IND270 Programming Assignment – Part A

Drink Vending Machine Simulation

**Introduction**

A new drink vending machine program is being designed and you have been contracted to write a C# console application that models the proposed functionality of this system.

**Initial Specification**

You are required to write a program written in C#, which will model the functions of the Drink Vending Machine.

The program will display a list of possible actions, read in the user’s choice and then perform the required action. Unlike a real vending machine, the user will be able to select any action from the displayed menu and the program will handle the situation.

Here is a very high-level algorithm of the assignment design:

***While not finished***

***Display list of possible actions***

***Input the user’s option***

***Perform the required action***

***End while***

**Assignment Specification**

A number of assumptions have been made so that the assignment can be completed using only the concepts covered in Lessons 1 to 4. You are free to use any valid programming constructs even if they have not been covered in lessons so far.

Our model of a Drink Vending Machine will permit the user to choose from the following main menu:

* Select a drink
* Insert a coin
* Obtain a refund of coins inserted
* Collect the drink and any change if necessary
* Exit the simulation

See the sample screen interactions at the end of this document for further details.

All input should be checked that it is within the acceptable range. If input is not within the acceptable range, then the user should be informed of the range and requested to re-enter the input.

When a user chooses an action, it will affect the state of the Drink Vending Machine as follows:

* **Select a drink (SelectProduct)**
  + A choice of three types of drinks will be displayed
    - Soft drink
    - Milk drink
    - Fruit juice
  + The user will select one type of drink or choose to cancel this selection
  + The cost associated with each drink is:
    - Soft drink $1.50
    - Milk drink $3.00
    - Fruit juice $2.20
* **Insert a coin (InsertCoin)**
  + If the user has selected a drink, then they will be asked to insert coins to the value of the drink. Coins will be accepted until the total amount entered equals or exceeds the value of the drink or the user decides to cancel entering coins.
  + The denominations of acceptable coins are displayed and the user is prompted to enter the coin value or to select 0 to cancel from this action
  + Only $2, $1, 50¢, 20¢ and 10¢ will be accepted
* **Obtain a refund (RefundCoins)**
  + If coins have been entered and a drink has not been issued, then the full value of coins entered is refunded
  + This will also be used when a drink has been issued and there is change to be refunded to the user
* **Collect the drink (CollectProduct)**
  + Providing that a drink has been selected and the amount of coins entered is enough to cover the cost of the drink, the drink is issued and any excess value of coins is refunded
  + Appropriate error messages are issued to cover the cases of
    - No drink had been selected
    - Not enough coins inserted to cover the cost of the drink
* **Exit the simulation**
  + If coins have been entered, then refund the coins entered before terminating the simulation

Some initial work had already been done on this project and the following specification of various methods has been developed and agreed upon with the client. You are required to use these methods as well as three class variables and one class constant. You can include additional class variables or methods if desired.

Class variables

Class variables should be declared at the beginning of the code. Place these after the class declaration but before your first method declaration:

double amountEntered // total value of coins entered so far

double drinkCost // the cost of the selected drink

bool drinkSelected // flag to indicate whether user has selected a drink

const int EXIT = 5

These can then be accessed from anywhere within the body of the class.

Methods

**void PerformOption(int option)**

Action: Option corresponds to a valid choice from the main menu, and the corresponding menu option is called

**void InsertCoin()**

Action: User has selected a drink and the user enters coins to the value of the drink or the user decides to cancel inserting coins

**double GetCoin()**

Action: Returns a valid coin value or 0

**void SelectProduct()**

Action: User selects a product

**void RefundCoins()**

Action: User obtains a refund or receives change.

**void CollectProduct()**

Action: User collects the drink

**Assumptions**

We assume the following for our simulation:

* The user will input data values of the correct type when requested to do so. That is, if an integer is expected, an integer value is entered; if it is a double that a double is entered. However the correct type does not imply that the value entered is acceptable and valid for that input.
* The vending machine has an endless supply of all three varieties of drink
* The vending machine has an endless supply of all coins for refund purposes

**Assignment Goals**

The solution of the problem will be a C# project which will use, at least, the structured programming constructs covered in Lessons 1 to 4.

