NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2012

INFORMATION TECHNOLOGY: PAPER II

Time: 3 hours 120 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 9 pages. Please check that your question paper is complete.

2. This question paper is to be answered using Object-Oriented Programming principles. Your program must make sensible use of methods and parameters.

3. This paper is divided into two sections. All candidates must answer both sections.

4. This paper is set in programming terms that are not specific to any particular programming language (Java/Delphi) or database (Access/MySQL).

5. Make sure that you answer the questions in the manner described because marks will be awarded for your solution according to the specifications that are given in the question.

6. Only answer what is asked in each question. For example, if the question does not ask for data validation, then no marks are awarded for it, and therefore no code needs to be written.

7. If you cannot get a section of code to work, comment it out so that it will not be executed and so that you can continue with the examination. If possible, try to explain the error to aid the marker.

8. When accessing files from within your code, DO NOT use full path names of the file, as this will create problems when the program is marked on a computer other than the one you are writing on. Merely refer to the files using their names and extensions, where necessary.

9. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper. You are advised to look at the supplied data files carefully.

10. Make sure that routines such as searches, sorts and selections are developed from first principles, and that you do not use the built-in features of a programming language for any of these routines.

11. All data structures must be defined and declared by you, the programmer. You may not use components provided within the interface to store and later retrieve data.

12. Read the whole question paper before you choose a data structure. You may find that there could be an alternative method of representing the data which will be more efficient considering the questions that are asked in the paper.

13. You must save all your work regularly on the disk you have been given, or the disk space allocated to you for this examination.

14. If there is a technical interruption that prevents you from writing your examination, e.g. a power failure, when you resume writing your examinations, you will only be given the time that was remaining when the interruption began. No extra time will be given to catch up work that was not saved.

15. Make sure that your examination number appears as a comment in every program that you code as well as on every page of hard copy that you hand in.

16. Print a code listing of all the programs/classes that you code. Printing must be done after the examination. You will be given half an hour to print after the examination is finished. Your teacher will tell you what arrangements have been made for the printing of your work.

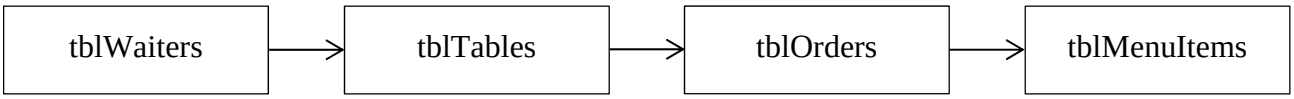
SCENARIO

Many students work as waiters in restaurants. This examination looks at a scenario where you are working in a burger restaurant. Your manager would like you to help her make better use of their computer systems to run the restaurant more efficiently. The skills that you learned in IT at school will come in useful.

SECTION A STRUCTURED QUERY LANGUAGE

The first task is to examine some of the output from the POS (point of sale) system so that the manager can see which waiters are performing well and which are not. The database is in the files that were provided for you for this examination. It is called **Waiters**.

There are four tables present in the database to store information about the waiters (tblWaiters), the tables of people that eat in the restaurant (tblTables), the orders that those tables place (tblOrders), and the menu items that they order (tblMenuItems). The following diagram illustrates this relationship between the tables:



The fields used in the database are discussed below. Below each description is a screen-shot of the first 10 rows of data for your convenience. The tables do contain more data.

tblWaiters

waiterID	This is an automatic numbering field which assigns each waiter a unique ID number.
waiterName	This field contains the names of the waiters.
waiterPhone	This field contains the telephone contact numbers of the waiters.

