education

Department:
Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2010

MARKS: 120

TIME: 3 hours

This question paper consists of 26 pages, 3 annexures and an information sheet.

INSTRUCTIONS AND INFORMATION

1. This is a three-hour examination. Because of the nature of this examination it is important to note that you will NOT be permitted to leave the examination room before the end of the examination session.

2. Answer EITHER 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/educator will tell you where to find them on the hard drive of the workstation you are using OR in which network folder it is. If the files are issued to you on a CD, you need to copy them onto your hard disk.

   QUESTION 1

Delphi:	**Java:**
FatFightersDB.mdb	FatFightersDB.mdb
Question1_U.pas	FatFighters.java
Question1_P.dpr	PeopleTb.txt
Question1_U.dfm	TestFF.java
PeopleTb.txt	WeighinTb.txt
WeighinTb.txt	FatFightersDB.odb

   QUESTION 2

Delphi:	**Java:**
Question2_P.dpr	TestMeals.java
Question2_U.dfm	Meals.txt
Question2_U.pas	
Meals.txt	

   QUESTION 3

Delphi:	**Java:**
Question3_P.dpr	TestSMSCompetition.java
Question3_U.pas	
Question3_U.dfm	

   If a disk or CD containing the files was issued to you, write your centre number and 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 only what is required by the question.

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 refer to any other resource material.

9. At the end of this examination session you will be required to hand in the stiffy or CD given to you by the invigilator with your work saved on it, or you must make sure that all your work has been saved on the network as explained to you by the invigilator/educator. Ensure that all files can be read.

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

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

12. Complete the separate information sheet that has been provided with the question paper and hand it to the invigilator at the end of this examination session.

SECTION A

Answer this section only if you studied **Delphi**.

Answer ALL the questions in this section.

SCENARIO

Fat Fighters is an organisation which tries to help people live healthy lifestyles and lose weight. Until now all their recipes, systems and records have been on manual systems. They have asked you to help them computerise their organisation by using computer software such as databases.

QUESTION 1: DELPHI – PROGRAMMING AND DATABASE

Fat Fighters run a programme which helps members monitor their weight on a fortnightly basis (every second week). Members register at a cost and their initial weight is recorded, as well as the goal weight they would like to achieve. Every second week they come to the Fat Fighters office to get themselves weighed and to have their weight recorded. A database called **FatFightersDB** has been developed. An incomplete program has been developed to process queries on the information in the given database. Your task will be to complete this program.

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

NOTE: If you cannot use the database provided, follow the instructions on **ANNEXURE B** to create the database before you answer any of the questions (QUESTIONS 1.1 to 1.6).

You received an incomplete Delphi project named **Question1_P.dpr** in the folder named **Question 1 Delphi**.

Do the following:

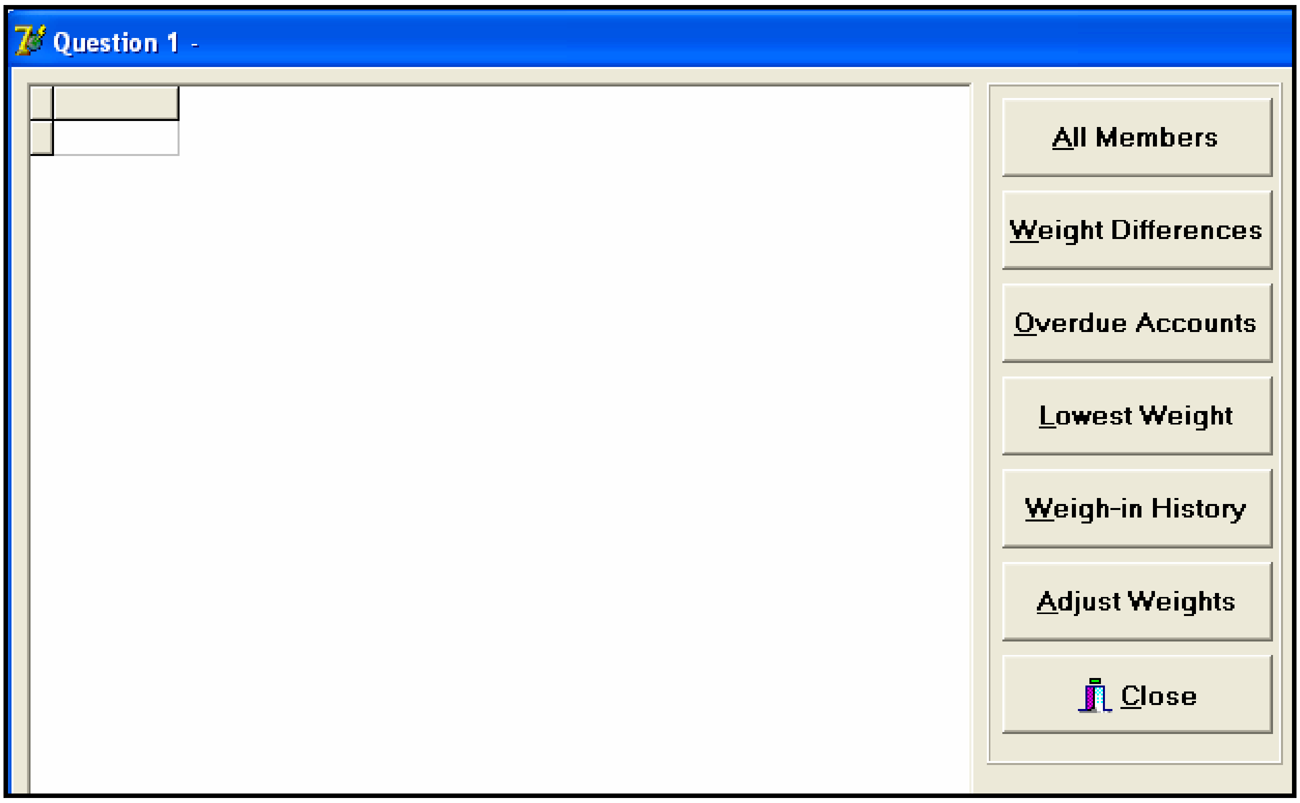
- Rename the folder **Question 1 Delphi** as **Quest1_X**, where X should be replaced with your examination number.
- Open Delphi and then open the file **Question1_P.dpr** in the folder **Quest1_X**. The program displays seven buttons and a DBGrid that will be used as an output component (see example on the next page).
- Add your examination number to the caption of the form to the right of 'Question 1 –'.
- Go to 'File/Save As …' and save the unit as **Question1_UXXXX** (where XXXX represents the last FOUR digits of your examination number).
- Go to 'File/Save Project As …' and save the project as **Question1_PXXXX** (where XXXX represents the last FOUR digits of your examination number).
- The program should be able to connect to the database **FatFightersDB**. When you do QUESTION 1.1 (which follows on the next page) and you find that the connectivity is not in place, use the steps in **ANNEXURE C** to establish connection with the database.

HINT: If your program cannot connect to the database, make sure that the database file **FatFightersDB** is in the same folder as your program. Your program will not work if the database file is in a different folder than your program. If not, copy the database file **FatFightersDB** 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 programming code and submit it for marking.

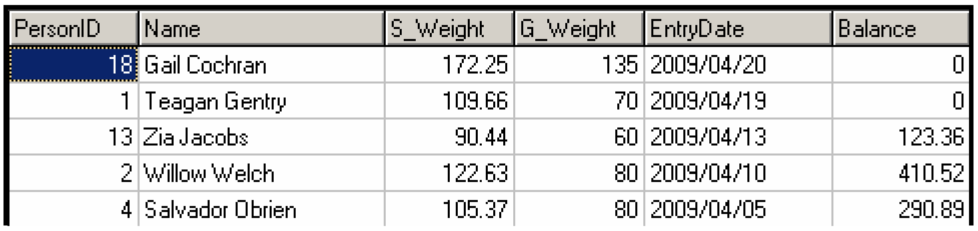
Marks will only be awarded for the programming code that contain the SQL statements in the unit named Question1_UXXXX.

