events stored in the array.

Complete the code to display the output as follows (on the next page):



```
Name                     Organiser                          Date          Income      Expenses
Fun Walk                 Mr Jackson,Harold James            2011/05/12    R 2500.00   R  500.00
Raffle#1                 Mrs Freeman,Jane                   2011/02/24    R  545.00   R    0.00
Bazaar                   Mrs Green,Lee-Ann Suzanne          2011/04/26    R  780.00   R  300.00
Raffle#2                 Mr Nthumba,Eric                    2011/03/11    R  375.00   R    0.00
LAN Gaming Competition   Mrs Xaba,Mary Elizabeth            2011/08/01    R 2400.00   R  600.00
Raffle#3                 Mr Wilson,Jeremy Peter Wayne       2011/07/25    R  425.00   R    0.00
Pancake Sale             Mrs Moodley,Harriot Charlene       2011/06/02    R  565.00   R  150.00
Battle of the Bands      Mr Anderson,Brandon Westley        2011/02/18    R 8900.00   R 3000.00
Talent Competition       Mrs Mohammed,Judith Diane Sheryl   2011/07/15    R 9500.00   R 2000.00
```

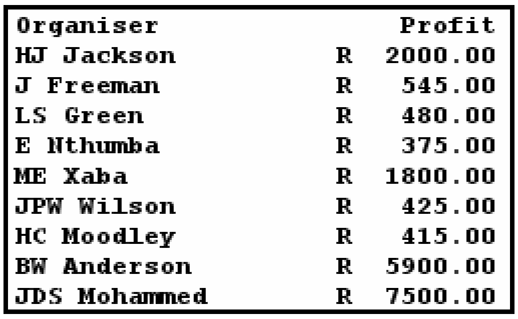
(2)

2.2.3 **Menu Option B**

When the user selects this menu option, the program must display a heading and call the relevant methods to display the name of the organiser and the profit that was made during each event. The name of the organiser must be displayed in the following format:

Initials<space>Last name

Example of the output:



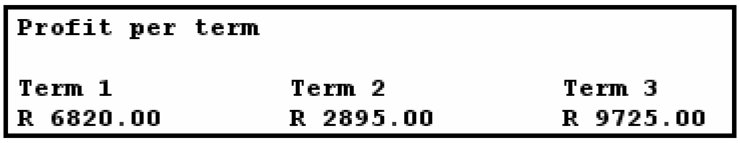
```
Organiser                Profit
HJ Jackson           R  2000.00
J Freeman            R   545.00
LS Green             R   480.00
E Nthumba            R   375.00
ME Xaba              R  1800.00
JPW Wilson           R   425.00
HC Moodley           R   415.00
BW Anderson          R  5900.00
JDS Mohammed         R  7500.00
```

(4)

2.2.4 **Menu Option C**

When the user selects this menu option, the program must use the **findTerm()** and the **calculateProfit()** methods to determine and display the total profit that was made during each term.

Example of the output:



```
Profit per term

Term 1           Term 2           Term 3
R 6820.00        R 2895.00        R 9725.00
```

(8)

- Make sure your examination number is entered as a comment in the first line of the test class **TestQuestion2XXXX.java**, as well as the object class **EventXXXX.java**.
- Save all the files (File/Save All).
- Printouts of the code for the classes **TestQuestion2XXXX.java** and **EventXXXX.java** will possibly be required (see Instruction 10 on page 3).

[49]

QUESTION 3: JAVA PROGRAMMING

A treasure hunt is one of the activities used as a fund-raiser during an annual market-day event at the school.

Participants will be required to pay an entry fee and select five different numbers from 1 to 20. Points are awarded for each number selected. Depending on the total number of points a participant has, different prizes are awarded.

Rules of the game:

- The numbers from 1 to 20 are used in the treasure hunt for participants to select from.
- A number of points are hidden behind each number.
- The participant must enter five different numbers in the range of 1 to 20.
- The hidden number of points awarded to each number that was entered will be added together to determine the total number of points for the participant.
- The participant is not allowed to enter the same number more than once. Every time a previously entered number is selected again, five points will be deducted from the total number of points for the participant.
- Every time a number outside the range of 1 to 20 is entered, five points will be deducted from the total number of points for the participant.
- A prize will be awarded to the participant depending on the total number of points that was scored.

