Data Structures and Algorithms

FINAL PROJECT REPORT

ONLINE SUPERMARKET SYSTEM



Girindra Ado - 2201843506

Robert Reden – 2201816612

Even semester 2018/2019

Computer Science Program

Binus University International

**Table of Contents**

I. Project Specification 2

A. Problem Description 2

B. Motivation 2

C. Aims 2

D. Project Rundown 3

E. Data Structure Implementation 3

II. Solution Design 4

A. UML Diagram 4

A. Code 5-10

III. Program Explanation 11

A. Program Manual 11-19

A. Time Complexity 19

A. Program and Demo Link 19

**I. Project Specification**

**A. Problem Description**

In modern times like these, online supermarkets aren’t as popular as it should be and people still drive all the way to the supermarket to buy their daily necessities which we find to be inefficient, time consuming, and quite a chore. Why trouble yourself with the extra work and gas bills when you can just sit comfortably at home, open your phone and order all the things you need.

**B. Motivation**

We chose supermarket billing system as our topic to work on because we think that online supermarkets will be very popular in the lifestyle of current and future generations. We hope that our solution will provide ease and efficiency in time and effort in buying products that you consume/use on a daily basis.

**C. Aims**

We aim to make a program to simulate supermarkets, where it will be able to display the products sold and its quantity and price including a billing system. It will also have a login/register system for the user/customer. We will try to make it as realistic and professional as possible.

**D. Project Rundown**

We tried to make the project as professional as possible, and to make it as close to our initial projected view on it. The end result is something that we are proud of making. It has all the functions that we wanted to make, and also very user friendly too.

Our project has 9 files in total, 5 cpp files and 4 header files:

We separated the declaration and the implementation of the program so that functions implemented in the cpp file are already known by the compiler and to make it more professional.

supermarket.cpp and supermarket.h contains the supermarket’s functions, to set the default items and the list of users in the supermarket. Items.cpp and Items.h contains the setter and getter methods for the item’s attributes and functions to alter its quantity. Cart.cpp and Cart.h contains the functions for the cart, such as adding items, removing items, view the contents of the cart, the billing function, etc. the last file is supermarketmain.cpp which is the main file used to run the program. smuser.cpp and smuser.h  contains the setter and getter methods for the user’s attributes and its constructor.

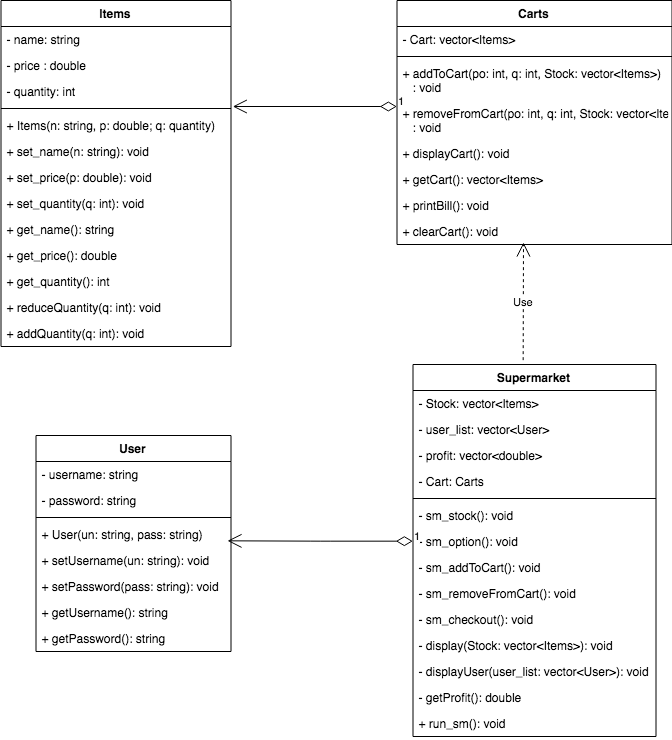
**E. Data Structure Implementation**

In our project, we implement OOP by using vector STL data structure. The vector STL suits our need within the project’s scope. We use the vector STL as a container to contain items in the supermarket and to store them into the cart.

We chose to use the vector STL because it is flexible, and stores data dynamically and provide easy access to the data as opposed to other STLs that doesn’t quite meet our requirements, such as array or linked list. Although alternatively, instead of using vector we could probably use the list STL, because it works similarly to vector.