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



1.1 Fat Fighters want to see a list with all the information on all of their members. Complete the code for the **All Members** button by formulating an SQL statement to display all the details of each member stored in the **tblPeople** table. Order the results by **EntryDate** in descending order.

Example of output of the first few records (on the next page):



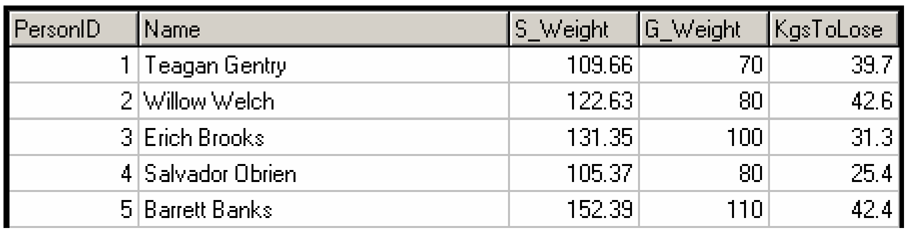
PersonID	Name	S_Weight	G_Weight	EntryDate	Balance
18	Gail Cochran	172.25	135	2009/04/20	0
1	Teagan Gentry	109.66	70	2009/04/19	0
13	Zia Jacobs	90.44	60	2009/04/13	123.36
2	Willow Welch	122.63	80	2009/04/10	410.52
4	Salvador Obrien	105.37	80	2009/04/05	290.89

:

NOTE: The date on your output may be in a different format, depending on the settings on your computer. Any format of the date will be acceptable. (5)

1.2 Fat Fighters want to know how much weight their members want to lose. This is calculated by subtracting their goal weight (**G_Weight**) from their initial weight (**S_Weight**). Complete the code for the **Weight Differences** button by formulating an SQL statement to display the **PersonID**, **Name**, **S_Weight**, **G_Weight** and the difference between the two weights in a field named **KgsToLose** for all the members. **KgsToLose** is a calculated field and must be rounded off to ONE decimal place.

Example of output of the first few records:



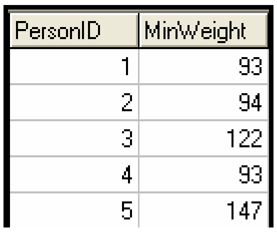
PersonID	Name	S_Weight	G_Weight	KgsToLose
1	Teagan Gentry	109.66	70	39.7
2	Willow Welch	122.63	80	42.6
3	Erich Brooks	131.35	100	31.3
4	Salvador Obrien	105.37	80	25.4
5	Barrett Banks	152.39	110	42.4

: (6)

1.3 Due to non-payment by members, Fat Fighters want to delete all members who have an outstanding balance greater than R400, except for **Uriel Knight** who has made special arrangements in terms of her account. Complete the code for the **Overdue Accounts** button by formulating an SQL statement to delete all members from the **tblPeople** table who have an outstanding balance greater than R400, except for **Uriel Knight**. (6)

1.4 Members want to know what their lowest weight was so far. Complete the code for the **Lowest Weight** button by formulating an SQL statement that will list the **PersonID** and the minimum weight (name this field **MinWeight**) of each individual member from the **tblWeighin** table.

Example of output:

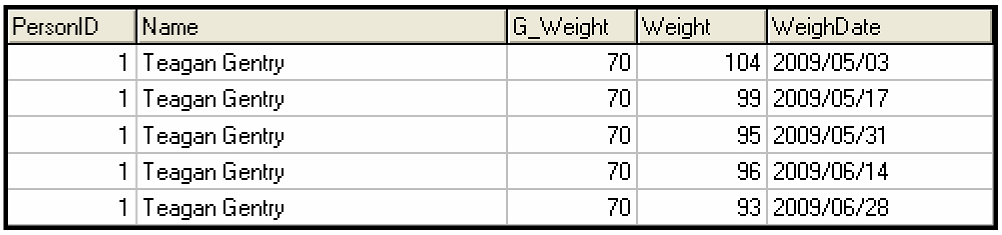


PersonID	MinWeight
1	93
2	94
3	122
4	93
5	147

: (6)

1.5 The program must allow the user to look up the weigh-in history of a member. Complete the code for the **Weigh-in History** button by allowing the user to enter the **PersonID** of a member and then formulate an SQL statement that will display the weigh-in history of the member. List the **PersonID**, **Name**, **G_Weight**, **Weight** and the **WeighDate** from the **tblWeighin** and **tblPeople** tables.

Example of output if the **PersonID** entered is 1:

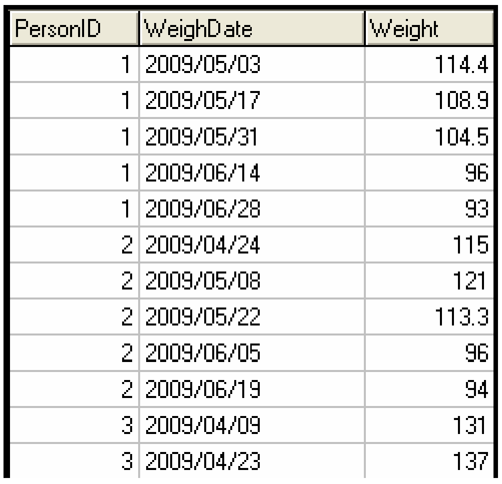


PersonID	Name	G_Weight	Weight	WeighDate
1	Teagan Gentry	70	104	2009/05/03
1	Teagan Gentry	70	99	2009/05/17
1	Teagan Gentry	70	95	2009/05/31
1	Teagan Gentry	70	96	2009/06/14
1	Teagan Gentry	70	93	2009/06/28

(10)

1.6 Fat Fighters have just realised that the scale that they used to weigh their members gave incorrect readings during the month of May. All readings for May should be 10% higher than recorded. Complete the code for the **Adjust Weights** button by formulating an SQL statement that will increase all readings in the **tblWeighin** table that were done in May by 10%. The contents of the **tblWeighin** table will be displayed after the adjustment has been done.

Example of output of the first few records:



PersonID	WeighDate	Weight
1	2009/05/03	114.4
1	2009/05/17	108.9
1	2009/05/31	104.5
1	2009/06/14	96
1	2009/06/28	93
2	2009/04/24	115
2	2009/05/08	121
2	2009/05/22	113.3
2	2009/06/05	96
2	2009/06/19	94
3	2009/04/09	131
3	2009/04/23	137

:

(7)

- 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 of the **Question1_UXXXX.pas** file will be required.

[40]

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

Fat Fighters allow their members to eat a limited amount of fats, proteins and carbohydrates (carbs) per meal. Points are calculated according to the amount of fats, proteins and carbohydrates eaten. Higher points indicate an unhealthy meal and lower points indicate a healthy meal. When points are calculated it should not exceed 50 points per day to be acceptable. During each meal (breakfast, lunch and supper) members record the amount of fats, proteins and carbohydrates that they eat. This record is in the form of a text file which is later e-mailed to FatFighters. You are required to write a program (as indicated in QUESTIONS 2.1 and 2.2) to process these text files.