Though the code could be written as *straight-line code,* the number of lines would make the solution difficult to read, almost impossible to understand, very hard to maintain and more than likely to contain many potential logical errors. A straight-line code solution will not receive full marks **even if** it fulfils the required functionality!

It is highly recommend that you use the methods specified above as well as few of your own design to manage the complexity as well as providing a structure to your solution. As a hint, each independent action or process should be a method. The body of the *Main* method should resemble the high-level algorithm given above with perhaps some minor additions and other methods would perform the required processing.

Sub-goals of the assignment are to experience:

* Top-down design & development
* Translating simple algorithms into C# code
* Writing user-defined methods
* Parameter passing
* Editing, compiling, building and running your program
* Testing your program
* Commenting your source code
* Producing a well documented solution

**Submission Details**

You are required to hand in two items.

1. The **first item** is the project folder containing your Visual Studio project for your simulation.

In your code, it is expected that:

* It conforms to the C# Coding Style Guide
* There is a class header comment
* All methods in the class will have a method header comment using either block style comments or XML style of comments
* There are inline comments where necessary

1. The **second item** is a Statement of Completeness.

A **Statement of Completeness** is a description in your own words of any issues relating to the state of your assignment - in other words what have you managed to complete, get working, and tested. It must include details of unresolved problems (like logical or run-time errors) if they exist and include information about assumptions made by you regarding the implementation of this assignment. If there are no additional assumptions or deviations from the specification, then you must state this fact.

It is expected that most people will implement all of the functionality of this assignment, have tested it adequately according to comprehensive test plans and are confident that the program behaves according to the given specifications.

The Statement of Completeness needs to be clear and concise - no more than two or three paragraphs.

You need to submit a zip file to Blackboard Assessment by Friday, Week 7. The zip file should contain the folder for your Visual Studio project and your Statement of Completeness. The zip file is to be named **YourName\_VendingMachine.zip**.

**Suggested Strategy**

Plan to be finished before the due date! Do not write all of the code in one sitting and then attempt a compilation.

Here is a suggested plan of attack:

First few days:

Read the assignment specification

Clarify anything unclear with the teacher

Think about how to do it

Develop test plan and cases

Design high-level algorithms for each independent part of the assignment

Begin to type program (with comments), in incremental stages. For example – ensure that you can correctly read in the required action first and then display the price of a selected car type before adding code to purchase or exchange cars. Seek help from the teacher if needed.

Next week:

Re-read the specification

Continue to refine the design of various sections to code

Enhance the test plan if necessary

Bring code for critical review by the teacher during consultation times or practicals

Finish initial coding

Week after:

Debug using a test plan (add more to the test plan if extra problems show up by testing)