**II. Solution Design**

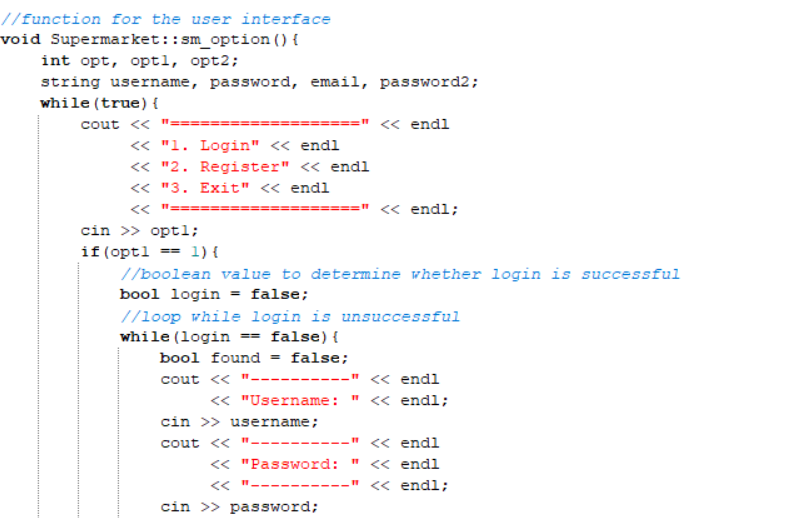
**A. UML Diagram**

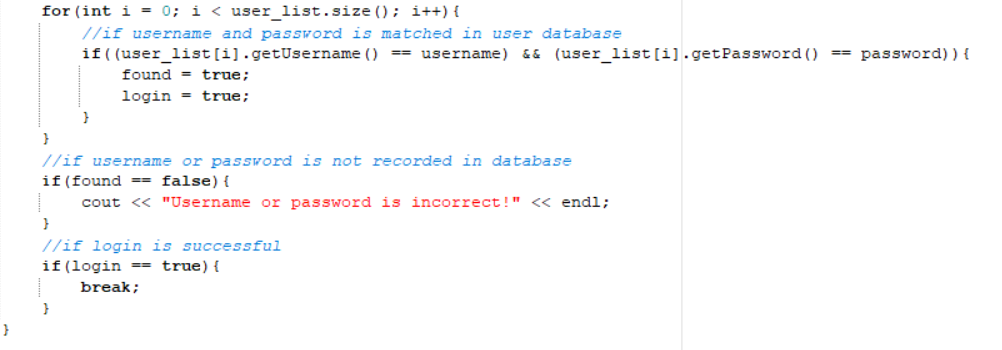
****

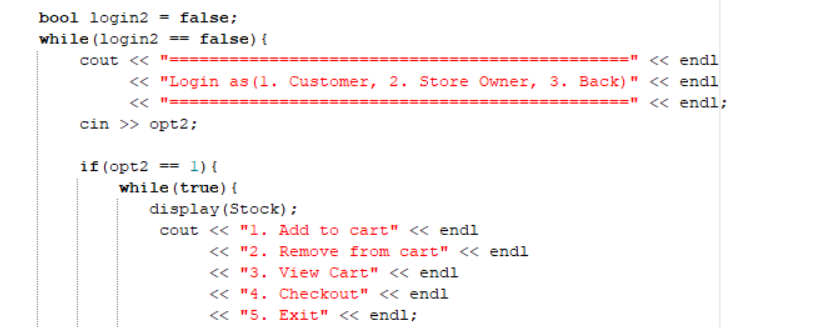
**B. Code**

After some suggestions made by our lecturer, we added some new features for our supermarket system. Namely, the addition of account creation, admin accessibility, and the ability to access the current stock of the supermarket and profits from recent purchases. For account creation, the user may be able to register one in the first window, or if one doesn’t feel like making a new account, there are stock accounts that we made in a txt. File already. If you do decide to create one, your account will be stored in said txt. File. We also have a txt. File that stores our supermarket stocks. By default everything has a quantity of 10. If you logged in as store owner/admin, you have the ability to access this stock and the store’s total profit for this session.

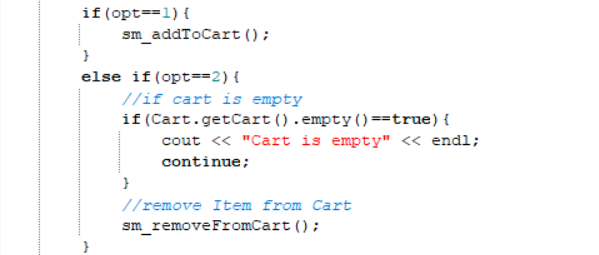
We added the option to login or register at the very first window. If user wants to shop, then they have to login first, and then login as customer to access the shop. The program stores usernames and passwords in the user database. It checks whether or not the user’s input for the username or password is already stored in the database. If not then the program asks the user to input a correct username/password.

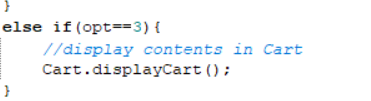
This is the UI that shows the option to login, register, or exit. It’s stored in a while loop so that the user may access it again if the user decides to exit the supermarket later. It uses a Boolean function to check if the login is successful. If not, then the program prompts the user to fill in the username and password again.