The data stored in the text file named **Meals.txt** contains information on the daily meals for one person. The format of the information per day is as follows:

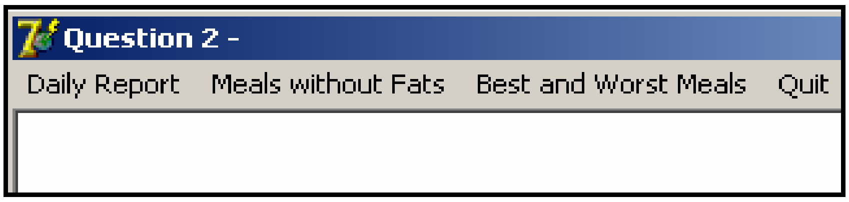
Day#Meal Time#Fats#Proteins#Carbs

An example of the data in the text file is:

```
Sun#Breakfast#2#0#1
Sun#Lunch#0#2#3
Wed#Supper#3#0#1
Thu#Breakfast#2#1#2
Thu#Lunch#1#2#2
Sun#Supper#2#3#0
Mon#Breakfast#0#2#2
Tue#Supper#3#2#2
Wed#Breakfast#1#2#1
Wed#Lunch#3#2#3
Thu#Supper#1#2#3
Fri#Breakfast#3#3#3
Mon#Lunch#0#1#1
Fri#Supper#3#2#3
Sat#Breakfast#3#1#2
Sat#Lunch#1#2#2
Mon#Supper#3#0#2
Tue#Breakfast#2#2#2
Tue#Lunch#3#0#1
Fri#Lunch#0#3#1
Sat#Supper#1#0#2
```

Do the following:

- Rename the folder **Question 2 Delphi** as **Quest2_X** (where X should be replaced with your examination number).
- Open Delphi and then open the file **Question2_P.dpr** in the folder **Quest2_X**.
- Go to 'File/Save As …' and save the unit as **Question2XXXX_U** (where XXXX represents the last FOUR digits of your examination number).
- Go to 'File/Save Project As …' and save the project as **Question2XXXX_P** (where XXXX represents the last FOUR digits of your examination number).
- The following menu will be displayed when you execute the program:



- Add your examination number to the caption of the form to the right of 'Question 2 –'.

2.1 Create an object class (another unit) named **MealXXXX** and save this unit as **MealXXXX** in your **Quest2_X** folder. XXXX should be replaced by the last FOUR digits of your examination number. All fields in this class are private and all methods public. The fields and methods you need to create and code are described below:

2.1.1 Define a class named **TMeal**. Add appropriately named and typed private fields to hold the following data:

- Day
- Meal (breakfast, lunch, supper)
- Amount of fats
- Amount of proteins
- Amount of carbohydrates (6 ÷ 2) (3)

2.1.2 Write a constructor method which accepts the following parameters: day, meal, fats, proteins and carbohydrates. All the fields must be initialised in the constructor. (4 ÷ 2) (2)

2.1.3 Write an appropriately named 'get' method (accessor method) to return the day of the meal. (4 ÷ 2) (2)

2.1.4 Write a typed method (function method) called **noFats** that returns the value **true** if the meal has a zero value for the fat amount and **false** if it has a non-zero value. (4 ÷ 2) (2)

2.1.5 Write a typed method (function method) called **calculatePoints** that returns the total number of points for a particular meal. The total points for a meal are calculated as follows:

(fats x 4) + (proteins x 2) + (carbohydrates x 2)

In addition, if the meal has a zero value for the amount of fats, 2 points are deducted from the total. If the meal has a value of more than 2 for fats, 10 points are added to the total. (6 ÷ 2) (3)

2.1.6 Write a method (function method) called **toString** that constructs and returns a string with information on the meal, formatted as follows:

Day<tab>Fats<tab>Proteins<tab>Carbohydrates<tab>Points<tab>Meal (6 ÷ 2) (3)

2.2 Write code to do the following in the **Question2XXXX_U.pas** file (the main unit) in the given program:

2.2.1 Create an array named **arrMeals** that holds 100 objects of **TMeal**. Write code in the **OnActivate** event handler of the form to read information from the text file **Meals.txt** into the array according to the following steps:

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

(b) Use a loop to do the following:
- Read a line of text from the text file.
- Separate the text into day, meal, fats, proteins and carbohydrates.
- Use this information to create a new **TMeal** object and place the object in the array named **arrMeals.**

(c) Use a counter field to keep track of the number of items in the array. (26 ÷ 2) (13)

2.2.2 **Menu Option: Daily Report**

When the user selects this menu option the program must:

(a) Allow the user to enter the first THREE letters of a day of the week

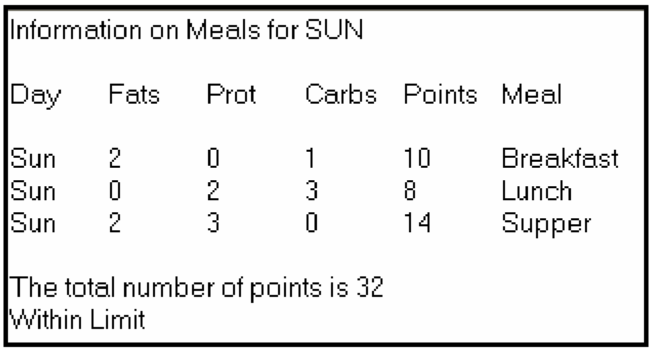
(b) Search through the array and each time a meal for that day is found:
- Display the meal's information using the **toString** method
- Add the total points for each meal to calculate the total number of points recorded in the file for that day

When the search is complete the program must:

(a) Display the total points for all meals for that day

(b) Display a message **'Within Limit'** if the total for the day is less than or equal to 50 points, or display a message **'Limit Exceeded'** if the total points for that day is greater than 50

Example: User enters **Sun**



```
Information on Meals for SUN

Day     Fats    Prot    Carbs   Points  Meal

Sun     2       0       1       10      Breakfast
Sun     0       2       3       8       Lunch
Sun     2       3       0       14      Supper

The total number of points is 32
Within Limit
```

(14 ÷ 2) (7)

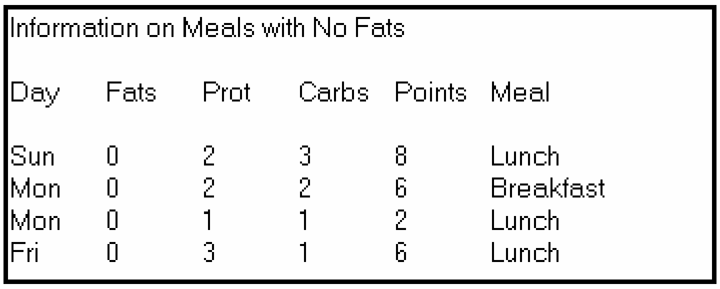
2.2.3 **Menu Option: Meals without Fats**

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

(a) Search the array for all meals which have zero fats.

(b) When one such meal is found, it should be displayed using the **toString** method.

Example of output:



```
Information on Meals with No Fats

Day     Fats    Prot    Carbs   Points  Meal

Sun     0       2       3       8       Lunch
Mon     0       2       2       6       Breakfast
Mon     0       1       1       2       Lunch
Fri     0       3       1       6       Lunch
```

(6 ÷ 2) (3)

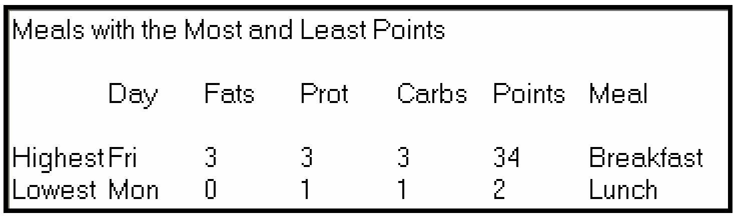
2.2.4 **Menu Option: Best and Worst Meals**

When the user selects this menu option the program must:

(a) Process the meals in the array and determine, using the **calculatePoints** method, the meal with the:
- Highest number of points
- Lowest number of points

(b) Display the information of the meals using the **toString** method once the meals with the highest and the lowest number of points have been determined

Example of output:



```
Meals with the Most and Least Points

        Day     Fats    Prot    Carbs   Points  Meal

Highest Fri     3       3       3       34      Breakfast
Lowest  Mon     0       1       1       2       Lunch
```

(12 ÷ 2) (6)

- Enter your examination number as a comment in the first line of the main class **Question2XXXX_U.pas**, as well as in the object class **MealXXXX.pas**.
- Save all the files ('File/Save All').
- Printouts of the code for the classes **Question2XXXX_U.pas** and **MealXXXX.pas** will be required.