NOTE: This is a problem-solving question. The following applies:

- You have to develop your own solution according to the specifications in the question paper.
- Good programming principles should be followed, for example descriptive variable names, indentation, et cetera.
- A modular programming approach should be followed. A maximum of **four marks** could be deducted if your solution is not modular, that is it has no subprograms with parameter passing.
- You may use one or more classes for this solution, but the program only has to work for ONE participant.

You have been given an incomplete program in the **Question3_Java** folder.

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**.
- Change the class name to **TestQuestion3XXXX** (where XXXX should be replaced with the last FOUR digits of your examination number).
- Add your examination number as a comment in the first line of the program.

A list of numbers from 1 to 20 and their corresponding points are stored in an array called **arrPoints**.

NOTE: The 20 elements in the array are not in numeric order.

Example of the declaration of the array **arrPoints**:

```
String [] arrPoints = {"12:40","20:0","13:0","3:0","15:0","9:0",
                       "19:50","10:0","8:90","11:0","1:0","5:30",
                       "16:0","14:100","4:0","17:0","18:20","6:0",
                       "7:0","2:20"};
```

The format of each entry in the array is as follows:

'12:40': **12** refers to the number (that the participant will select) and **40** refers to the number of points hidden behind number 12.

'20:0': **20** refers to the number (that the participant will select) and **0** refers to the number of points hidden behind number 20, et cetera.

Complete the code to do the following:

3.1 Allow the participant to enter five numbers.

Create an array to store the entries entered by the user.

The following must be done for each number entered:

- Add the corresponding points for the number entered to the participant's total.
- Delete the entry from the given array. Example: If 8 is entered, the entry '8:90' must be deleted from array **arrPoints**.
- Store the entry in a **new array** in the same format as it is in the given array. Example: If 8 is entered, the entry '8:90' must be stored in the new array.
- If a number is entered more than once:
  - o Five points must be deducted from the participant's total points each time.
  - o The entry is stored as 'number ALREADY SELECTED'. Example: If 8 is entered a second time, the entry in the new array will be '8 ALREADY SELECTED'.
- If a number outside the range of 1 to 20 is entered:
  - o Five points must be deducted from the participant's total points each time.
  - o The entry is stored as 'number INVALID NUMBER' in the new array. Example: If 21 is entered, the entry in the new array will be '21 INVALID NUMBER'. (26)

3.2 Display the following:

- The remaining entries in the treasure hunt after the participant's choices
- The participant's choices, the points achieved and the corresponding prize

The prize awarded to the participant will depend on the number of total points scored.

Total Points	Prize
<= 0	No Prize
1–100	Teddy Bear
101–200	Fishing Rod
> 200	Gym Membership

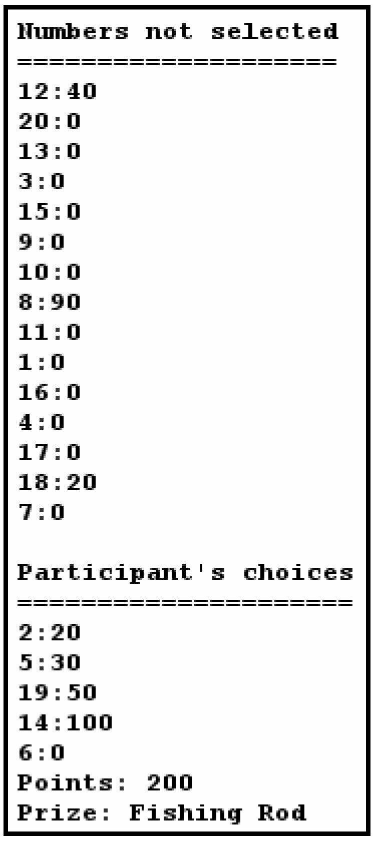
(12)

Example test runs:

Example 1:
Input:

The following numbers are entered one by one: **2**; **5**; **19**; **14**; **6**.