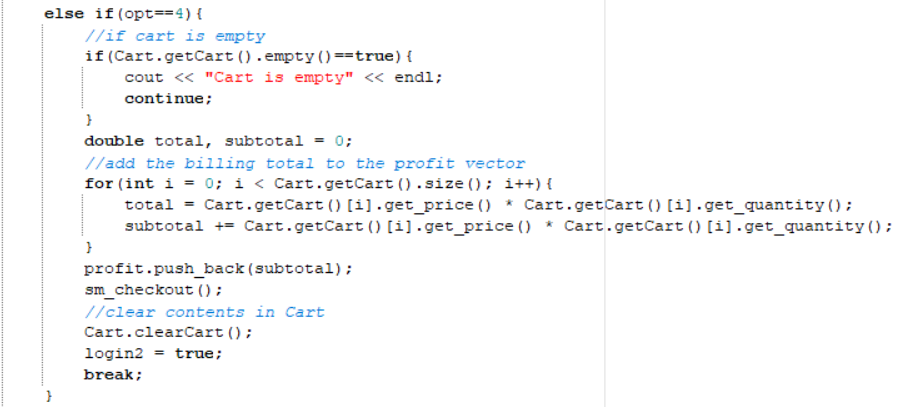
The following function checks if the username and password is correct. If they are, then the login function stops and the program shows the following user interface for the user to choose to either login as customer or store owner/admin:



If the user decides to login as customer, then the program gives options to add to/remove from cart, view cart, and checkout

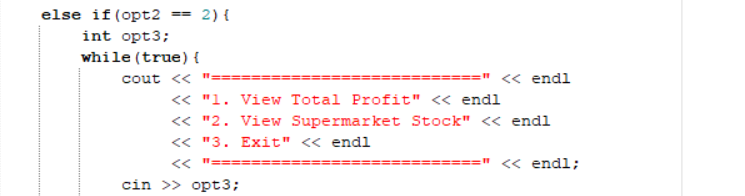
If the user wants to add an item to the cart, then the program runs the addToCart function. If the user wants to remove an item from the cart, then the program will run the removeFromCart function, else if the cart is empty then the program prints out a notification stating the cart is empty.

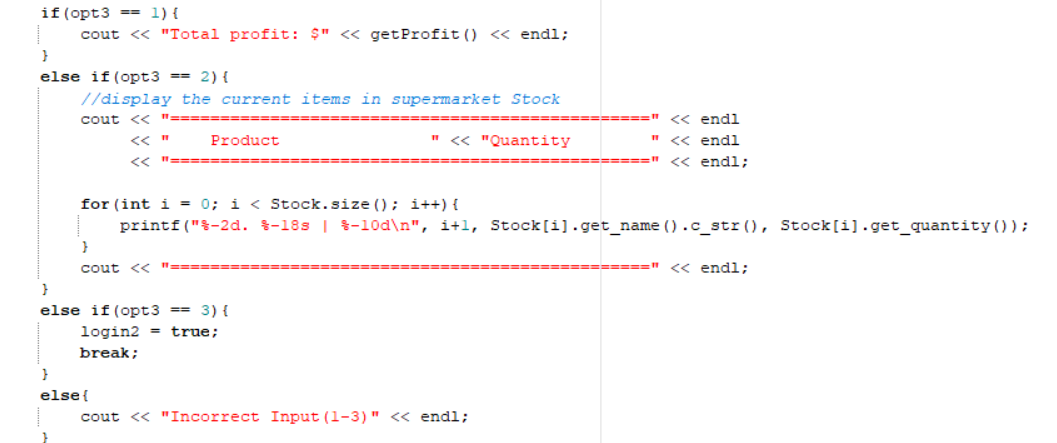


If the user wants to see the content of the cart then the program runs the displayCart function.  
  
  
  
This is if the user wants to check out and pay for the items in the cart. The total value of the items will then be stored in a vector that can later be accessed as store owner to see the total profit of the supermarket in this session. At the end of the function, the contents of the cart are cleared.

All the functions that’s mentioned above will be elaborated later in the report.

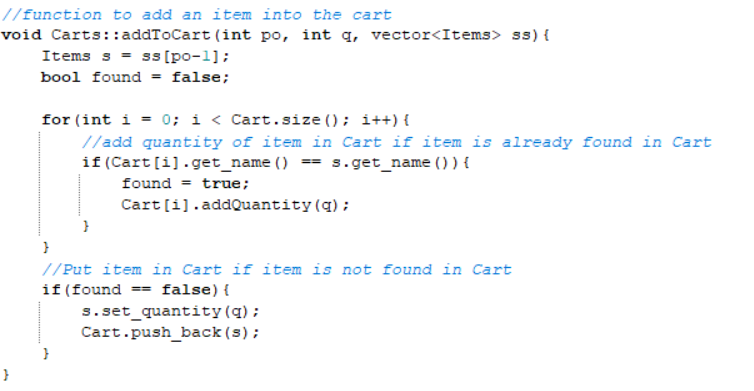
If the user logs in as store owner instead on customer, then the program will then prompt the user to view total profit or see the supermarket stock.