tblWaiters

waiterID	waiterName	waiterPhone
1	Erin	083 276 2000
2	Matthew	071 584 3000
3	Nomfundo	087 654 3000
4	Simphiwe	072 111 2000
5	Hayden	079 073 0000
6	Ruth	078 765 5000
7	Kwame	082 210 0000
8	Morgan	076 017 4000
9	Vashni	077 517 8000
10	Ramola	085 551 2000

tblTables

tableID	This field contains the number of each table.
tableWaiterID	This field contains the ID number of the waiter who served the table. This key is a foreign key for the **tblWaiters** table.
tableGuests	This field contains the number of guests who were dining at the table.
tableAmountPaid	This field contains the amount of money that was paid by the guests for their meal. This value includes the tip that was paid.

tblTables

tableID	tableWaiterID	tableGuests	tableAmountPaid
1	5	7	450.00
2	10	2	250.00
3	3	6	480.00
4	7	7	680.00
5	9	7	710.00
6	5	10	620.00
7	2	7	860.00
8	1	11	880.00
9	3	12	750.00
10	6	1	90.00

tblOrders

orderTableID	This field contains the table number that the order is destined for. This field is a foreign key for the **tblTables** table.
orderMenuItemID	This field contains the ID number of the menu item that was ordered. This field is a foreign key for the **tblMenuItems** table.
orderQuantity	This field contains the quantity of the menu item that was ordered.

tblOrders

orderTableID	orderMenuItemID	orderQuantity
1	1	3
1	3	1
1	7	1
1	11	4
1	13	1
1	14	2
1	15	1
1	17	1
1	21	2
2	1	3

tblMenuItems

menuItemID	This field contains a unique numeric value for each item on the menu.
menuDescription	This field contains a description of the menu item.
menuCategory	This field contains either the value 'Food' or 'Drinks'. It is used to classify each menu item.
menuSalesPrice	This field contains the selling price of each menu item. This is the price that appears on the menu and which the customers must pay.
menuCostPrice	This field contains the actual cost of the menu item. This includes the cost of the ingredients, the labour to produce the item and the electricity and water that were used to make it.

tblMenuItems

menuItemID	menuDescription	menuCategory	menuSalesPrice	menuCostPrice
1	Hamburger and Chips	Food	34.95	23.20
2	Cheese Burger and Chips	Food	39.95	24.72
3	Bacon Burger and Chips	Food	44.95	29.36
4	Veggie Burger and Chips	Food	32.95	22.23
5	Hamburger	Food	29.95	19.96
6	Cheese Burger	Food	34.95	20.30
7	Bacon Burger	Food	39.95	25.21
8	Veggie Burger	Food	29.95	18.55
9	Chips - Large Plate	Food	14.95	5.30
10	Cola	Drinks	8.99	4.53

1.1 Write a query that will list all the waiter information, in alphabetical order by the waiters' names. (3)

1.2 Write a query that will list the table numbers of all the tables who had either only 1 guest, or 10 or more guests at the table. Your results should list only the table number and the number of guests. (3)

1.3 Write a query that will list the descriptions of any menu items that include 'Chips' as part of their descriptions. (3)

1.4 Write a query that will show the amount of profit earned on each drink item on the menu. Show only the name of the menu item and the profit earned on that item. Ensure that the calculated field is given an appropriate heading. (3)

1.5 Write a query that will insert a new waiter, Busi, into **tblWaiters**. Her cell number is 083 469 9000. Allow the database to automatically determine her **waiterID**. (3)

1.6 Percentage mark-up is calculated with the following formula:

$$P = \left(\frac{s}{c} - 1 \right) \times 100$$

where: P is the percentage mark-up
s is the sales price
c is the cost price

Write a query that will list the menu item descriptions and percentage mark-ups in descending order with the items having the highest percentage mark-up first and the lowest mark-up last. (5)

1.7 Write a query that will display how many of each menu item in the **tblMenuItems** table has been sold. Your results should display only the name of the menu item and the number of those items that have been sold. (5)

1.8 Write a query that will show how many tables each one of the waiters has served and the average amount that each table paid, including their tip. Your results must show the waiters' names, the number of tables, and the average amount taken from each table. (7)