Perhaps another review

Re-read the specification to make sure you have done exactly what is required

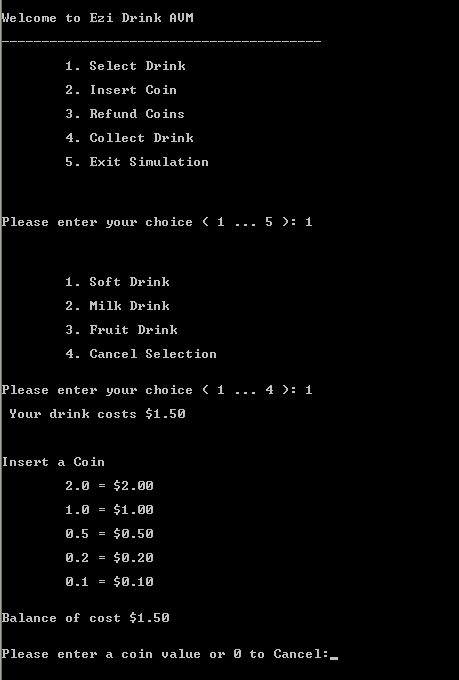
Submit assignment via Blackboard

**Final Comments**

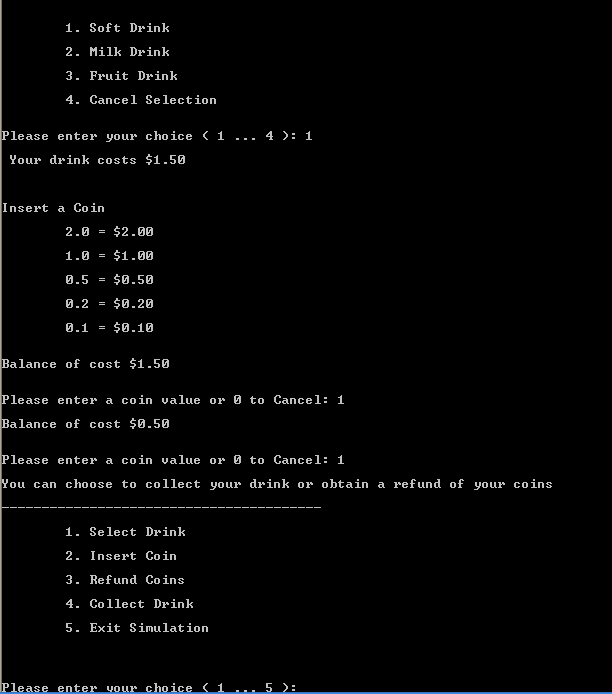
This assignment is **not** about screen design. Although you are free to alter and enhance the output statements, it will not gain any additional marks.

The following pages are examples of the required basic screen interaction – note that not all possible interactions have been displayed.

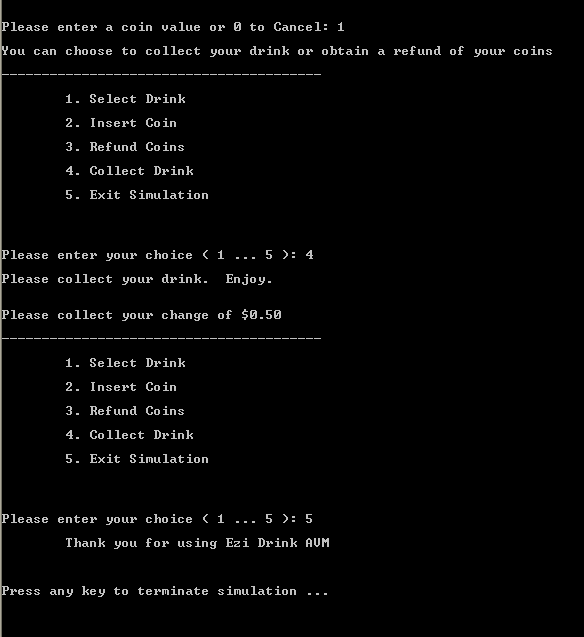
**Sample screen interactions**



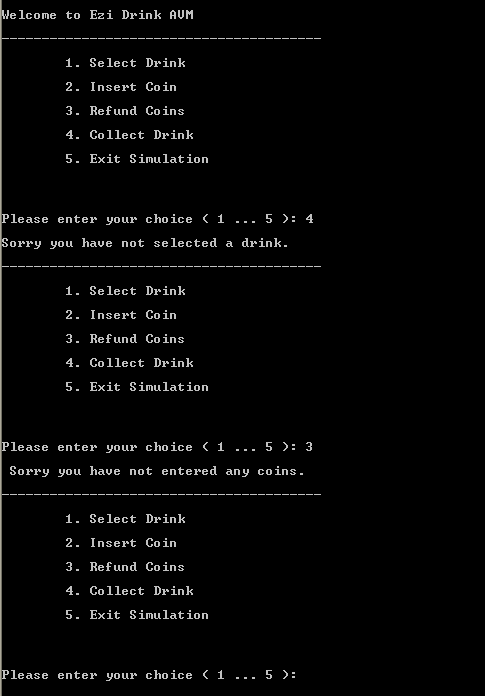
*Figure 1: Initial menu is displayed. User selects a soft drink which costs $1.50 and then is requested to enter a coin value or 0 to cancel.*