[44]

QUESTION 3: DELPHI – PROGRAMMING

Fat Fighters is busy with an SMS competition where the following question is asked: 'What activity will contribute towards a healthy lifestyle – Eat, Sleep or Exercise?' The correct answer to the question is **Exercise**. Participants will SMS their answers using their cellphones to stand a chance of winning membership to HealthE for a period of one year. You will be required to assist in completing a program that will process the results of this competition.

You received an incomplete program in the folder named **Question 3 Delphi.** The program generates an array of 20 strings in the following format:

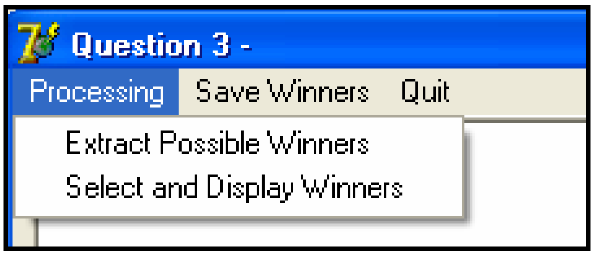
```
Cellphone number of the participant : answer
```

Example of the first few entries in the array:

```
arrEntries[1] := '082 345 4571:Exercise';
arrEntries[2] := '082543 2341:Exercise';
arrEntries[3] := '082 234 1241:EXERCISE';
arrEntries[4] := '0821239876:Eat';
arrEntries[5] := '083123 6123:Sleep';
:
```

Do the following:

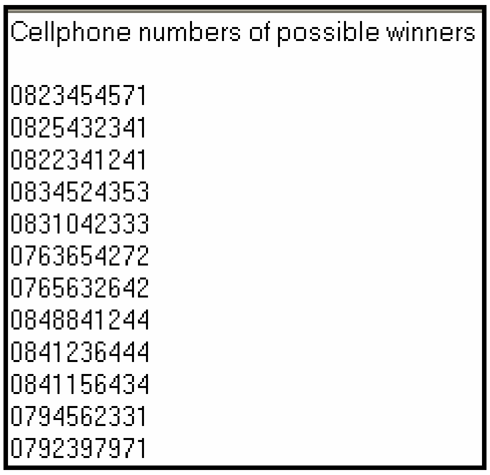
- Rename the folder named **Question 3 Delphi** to **Quest3_X** (where X should be replaced with your examination number).
- Open the Delphi program in this folder.
- Save the unit as ('File/Save As') **SMSCompXXXX_U** and the project as ('File/Save Project As') **SMSCompXXXX_P** inside the folder (XXXX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



3.1 Complete the **Extract Possible Winners** option on the menu so that the information in the array will be processed and the cellphone numbers of all the participants who sent an SMS with the correct answer will be displayed. As part of the solution, you have to write a subprogram with passing of parameter(s) to remove all the spaces from a cellphone number. The cellphone numbers must be displayed in the format shown on the next page.

NOTE: This list of cellphone numbers will be required again in the solution to QUESTION 3.2.

Example of output:



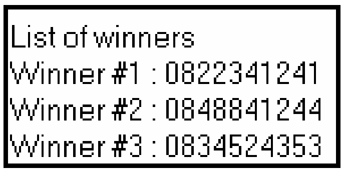
```
Cellphone numbers of possible winners

0823454571
0825432341
0822341241
0834524353
0831042333
0763654272
0765632642
0848841244
0841236444
0841156434
0794562331
0792397971
```

(19)

3.2 In the option **Select and Display Winners**, the program must randomly select and display three winners from the list of possible winners generated in QUESTION 3.1 (see example of output in QUESTION 3.1 above). The program must ensure that a person can only be selected once as a winner. A specific cellphone number should therefore only appear once in the list of winners. Display the cellphone numbers of the winners in the following format:

```
Winner # :  cellphone number
```

Example of output:

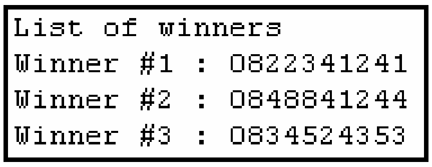


```
List of winners
Winner #1 : 0822341241
Winner #2 : 0848841244
Winner #3 : 0834524353
```

NOTE: The cellphone numbers of the winners will be different each time this option is executed due to random selection. (10)

3.3 In the **Save Winners** option, write a heading and the cellphone numbers of the three winners to a text file in the same format as in QUESTION 3.2.

Example of the content of the text file:



```
List of winners
Winner #1 : 0822341241
Winner #2 : 0848841244
Winner #3 : 0834524353
```

(7)

- Enter your examination number as a comment in the first line of the unit **SMSCompXXXX_U**.
- Save the unit and the project ('File/Save All').
- A printout of the code for the unit **SMSCompXXXX_U** will be required. **[36]**

TOTAL: 120

SECTION B

Answer this section only if you studied **Java**.

Answer ALL the questions in this section.

SCENARIO

Fat Fighters is an organisation which tries to help people live healthy lifestyles and lose weight. Until now all their recipes, systems and records have been on manual systems. They have asked you to help them computerise their organisation by using computer software such as databases.

QUESTION 1: JAVA – PROGRAMMING AND DATABASE

Fat Fighters run a programme which helps members monitor their weight on a fortnightly basis (every second week). Members register at a cost and their initial weight is recorded, as well as the goal weight they would like to achieve. Every second week they come to the Fat Fighters office to get themselves weighed and to have their weight recorded. A database called **FatFightersDB** has been developed. An incomplete program has been developed to process queries on the information in the given database. Your task will be to complete this program.

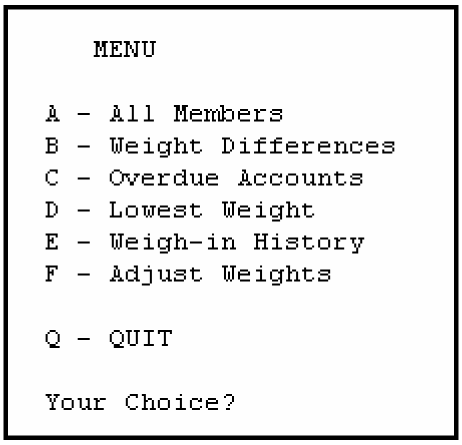
NOTE: The design of the tables in the **FatFightersDB** database and sample data for this question can be found in **ANNEXURE A: Table Description Sheet**.

NOTE: If you cannot use the database provided, follow the instructions in **ANNEXURE B** to create the database before you answer any of the questions (QUESTIONS 1.1 to 1.6).

You received an incomplete Java program, in the folder named **Question 1 Java**, with a test class named **TestFF.java** and an object class named **FatFighters.java** which will display the results of the queries.

Do the following:

- Rename the folder **Question 1 Java** as **Quest1_X**, where X should be replaced with your examination number.
- Open this folder and rename the **TestFF.java** file as **TestFFXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Open the incomplete program **TestFFXXXX.java**. Change the name of the class to **TestFFXXXX** (where XXXX represents the last FOUR digits of your examination number).
- The program will not run because of the incomplete SQL-statements. Once you have entered the correct SQL statements, the program will display a simple menu with seven options (see next page).



```
    MENU

A - All Members
B - Weight Differences
C - Overdue Accounts
D - Lowest Weight
E - Weigh-in History
F - Adjust Weights

Q - QUIT

Your Choice?
```

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 **FatFighters.java**.

HINT: If your program cannot connect to the database, make sure that the database file **FatFightersDB** is in the same folder as the files **TestFFXXXX.java** and **FatFighters.java**. Your program will not work if the database file is in a different folder than your Java program. If not, copy the database file **FatFightersDB** 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 programming code and submit it for marking.