1.9 The restaurant is running a competition amongst the waiters. Any waiter who sells a 'Giant Burger' or 'Giant Burger with Chips' will get a R10 reward for each meal sold. Write a query to calculate how much each waiter is owed from this promotion. Your query should show only the waiters' names and the amount that they have won. (8)

40 marks

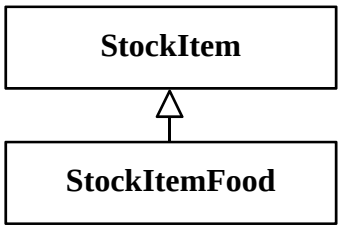
SECTION B OBJECT ORIENTED PROGRAMMING

The restaurant has to take care that all the stock that it uses is accounted for. Presently, there is no way to track the stock. If the stock is tracked carefully, it will help with the re-ordering of stock items that are running low and which might run out. The owners of the restaurant would like you to write a simple stock management system to help them keep better control of their restaurant stock and their ordering.

The stock items fall into two categories: the first are standard stock items that should remain constant. These are things like plates, knives and forks which are not used up in the making of the food. However, their stock can decrease as they either break or get lost. A stocktake is done periodically to see whether these items need to be re-ordered.

The second category of stock items is used in the preparation of food. Each of these food items has a certain minimum level that must be maintained so that the kitchen does not run out of the ingredients it needs to make the food. If the amount of stock drops below the minimum level, an order must be placed for more stock. The store rooms and fridges in the restaurant only have a certain capacity, and so the restaurant can only keep a certain amount of these items at any time.

After some consideration, you decide on the following class structure:



You are given a file called **StockList.txt** which contains a list of stock items. The first five lines of that file are provided here for your reference (the actual file contains many more lines and you are advised to examine it carefully).

```
Plates#76#plates
Salt#7.6#kg#1#10
Whole-wheat bread rolls#15#rolls#24#48
Knives#105#knives
Tomatoes#5.3#kg#1#7.5
```

The lines in the file either have three or five fields present.

The first three fields on all lines indicate the name of the stock item, the quantity currently in stock and the unit that the item is measured in. For example, for the stock item 'Plates', there are 76 plates in stock. 'Salt' is measured in 'kg' and there are 7.6 kg in stock. The units are used for display purposes only.

Some lines have two additional fields (to make five in total). These two fields represent the minimum stock on hand before an order is placed, and the maximum stock that can be on hand at one time.

For example, on the third line of the text file, we see that the restaurant currently has 15 'Whole-wheat bread rolls' in stock. We need to order more because the minimum level required is 24. The maximum level is 48 which means we must order (48 – 15) = 33 bread rolls to return the stock to its maximum level. We only need to place an order if the current level is below the minimum required level.

A second file, **stockusage.txt**, is also provided and its purpose and contents are explained later in Question 6.

QUESTION 2

Use the class diagram below to create a new class called **StockItem**. This class will be used to create objects that will store the details about the stock levels of the different items of stock. The diagram below indicates the properties and methods that are required.

StockItem
Properties: - String description - double quantity - String units
Methods: + Constructor (String newDescription, double newQuantity, String newUnit) + getDescription (): String + getQuantity (): double + getUnits (): String + setQuantity (double newQuantity) + changeQuantity (double difference) + toString (): String

The **changeQuantity** method should *increase* the overall quantity of the stock item by the amount that was passed through in the parameter. Therefore, if there are 10 units of an item in stock, and the **changeQuantity** method is called with a parameter of 3, the **quantity** property should be increased to 10 + 3 = 13. If a negative amount is put in as a parameter, the quantity will *decrease*. For example 10 + (–6) = 4.