Output:

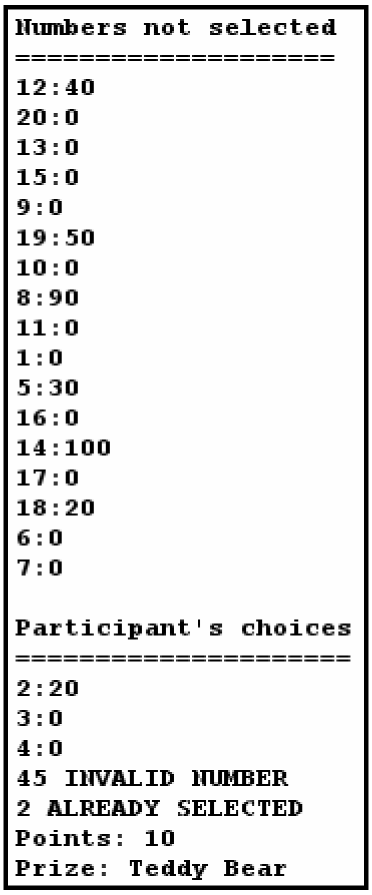


```
Numbers not selected
====================
12:40
20:0
13:0
3:0
15:0
9:0
10:0
8:90
11:0
1:0
16:0
4:0
17:0
18:20
7:0

Participant's choices
=====================
2:20
5:30
19:50
14:100
6:0
Points: 200
Prize: Fishing Rod
```

Example 2:
Input:

The following numbers are entered one by one: **2**; **3**; **4**; **45**; **2**.

Output:



```
Numbers not selected
====================
12:40
20:0
13:0
15:0
9:0
19:50
10:0
8:90
11:0
1:0
5:30
16:0
14:100
17:0
18:20
6:0
7:0

Participant's choices
=====================
2:20
3:0
4:0
45 INVALID NUMBER
2 ALREADY SELECTED
Points: 10
Prize: Teddy Bear
```

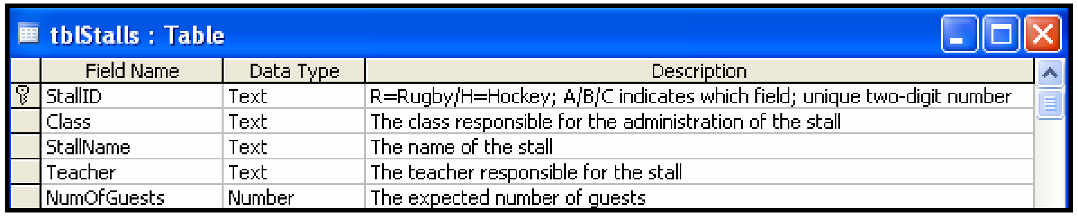
- Make sure your examination number is entered as a comment in the first line of the class **TestQuestion3XXXX.java**, as well as any other class(es) you have created with code.
- Save the class(es).
- A printout of the code for the class **TestQuestion3XXXX.java**, as well as any other class(es) you have created, will possibly be required (see Instruction 10 on page 3). **[38]**

TOTAL SECTION B: 120
GRAND TOTAL: 120

ADDENDUM A: Table Description Sheet

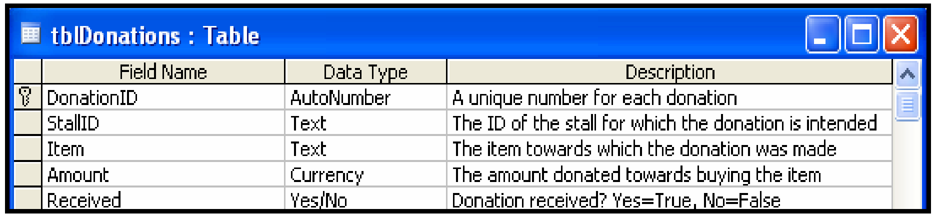
This sheet shows the data structure and sample data for the tables used in the **FundsDB** database in **Question 1.**