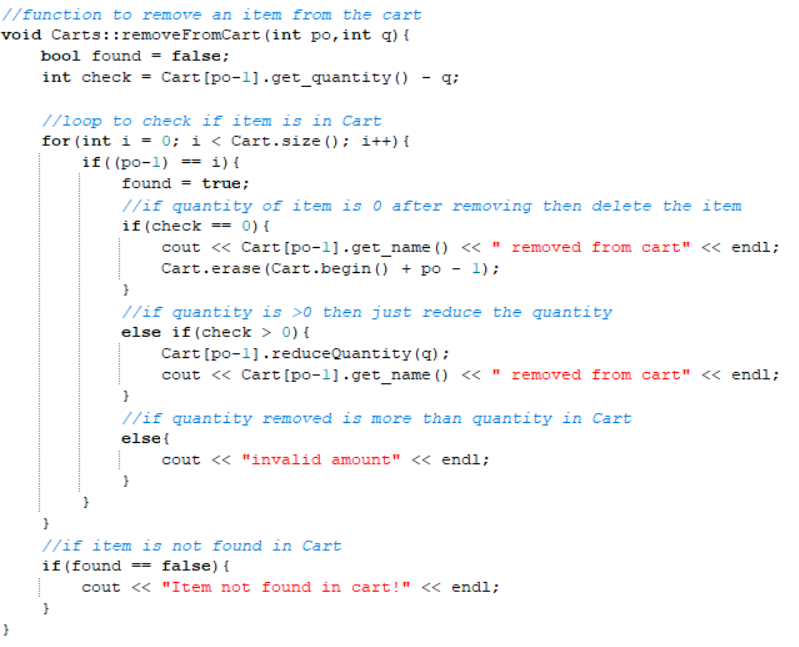
If the store owner wants so see total profit, then the program runs the getProfit function. If they want to check out the stock, then the program will show a new UI that also prints out the current stocks by using calculation to subtract any items that’s been sold in this session with the default amount, which is 10.

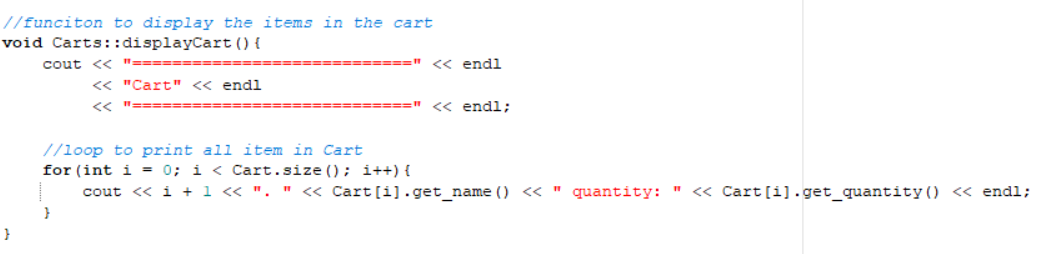
**Cart Functions**

The Cart class has functions essential to the supermarket billing system. Such as the aforementioned addToCart function, in which the program checks if the amount of items inputted has already been put in the cart. The items will be put if after the program checks, the item is not in the cart already.