The **toString** method should return a String comprised of the **description**, the **quantity** and the **unit**. It must be formatted as follows:

```
<description>: <quantity> <units>
```

for example: `Mayonnaise: 7.0 litres`

[13]

QUESTION 3

3.1 Write code to create a new class called **StockItemFood** which extends the **StockItem** class. (1)

3.2 Write code to create two properties that will store the minimum acceptable level of stock allowed in the restaurant and the maximum allowed level of stock. Choose an appropriate data type for these properties. These values should not be visible from outside the class. (3)

3.3 Write code to create a constructor method that will initialise both of these properties in addition to the parent class's properties. (4)

3.4 Write code to create a method called **mustOrder** that returns a Boolean value of **true** if the quantity in stock of the item is below the minimum level. Otherwise, the method should return the value **false**. (4)

3.5 Write code to create a method called **getOrderAmount** that will determine the amount of the stock that must be ordered to reach the maximum stock level. (3)

[15]

QUESTION 4

4.1 Write code to create a class called **StockManager**. (1)

4.2 Write code to declare instance variables in the class of an array that can be used to store up to 100 different **StockItem** objects, and an integer counter to keep track of how many stock items are in the array. These two instance variables should not be accessible outside this class. (4)

4.3 Write code to create a constructor method that will read the contents of the file called **stocklist.txt**. The file contains information for both normal stock items and those that have ordering levels. Read in each line from the file and instantiate the appropriate object (one of **StockItem** or **StockItemFood**) and append it to the array. (10)

4.4 Write code to create a method called **getStockList**. This method must return a String that contains all the stock items and their current levels of stock. Each stock item should appear on its own line. Your solution must make use of the **toString** methods that you have already created in the questions above. (7)

4.5 Write code to create a method called **getOrderingList**. This method must return a String that contains all the stock items that need to be ordered, based on their stock levels. This list must show the amount of stock that needs to be ordered and not the current stock levels. It should be formatted in a similar way to the **toString** method. For example, the line in the list for 'Whole-wheat bread rolls' would display

```
Whole-wheat bread rolls: 33 rolls
```

since 33 rolls need to be ordered. Do not change the **toString** method in the **StockItem** class. You still need to use that method to show the current stock amounts. (8)

4.6 Write code to create a method called **findStockItem**. This method must accept a String as a parameter (the stock description to search for) and return a **StockItem** object with a matching description. If no object can be found with a matching description, the method must return a **null/nil** value. (5)

[35]

QUESTION 5

5.1 Write code to create a class called **RestaurantDriver** which will contain the code for the user interface. (1)

5.2 Write code to declare a **StockManager** object and instantiate it at an appropriate place in the code. (1)

5.3 Write code that will display, with an appropriate heading above each list, the current stock list and the current order list. (3)

[5]

QUESTION 6

6.1 You are also provided with a file called **StockUsage.txt**. This file contains the results of a stocktake that was done in the store.

The first six lines of that file are provided here and explained below:

```
Patties - Chicken: 120
Salt: used 1.3
Mushrooms: bought 15
Knives: 115
Patties - Beef: used 120
Tomato Sauce: used 4.5
```

Each line has a description of a stock item followed by a colon and then one of three things which can describe the stock changes.

- The first line is an example of where the current stock level is given by indicating only a single number. In this case, the stock level should be set to this amount. On the first line, we are told that there are 120 chicken patties in stock.
- The second line is an example of where we have **used** a certain amount of stock and so the level needs to be reduced by this amount. In this case, the stock of salt needs to be reduced by 1.3.
- The third line is an example of where we have **bought** new stock. The stock level needs to be increased by this amount. In this case, the stock of mushrooms needs to be increased by 15.

Notes:

- The units of each stock item ('kg', 'patties', etc.) are not given in this file but they match the units that were used before. No conversion between units is needed.
- The stock items in the **StockUsage.txt** file are not in the same order as in the previous file and some stock items may have more than one entry in this file.

Write code to create an appropriate method in the **StockManager** class that will use the data present in the **StockUsage.txt** file and adjust the stock levels accordingly. (10)

6.2 Add code in the **RestaurantDriver** program created in Question 5 after the initial stock list and ordering list are displayed, to perform a stocktake as programmed in Question 6.1. Once the stocktake has been performed, display the updated stock list and order list. (2)

[12]

80 marks

Total: 120 marks