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

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

Complete the SQL statements in the file **TestFFXXXX.java** for each menu option as indicated in QUESTIONS 1.1 to 1.6 below. The code to pass the SQL statements to the relevant methods in the file named **FatFighters.java** has been given to you. You need only complete the SQL statements in the **TestFFXXXX.java** file.

1.1 Fat Fighters want to see a list with all the information on all of their members. Complete the code for the **All Members** menu option by formulating an SQL statement to display all the details of each member stored in the **tblPeople** table. Order the results by **EntryDate** in descending order.

Example of output of the first few records (on the next page):

```
PersonID  Name              S_Weight  G_Weight  EntryDate   Balance
======================================================================
18        Gail Cochran      172.25    135       2009-04-20   0.0
1         Teagan Gentry     109.66    70        2009-04-19   0.0
13        Zia Jacobs        90.44     60        2009-04-13   123.36
2         Willow Welch      122.63    80        2009-04-10   410.52
4         Salvador Obrien   105.37    80        2009-04-05   290.89
:
```

NOTE: The date on your output may be in a different format, depending on the settings on your computer. Any format of the date will be acceptable. (5)

1.2 Fat Fighters want to know how much weight their members want to lose. This is calculated by subtracting their goal weight (**G_Weight**) from their initial weight (**S_Weight**). Complete the code for the **Weight Differences** menu option by formulating an SQL statement to display the **PersonID**, **Name**, **S_Weight**, **G_Weight** and the difference between the two weights in a field named **KgsToLose** for all the members. **KgsToLose** is a calculated field and must be rounded off to ONE decimal place.

Example of output of the first few records:

```
PersonID  Name              S_Weight  G_Weight  KgsToLose
============================================================
1         Teagan Gentry     109.66    70        39.7
2         Willow Welch      122.63    80        42.6
3         Erich Brooks      131.35    100       31.3
4         Salvador Obrien   105.37    80        25.4
5         Barrett Banks     152.39    110       42.4
:
```

(6)

1.3 Due to non-payment by members, Fat Fighters want to delete all members who have an outstanding balance greater than R400, except for **Uriel Knight** who has made special arrangements in terms of her account. Complete the code for the **Overdue Accounts** menu option by formulating an SQL statement to delete all members from the **tblPeople** table who have an outstanding balance greater than R400, except for **Uriel Knight**. (6)

1.4 Members want to know what their lowest weight was so far. Complete the code for the **Lowest Weight** menu option by formulating an SQL statement that will list the **PersonID** and the minimum weight (name this field **MinWeight**) of each individual member from the **tblWeighin** table.

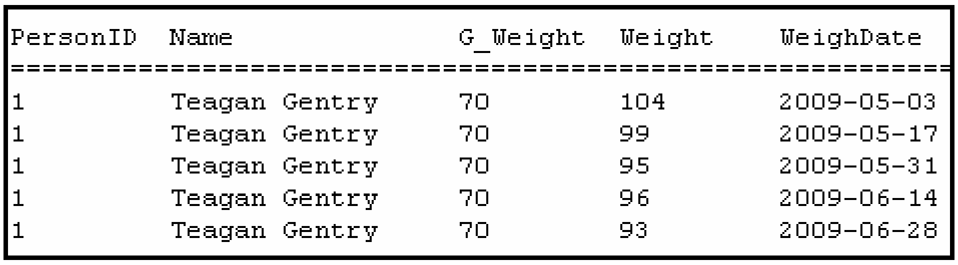
Example of output:

```
PersonID  MinWeight
====================
1         93
2         94
3         122
4         93
5         147
:
```

(6)

1.5 The program must allow the user to look up the weigh-in history of a member. Complete the code for the **Weigh-in History** menu option by allowing the user to enter the **PersonID** of a member and then formulate an SQL statement that will display the weigh-in history of the member. List the **PersonID**, **Name**, **G_Weight**, **Weight** and the **WeighDate** from the **tblWeighin** and **tblPeople** tables.

Example of output if the **PersonID** entered is 1:

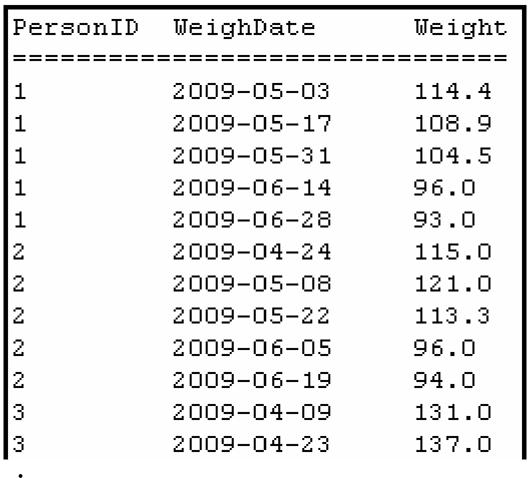


```
PersonID  Name              G_Weight  Weight    WeighDate
==========================================================
1         Teagan Gentry     70        104       2009-05-03
1         Teagan Gentry     70        99        2009-05-17
1         Teagan Gentry     70        95        2009-05-31
1         Teagan Gentry     70        96        2009-06-14
1         Teagan Gentry     70        93        2009-06-28
```

(10)

1.6 Fat Fighters have just realised that the scale that they used to weigh their members gave incorrect readings during the month of May. All readings for May should be 10% higher than recorded. Complete the code for the **Adjust Weights** menu option by formulating an SQL statement that will increase all readings in the **tblWeighin** table that were done in May by 10%. The contents of the **tblWeighin** table will be displayed after the adjustment has been done.

Example of output of the first few records:



```
PersonID  WeighDate       Weight
================================
1         2009-05-03      114.4
1         2009-05-17      108.9
1         2009-05-31      104.5
1         2009-06-14      96.0
1         2009-06-28      93.0
2         2009-04-24      115.0
2         2009-05-08      121.0
2         2009-05-22      113.3
2         2009-06-05      96.0
2         2009-06-19      94.0
3         2009-04-09      131.0
3         2009-04-23      137.0
```

:

(7)

- Enter your examination number as a comment in the first line of the file named **TestFFXXXX.java** containing the SQL statements.
- Save the **TestFFXXXX.java** and the **FatFighters.java** files.
- A printout of the code of the **TestFFXXXX.java** file will be required.

[40]

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING

Fat Fighters allow their members to eat a limited amount of fats, proteins and carbohydrates (carbs) per meal. Points are calculated according to the amount of fats, proteins and carbohydrates eaten. Higher points indicate an unhealthy meal and lower points indicate a healthy meal. When points are calculated it should not exceed 50 points per day to be acceptable. During each meal (breakfast, lunch and supper) members record the amount of fats, proteins and carbohydrates that they eat. This record is in the form of a text file which is later e-mailed to FatFighters. You are required to write a program (as indicated in QUESTION 2.1 and 2.2) to process these text files.