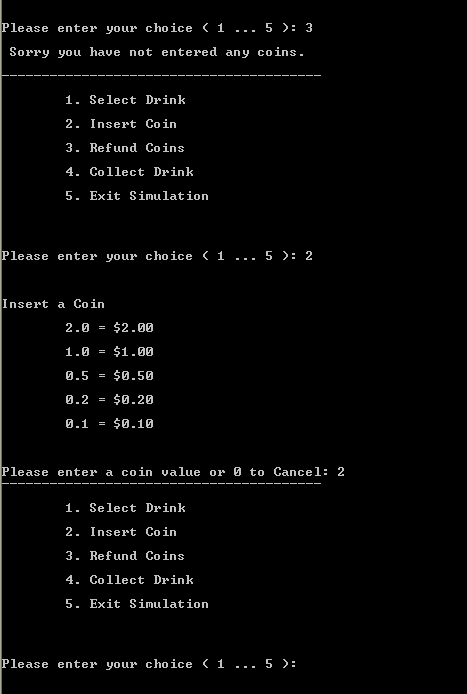
*Figure 2: User enters $1 followed by another $1 and the main menu is redisplayed.*



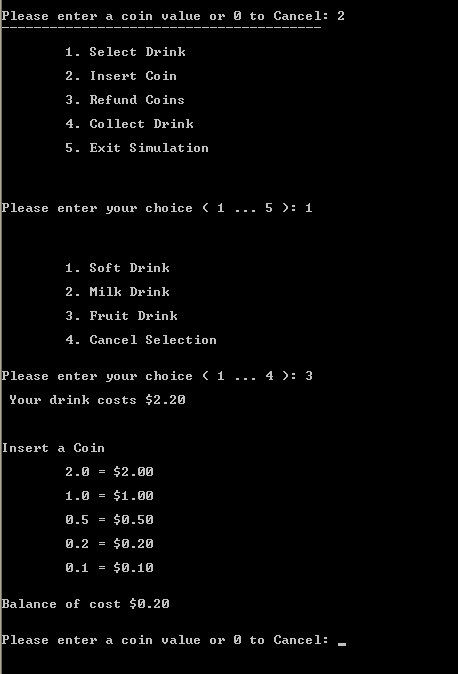
*Figure 3: User chooses to collect the drink and is informed to collect the change. User then decides to exit the simulation.*



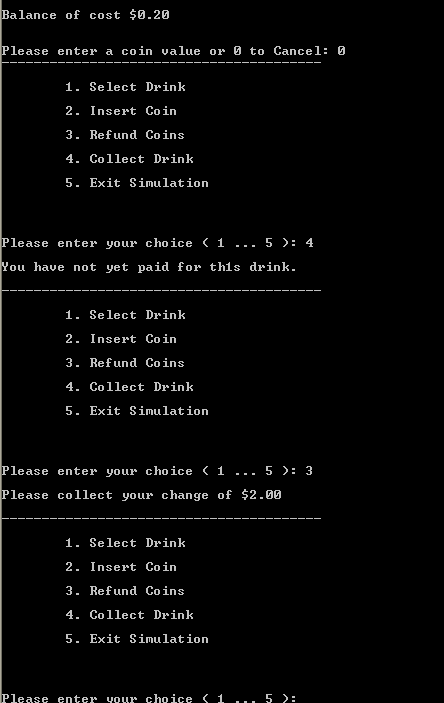
*Figure 4: User attempts to collect a drink before having selected one. User then attempts to obtain a refund though they have not entered any coins.*



*Figure 5: User selects to enter a coin (in this case, $2).*



*Figure 6: User chooses a fruit drink and is informed that a further 20 cents is required.*



*Figure 7: User does not enter another coin but attempts to collect the drink and is informed that they have not paid for the drink. User decides to obtain a refund.*