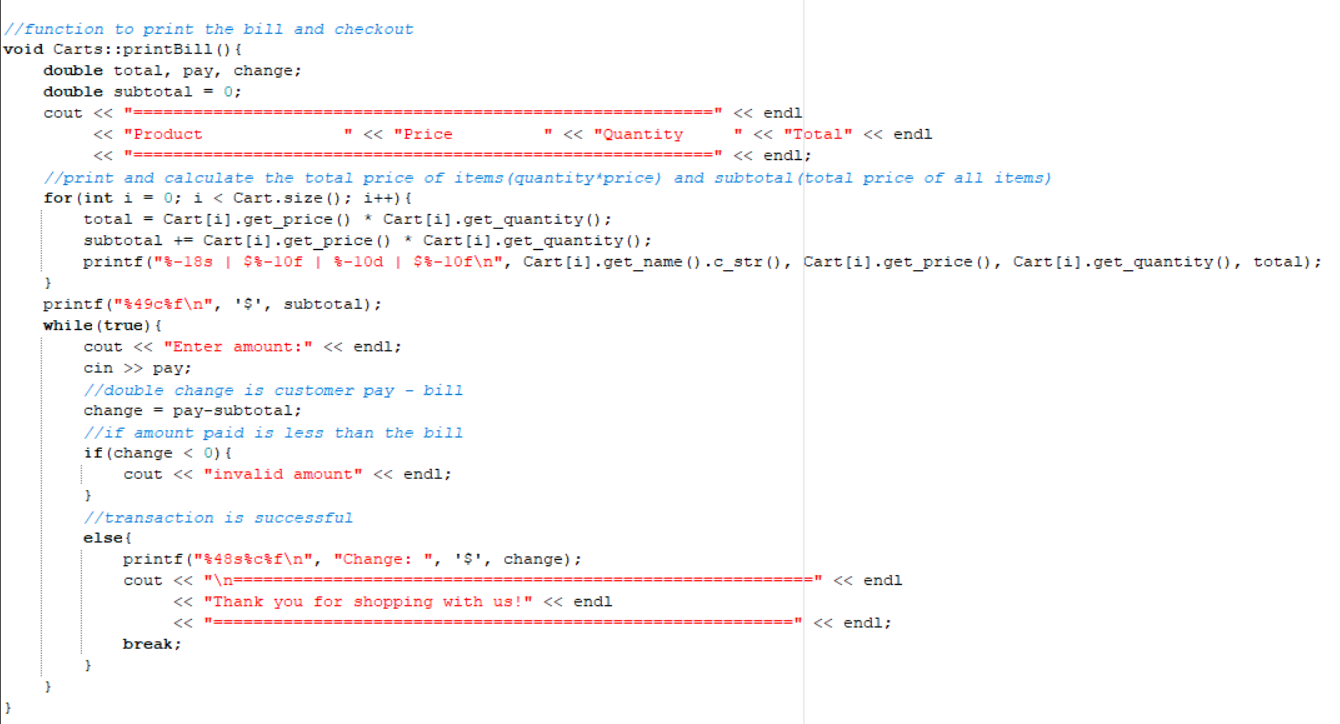
There’s also the removeFromCart function. The function checks if the item that is to be removed is in the cart or not.

The program will delete the item entirely if after the subtraction the item quantity is below 0, and if not then the function just normally subtract the item.

The program will tell you an invalid amount if you input an integer more than the actual quantity of items in the cart.



The above function is to display items in the cart. The program will run through a loop that prints out everything in the cart alongside their respective quantity. Quantity (integer) is an object that is created in the Item class, alongside price (double) and name (string).