The data stored in the text file named **Meals.txt** contains information on the daily meals for one person. The format of the information per day is as follows:

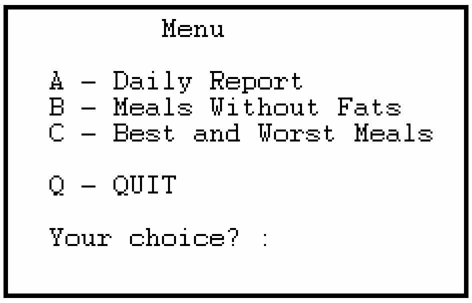
Day#Meal Time#Fats#Proteins#Carbs

An example of the data in the text file is:

```
Sun#Breakfast#2#0#1
Sun#Lunch#0#2#3
Wed#Supper#3#0#1
Thu#Breakfast#2#1#2
Thu#Lunch#1#2#2
Sun#Supper#2#3#0
Mon#Breakfast#0#2#2
Tue#Supper#3#2#2
Wed#Breakfast#1#2#1
Wed#Lunch#3#2#3
Thu#Supper#1#2#3
Fri#Breakfast#3#3#3
Mon#Lunch#0#1#1
Fri#Supper#3#2#3
Sat#Breakfast#3#1#2
Sat#Lunch#1#2#2
Mon#Supper#3#0#2
Tue#Breakfast#2#2#2
Tue#Lunch#3#0#1
Fri#Lunch#0#3#1
Sat#Supper#1#0#2
```

Do the following:

- Rename the folder **Question 2 Java** as **Quest2_X** (where X should be replaced with your examination number).
- Open this folder and rename the **TestMeals.java** file as **TestMealsXXXX.java** (where XXXX must be replaced by the last FOUR digits of your examination number).
- Open the incomplete program **TestMealsXXXX.java**. Change the name of the class to **TestMealsXXXX** (where XXXX must be replaced by the last FOUR digits of your examination number).
- The following menu will be displayed when you execute the program:



```
          Menu

  A - Daily Report
  B - Meals Without Fats
  C - Best and Worst Meals

  Q - QUIT

  Your choice? :
```

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

2.1 Create an object class named **MealXXXX.java** and save this class as **MealXXXX.java** in your **Quest2_X** folder. XXXX should be replaced by the last FOUR digits of your examination number. All fields in this class are private and all methods public. The fields and methods you need to create and code are described below:

2.1.1 In the **MealXXXX** class, add appropriately named and typed private fields to hold the following data:

- Day
- Meal (breakfast, lunch, supper)
- Amount of fats
- Amount of proteins
- Amount of carbohydrates (6 ÷ 2) (3)

2.1.2 Write a constructor method which accepts the following parameters: day, meal, fats, proteins and carbohydrates. All the fields must be initialised in the constructor. (4 ÷ 2) (2)

2.1.3 Write an appropriately named 'get' method (accessor method) to return the day of the meal. (4 ÷ 2) (2)

2.1.4 Write a typed method called **noFats** that returns the value **true** if the meal has a zero value for the fat amount and **false** if it has a non-zero value. (4 ÷ 2) (2)

2.1.5 Write a typed method called **calculatePoints** that returns the total number of points for a particular meal. The total points for a meal are calculated as follows:

(fats x 4) + (proteins x 2) + (carbohydrates x 2)

In addition, if the meal has a zero value for the amount of fats, 2 points are deducted from the total. If the meal has a value of more than 2 for fats, 10 points are added to the total. (6 ÷ 2) (3)

2.1.6 Write a method called **toString** that constructs and returns a string with information on the meal, formatted as follows:

Day<tab>Fats<tab>Proteins<tab>Carbohydrates<tab>Points<tab>Meal (6 ÷ 2) (3)

2.2 Write code to do the following in the **TestMealXXXX.java** file (the main class) in the given program:

2.2.1 Create an array named **arrMeals** that holds 100 objects of **MealXXXX**. Write code to read information from the text file **Meals.txt** according to the following steps:

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

(b) Use a loop to do the following:
- Read a line of text from the text file.
- Separate the text into day, meal, fats, proteins and carbohydrates.
- Use this information to create a new **MealXXXX** object and place the object in the array named **arrMeals.**

(c) Use a counter field to keep track of the number of items in the array. (26 ÷ 2) (13)

2.2.2 **Menu Option A: Daily Report**

When the user selects this menu option the program must:

(a) Allow the user to enter the first THREE letters of a day of the week

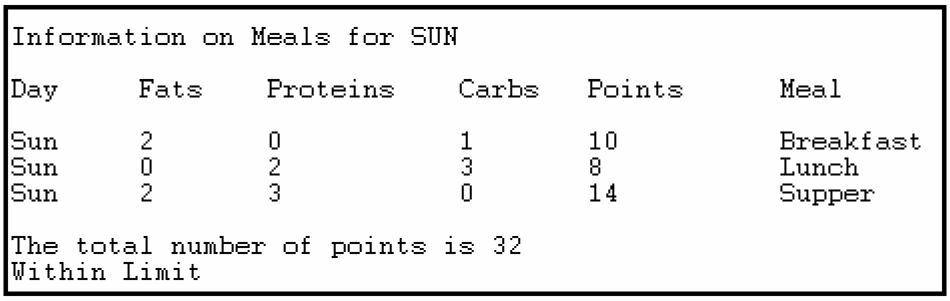
(b) Search through the array and each time a meal for that day is found:
- Display the meal's information using the **toString** method
- Add the total points for each meal to calculate the total number of points recorded in the file for that day

When the search is completed the program must:

(a) Display the total points for all meals for that day

(b) Display a message **'Within Limit'** if the total for the day is less than or equal to 50 points, or display a message **'Limit Exceeded'** if the total points for that day is greater than 50

Example: User enters **Sun**



```
Information on Meals for SUN

Day      Fats     Proteins     Carbs    Points       Meal

Sun      2        0            1        10           Breakfast
Sun      0        2            3        8            Lunch
Sun      2        3            0        14           Supper

The total number of points is 32
Within Limit
```

(14 ÷ 2) (7)

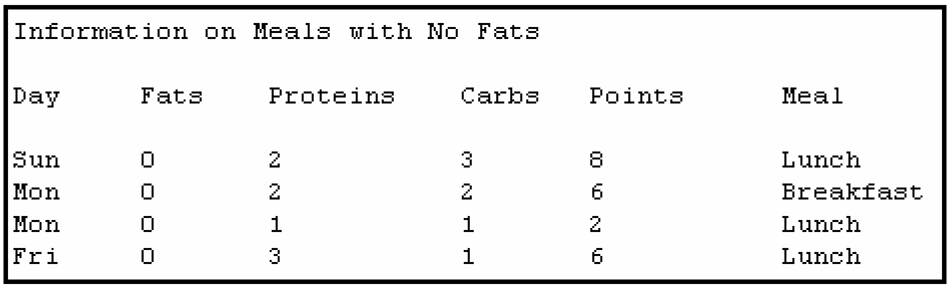
2.2.3 **Menu Option B: Meals without Fats**

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

(a) Search the array for all meals which have zero fats.

(b) When one such meal is found, it should be displayed using the **toString** method.

Example of output:



```
Information on Meals with No Fats

Day      Fats     Proteins     Carbs    Points       Meal

Sun      0        2            3        8            Lunch
Mon      0        2            2        6            Breakfast
Mon      0        1            1        2            Lunch
Fri      0        3            1        6            Lunch
```

(3)

(6 ÷ 2)

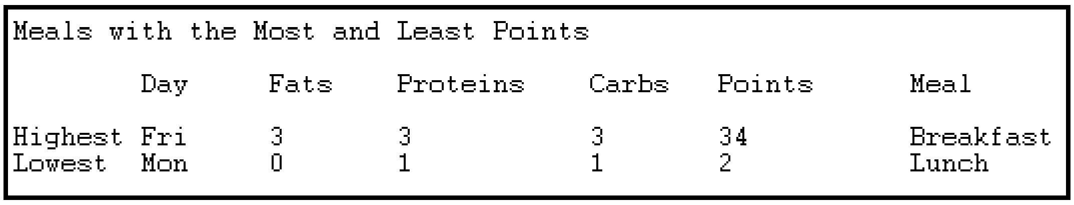
2.2.4 **Menu Option C: Best and Worst Meals**

When the user selects this menu option the program must:

(a) Process the meals in the array and determine, using the **calculatePoints** method, the meal with the:
- Highest number of points
- Lowest number of points

(b) Display the information of the meals using the **toString** method once the meals with the highest and the lowest number of points have been determined

Example of output:



```
Meals with the Most and Least Points

        Day     Fats    Proteins    Carbs   Points      Meal

Highest Fri     3       3           3       34          Breakfast
Lowest  Mon     0       1           1       2           Lunch
```

(12 ÷ 2) (6)

- Enter your examination number as a comment in the first line of the main class **TestMealsXXXX.java-B** as well as in the object class **MealXXXX**.
- Save all the files ('File/Save All').
- Printouts of the code for the classes **TestMealsXXXX.java** and **MealXXXX** will be required.