tblStalls Table Structure


tblStalls : Table

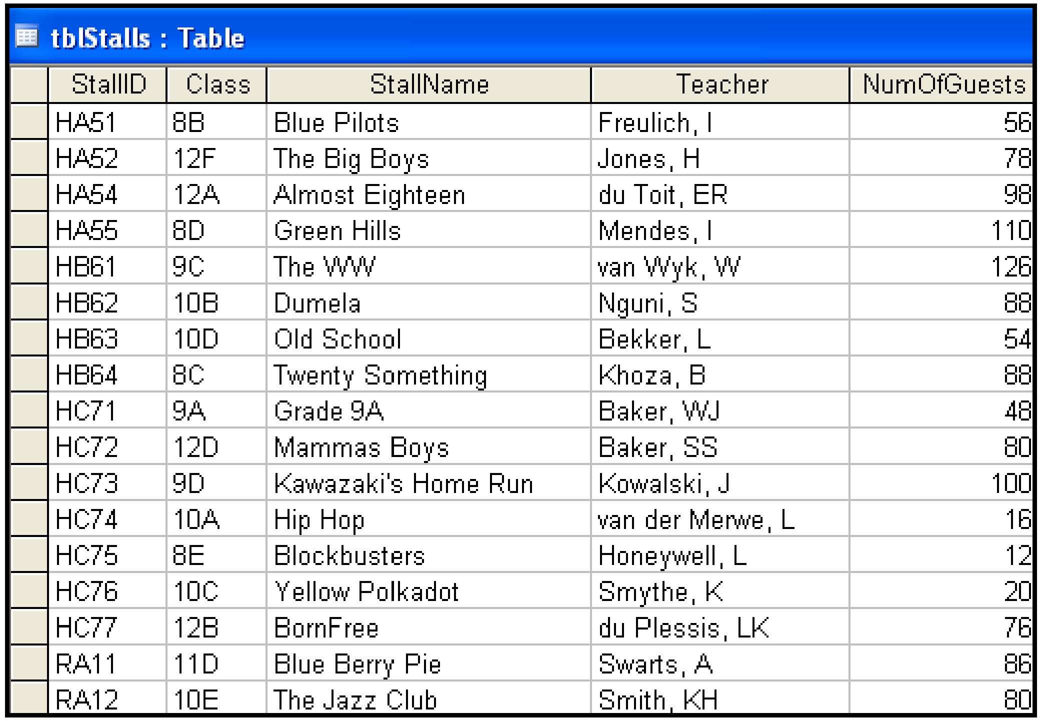
Field Name	Data Type	Description
StallID	Text	R=Rugby/H=Hockey; A/B/C indicates which field; unique two-digit number
Class	Text	The class responsible for the administration of the stall
StallName	Text	The name of the stall
Teacher	Text	The teacher responsible for the stall
NumOfGuests	Number	The expected number of guests

tblDonations Table Structure


tblDonations : Table

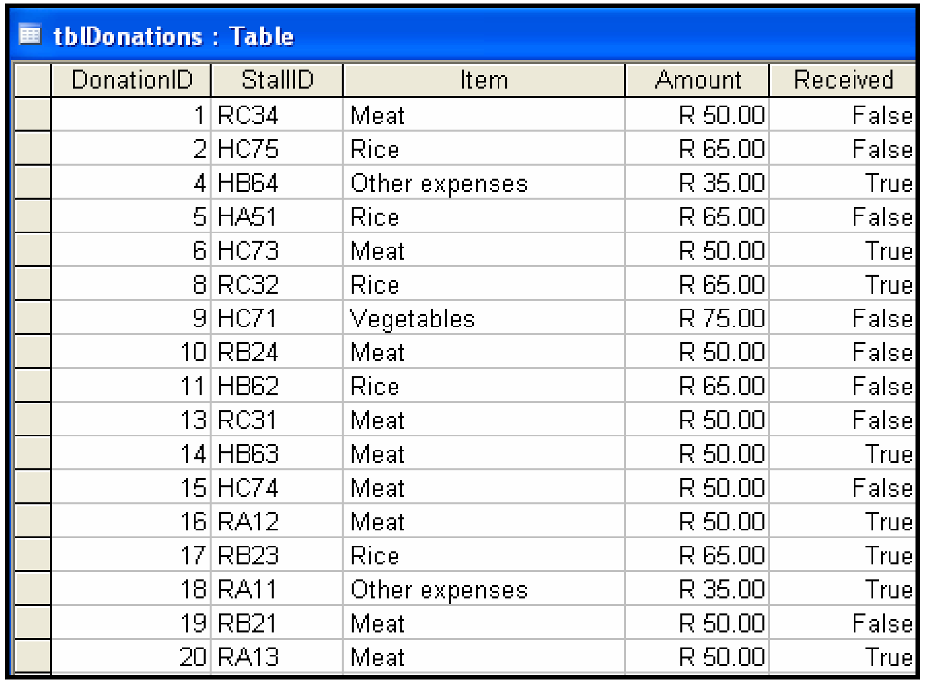
Field Name	Data Type	Description
DonationID	AutoNumber	A unique number for each donation
StallID	Text	The ID of the stall for which the donation is intended
Item	Text	The item towards which the donation was made
Amount	Currency	The amount donated towards buying the item
Received	Yes/No	Donation received? Yes=True, No=False