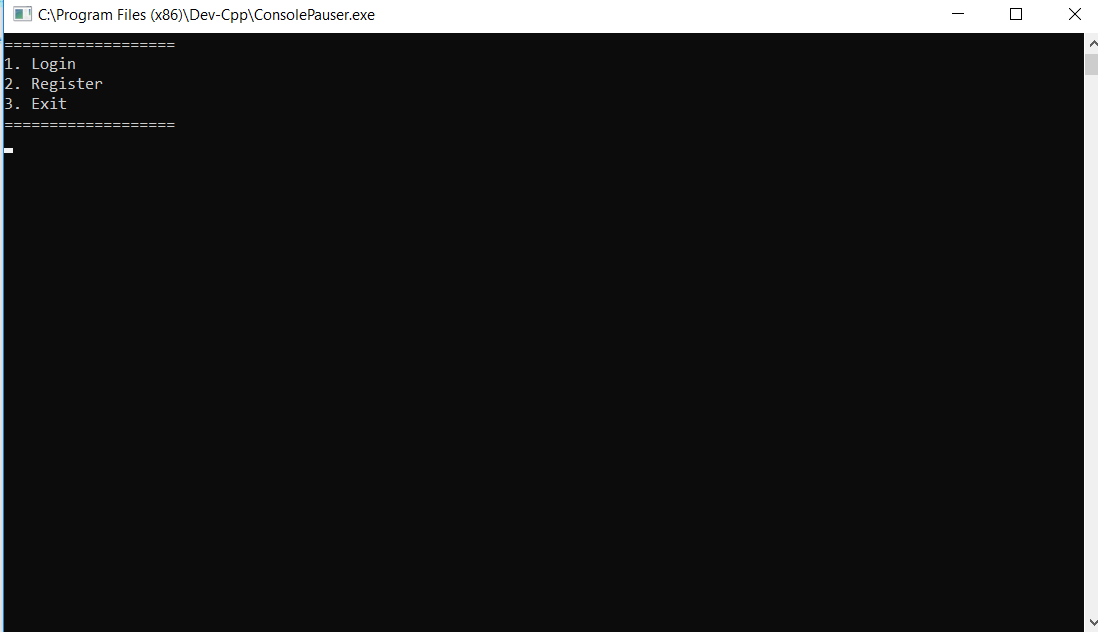


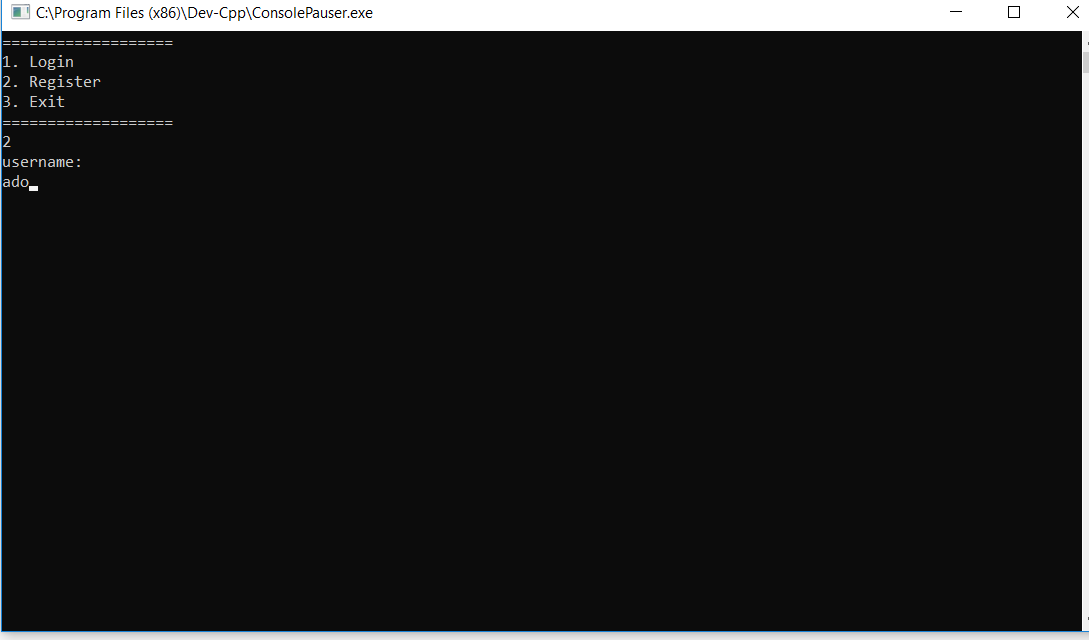
The above function is for the checkout and payment of the supermarket. It has the variables total, pay, and change, and subtotal which by default is 0. The calculation is the quantity times the price of the items in the cart. Every items in the cart will be added to count the new subtotal. The user will be required to input a selected amount of money to pay the bill. The program will calculate the amount entered. If it’s fewer than the subtotal, the customer will be required to input more amount. If it’s not more than the subtotal, then the program will subtract the amount entered with the subtotal to give you your change.