[44]

QUESTION 3: JAVA – PROGRAMMING

Fat Fighters is busy with an SMS competition where the following question is asked: 'What activity will contribute towards a healthy lifestyle – Eat, Sleep or Exercise?' The correct answer to the question is **Exercise**. Participants will SMS their answers using their cellphones to stand a chance of winning membership to HealthE for a period of one year. You will be required to assist in completing a program that will process the results of this competition.

You received an incomplete program in the folder named **Question 3 Java.** The program generates an array of 20 strings in the following format:

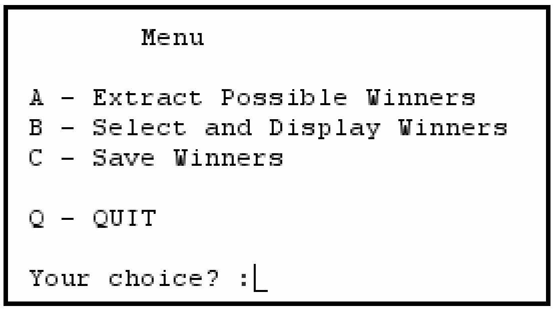
```
Cellphone number of the participant : answer
```

Example of the first few entries in the array:

```
arrEntries[1] := '082 345 4571:Exercise';
arrEntries[2] := '082543 2341:Exercise';
arrEntries[3] := '082 234 1241:EXERCISE';
arrEntries[4] := '0821239876:Eat';
arrEntries[5] := '083123 6123:Sleep';
:
```

Do the following:

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

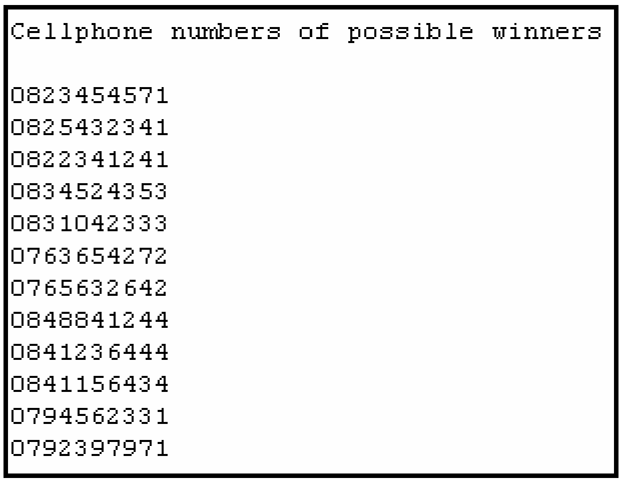


```
        Menu

A - Extract Possible Winners
B - Select and Display Winners
C - Save Winners

Q - QUIT

Your choice? :_
```

3.1 Complete the **Extract Possible Winners** option on the menu so that the information in the array will be processed and the cellphone numbers of all the participants who sent an SMS with the correct answer will be displayed. As part of the solution, you have to write a method with passing parameters to remove all the spaces from a cellphone number. The cellphone numbers must be displayed in the format shown on the next page.

NOTE: This list of cellphone numbers will be required again in the solution to QUESTION 3.2.

Example of output:



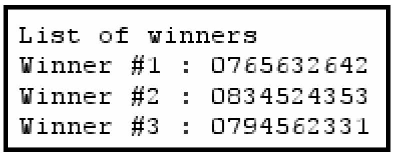
```
Cellphone numbers of possible winners

0823454571
0825432341
0822341241
0834524353
0831042333
0763654272
0765632642
0848841244
0841236444
0841156434
0794562331
0792397971
```

(19)

3.2 In the option **Select and Display Winners**, the program must randomly select and display three winners from the list of possible winners generated in QUESTION 3.1 (see example of output in QUESTION 3.1 above). The program must ensure that a person can only be selected once as a winner. A specific cellphone number should therefore only appear once in the list of winners. Display the cellphone numbers of the winners in the following format:

```
Winner # :  cellphone number
```

Example of output:

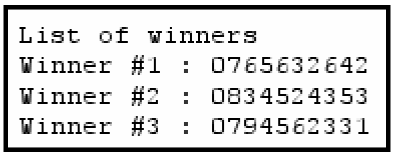


```
List of winners
Winner #1 : 0765632642
Winner #2 : 0834524353
Winner #3 : 0794562331
```

NOTE: The cellphone numbers will be different each time this option is executed due to random selection. (10)

3.3 In the **Save Winners** option, write a heading and the cellphone numbers of the three winners to a text file in the same format as in QUESTION 3.2.

Example of the content of the text file:



```
List of winners
Winner #1 : 0765632642
Winner #2 : 0834524353
Winner #3 : 0794562331
```

(7)

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

[36]

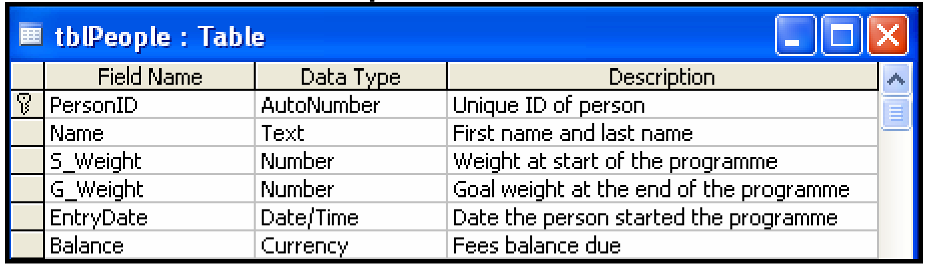
TOTAL: 120

MPUMALANGA

ANNEXURE A: Table Description Sheet

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

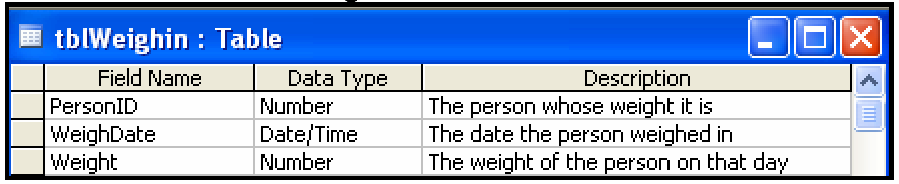
tblPeople Table Structure



tblPeople : Table

Field Name	Data Type	Description
PersonID	AutoNumber	Unique ID of person
Name	Text	First name and last name
S_Weight	Number	Weight at start of the programme
G_Weight	Number	Goal weight at the end of the programme
EntryDate	Date/Time	Date the person started the programme
Balance	Currency	Fees balance due