tblStalls Table Sample Data


tblStalls : Table

StallID	Class	StallName	Teacher	NumOfGuests
HA51	8B	Blue Pilots	Freulich, I	56
HA52	12F	The Big Boys	Jones, H	78
HA54	12A	Almost Eighteen	du Toit, ER	98
HA55	8D	Green Hills	Mendes, I	110
HB61	9C	The WW	van Wyk, W	126
HB62	10B	Dumela	Nguni, S	88
HB63	10D	Old School	Bekker, L	54
HB64	8C	Twenty Something	Khoza, B	88
HC71	9A	Grade 9A	Baker, WJ	48
HC72	12D	Mammas Boys	Baker, SS	80
HC73	9D	Kawazaki's Home Run	Kowalski, J	100
HC74	10A	Hip Hop	van der Merwe, L	16
HC75	8E	Blockbusters	Honeywell, L	12
HC76	10C	Yellow Polkadot	Smythe, K	20
HC77	12B	BornFree	du Plessis, LK	76
RA11	11D	Blue Berry Pie	Swarts, A	86
RA12	10E	The Jazz Club	Smith, KH	80

tblDonations Table Sample Data


tblDonations : Table

DonationID	StallID	Item	Amount	Received
1	RC34	Meat	R 50.00	False
2	HC75	Rice	R 65.00	False
4	HB64	Other expenses	R 35.00	True
5	HA51	Rice	R 65.00	False
6	HC73	Meat	R 50.00	True
8	RC32	Rice	R 65.00	True
9	HC71	Vegetables	R 75.00	False
10	RB24	Meat	R 50.00	False
11	HB62	Rice	R 65.00	False
13	RC31	Meat	R 50.00	False
14	HB63	Meat	R 50.00	True
15	HC74	Meat	R 50.00	False
16	RA12	Meat	R 50.00	True
17	RB23	Rice	R 65.00	True
18	RA11	Other expenses	R 35.00	True
19	RB21	Meat	R 50.00	False
20	RA13	Meat	R 50.00	True

ADDENDUM B: Instructions to create the database FundsDB.mdb

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

- Use the two text files named **tblStalls** and **tblDonations** that have been supplied. Create your own database with the name **FundsDB** that includes a table named **tblStalls** and another table named **tblDonations** 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 **tblStalls** table stores data on the stalls at the event. The fields in the **tblStalls** table are defined as follows:

Field Name	**Type**	**Size**	**Description**
StallID	Text	5	R=Rugby/H=Hockey; A/B/C indicates which field; unique two-digit number
Class	Text	3	The class responsible for the administration of the stall
StallName	Text	30	The name of the stall
Teacher	Text	30	The teacher responsible for the stall
NumOfGuests	Number	Integer	The expected number of guests

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

The **tblDonations** table stores data on the donations. The fields in the **tblDonations** table are defined as follows:

Field Name	**Type**	**Size**	**Description**
DonationID	AutoNumber	LongInt	A unique number for each donation
StallID	Text	5	The ID of the stall for which the donation is intended
Item	Text	30	The item towards which the donation was made
Amount	Currency	Double	The amount donated towards buying the item
Received	Yes/No	True/False	Donation received? Yes=True, No=False

See ADDENDUM A for an example of the data in the **tblDonations** 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 **qryQOne**.
- 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 **FundsDB** file. You will find this file in the **Question1_Delphi** folder. Once you have found it, select the **FundsDB** 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.

INFORMATION TECHNOLOGY P1

NOVEMBER 2011 (1)

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

FOLDER NAME ____________________

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

Question number	**Saved (*tick* ✓)**	**Maximum mark**	**Mark achieved**	**Marker initial/code**
1		33		
2		49		
3		38		
TOTAL		**120**		

Comment *(for official use only)*

____

____

____

____

____

____