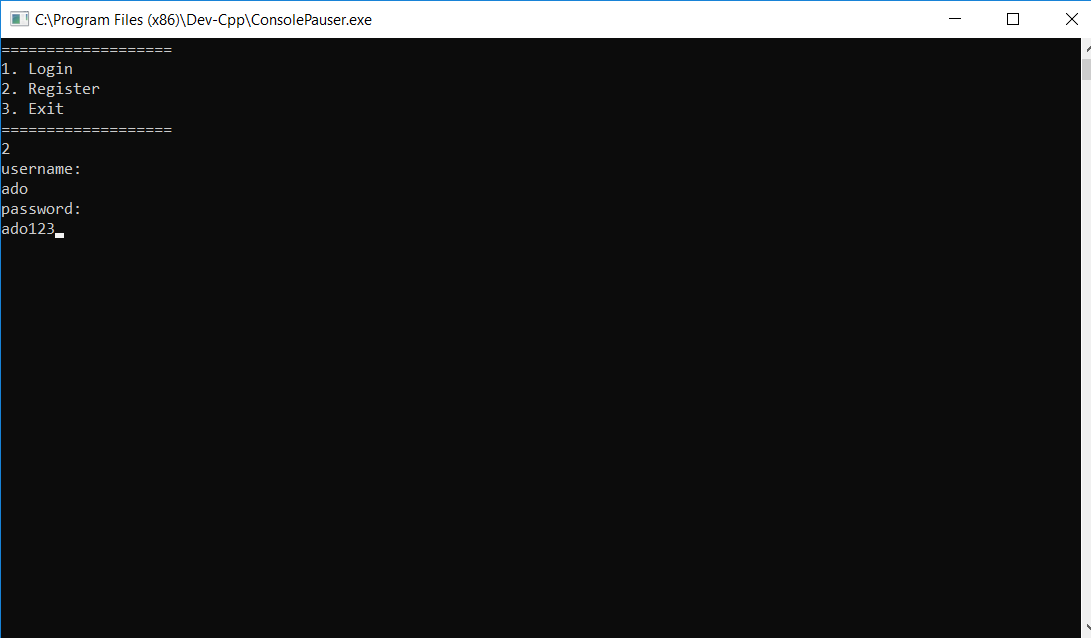
**III. Program Explanation**

**A. Program Manual**

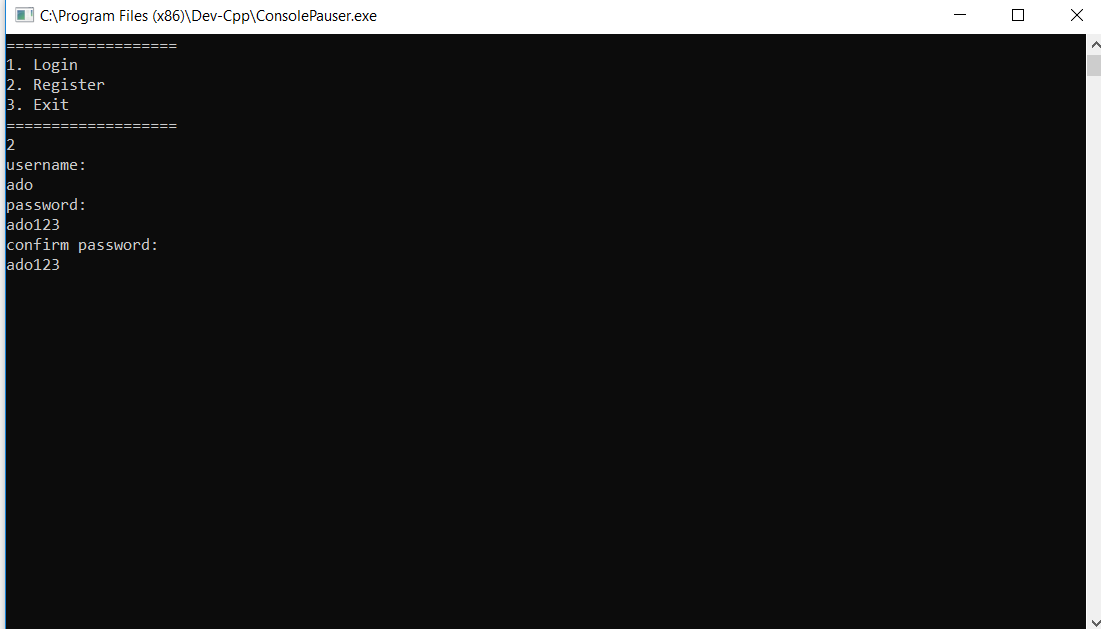
When first starting the program, the first thing you’ll see the UI that prompts you to login or register or exit.

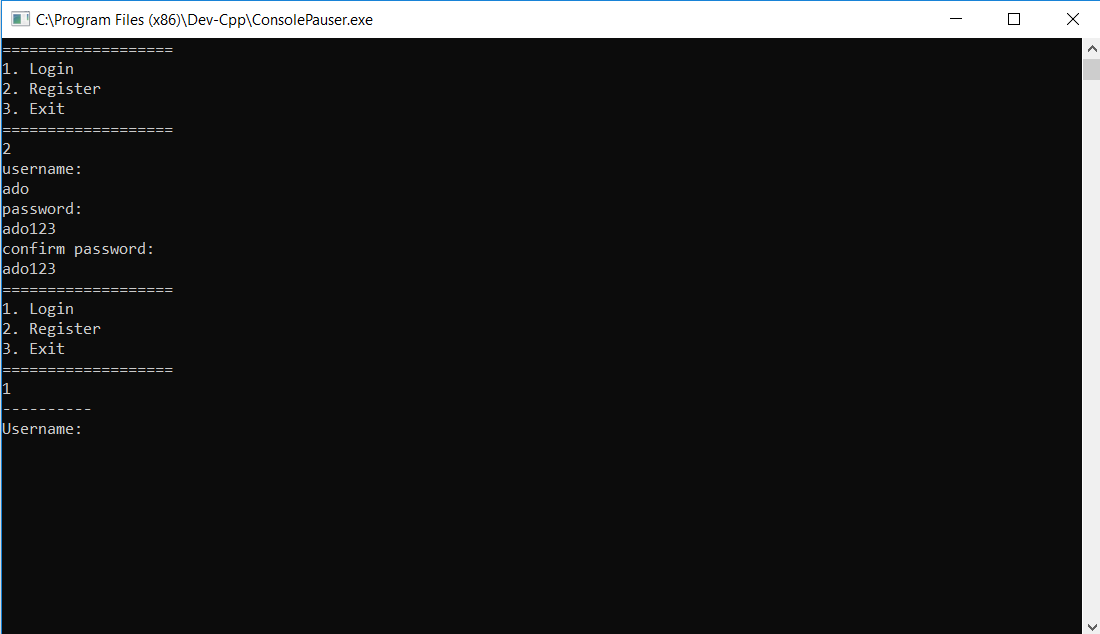
1. Since we don’t have an account yet, we should register first. So we input 2 (two).