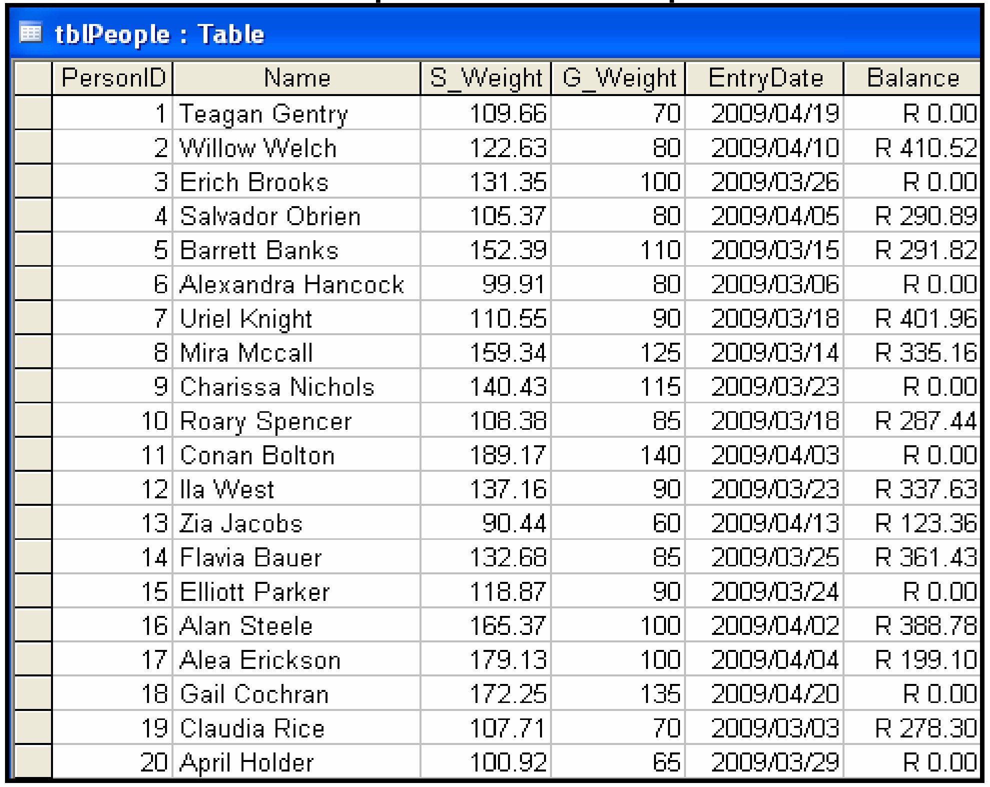
tblWeighin Table Structure



tblWeighin : Table

Field Name	Data Type	Description
PersonID	Number	The person whose weight it is
WeighDate	Date/Time	The date the person weighed in
Weight	Number	The weight of the person on that day

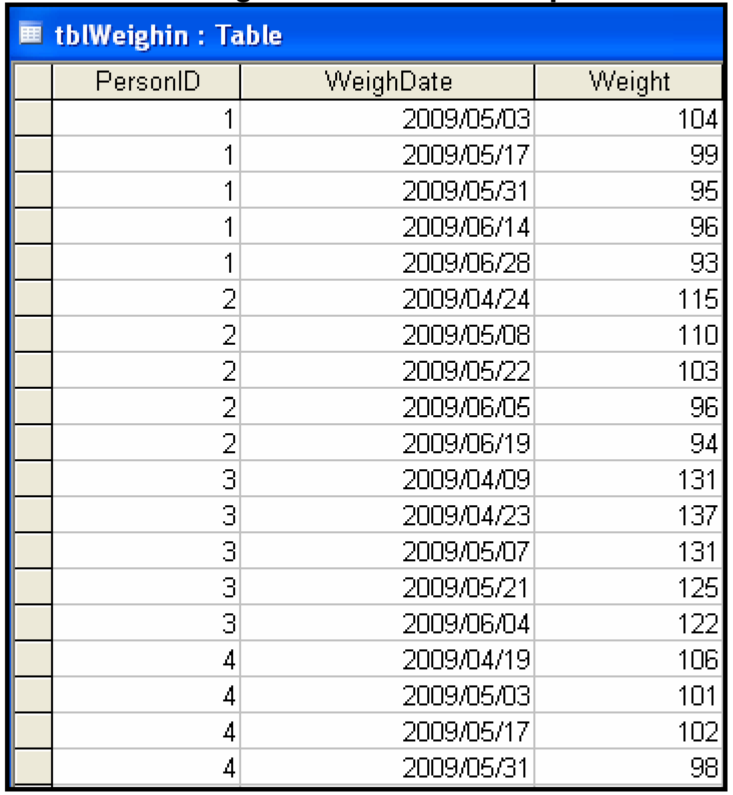
tblPeople Table Data Sample



tblPeople : Table

PersonID	Name	S_Weight	G_Weight	EntryDate	Balance
1	Teagan Gentry	109.66	70	2009/04/19	R 0.00
2	Willow Welch	122.63	80	2009/04/10	R 410.52
3	Erich Brooks	131.35	100	2009/03/26	R 0.00
4	Salvador Obrien	105.37	80	2009/04/05	R 290.89
5	Barrett Banks	152.39	110	2009/03/15	R 291.82
6	Alexandra Hancock	99.91	80	2009/03/06	R 0.00
7	Uriel Knight	110.55	90	2009/03/18	R 401.96
8	Mira Mccall	159.34	125	2009/03/14	R 335.16
9	Charissa Nichols	140.43	115	2009/03/23	R 0.00
10	Roary Spencer	108.38	85	2009/03/18	R 287.44
11	Conan Bolton	189.17	140	2009/04/03	R 0.00
12	Ila West	137.16	90	2009/03/23	R 337.63
13	Zia Jacobs	90.44	60	2009/04/13	R 123.36
14	Flavia Bauer	132.68	85	2009/03/25	R 361.43
15	Elliott Parker	118.87	90	2009/03/24	R 0.00
16	Alan Steele	165.37	100	2009/04/02	R 388.78
17	Alea Erickson	179.13	100	2009/04/04	R 199.10
18	Gail Cochran	172.25	135	2009/04/20	R 0.00
19	Claudia Rice	107.71	70	2009/03/03	R 278.30
20	April Holder	100.92	65	2009/03/29	R 0.00

tblWeighin Table Data Sample



tblWeighin : Table

PersonID	WeighDate	Weight
1	2009/05/03	104
1	2009/05/17	99
1	2009/05/31	95
1	2009/06/14	96
1	2009/06/28	93
2	2009/04/24	115
2	2009/05/08	110
2	2009/05/22	103
2	2009/06/05	96
2	2009/06/19	94
3	2009/04/09	131
3	2009/04/23	137
3	2009/05/07	131
3	2009/05/21	125
3	2009/06/04	122
4	2009/04/19	106
4	2009/05/03	101
4	2009/05/17	102
4	2009/05/31	98

Copyright reserved

ANNEXURE B: Instructions to create the database FatFighersDB

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

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

The **tblPeople** table stores data on the people who are on the Fat Fighters program. The fields in the **tblPeople** table are defined as follows:

Field Name	**Type**	**Size**	**Comment**
PersonID	AutoNumber	Longint	Unique code of the person
Name	Text	30	First name and last name
S_Weight	Number	Double	Weight at the start of the programme
G_Weight	Number	Integer	Goal weight at the end of the programme
EntryDate	Date/Time	ShortDate	Date of entry into programme
Balance	Currency		Outstanding fees

See ANNEXURE A for examples of the data contained in the **tblPeople** table.

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

Field Name	**Type**	**Size**	**Comment**
PersonID	Number	Integer	ID of the person who weighed in
WeighDate	Date/Time	ShortDate	Date that the person weighed in
Weight	Number	Double	Weight of the person that weighed in

See ANNEXURE A for examples of the data contained in the **tblWeighin** table.

ANNEXURE C: Instruction to connect to the database in Delphi

Do the following to establish a connection with the database in Delpi:

- Click on the ADOQuery component named **qryFF**.
- Click on the ellipsis button (three dots) to the right of the 'Connection' string property in the 'Object Inspector'.
- Click on the 'Build' button which takes you to the Data Link Properties dialog box.
- Select 'Microsoft Jet 4.0 OLE DB Provider' and click on 'Next'.
- The first option on the 'Connection' tab sheet allows you to browse and find the **FatFightersDB** file.
- Remove the user name 'Admin'.
- Click on the 'Test Connection' button.
- Click 'OK' on each one of the open dialog windows.

Copyright reserved

120

INFORMATION TECHNOLOGY PAPER 1

INFORMATION SHEET *(to be completed by candidates)*

EXAMINATION CENTRE ____________________

EXAMINATION NUMBER ____________________

WORK STATION NUMBER ____________________

Programming Language Used
Mark appropriate box with a cross (X)

Delphi	Java

FOLDER NAME ____________________

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

Question number	File name	Saved (*tick* ✓)	Maximum mark	Mark achieved	Marker initial/ code
1			40		
2			44		
3			36		
TOTAL			**120**		

Comment (*for official use only)*

____

____

____

____

____

____

Copyright reserved