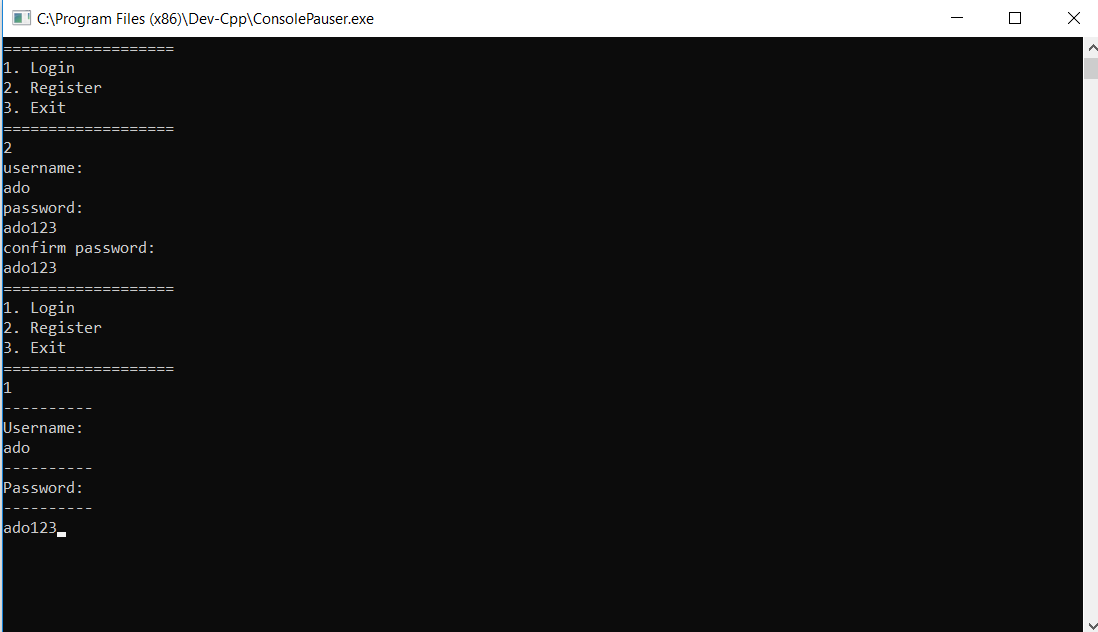
2. The program will ask me to input a username. I inputted my name for it.



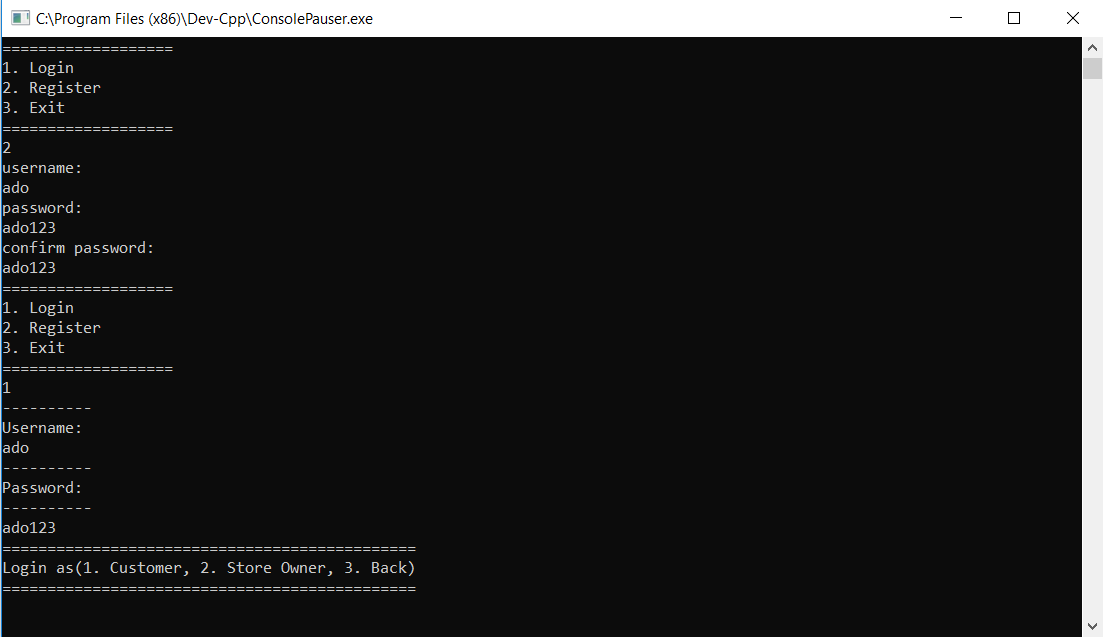
3. The program then asks for a password. I used ado123 for it.

4. The program will then ask you to confirm your password, and you’ll have to enter it again.

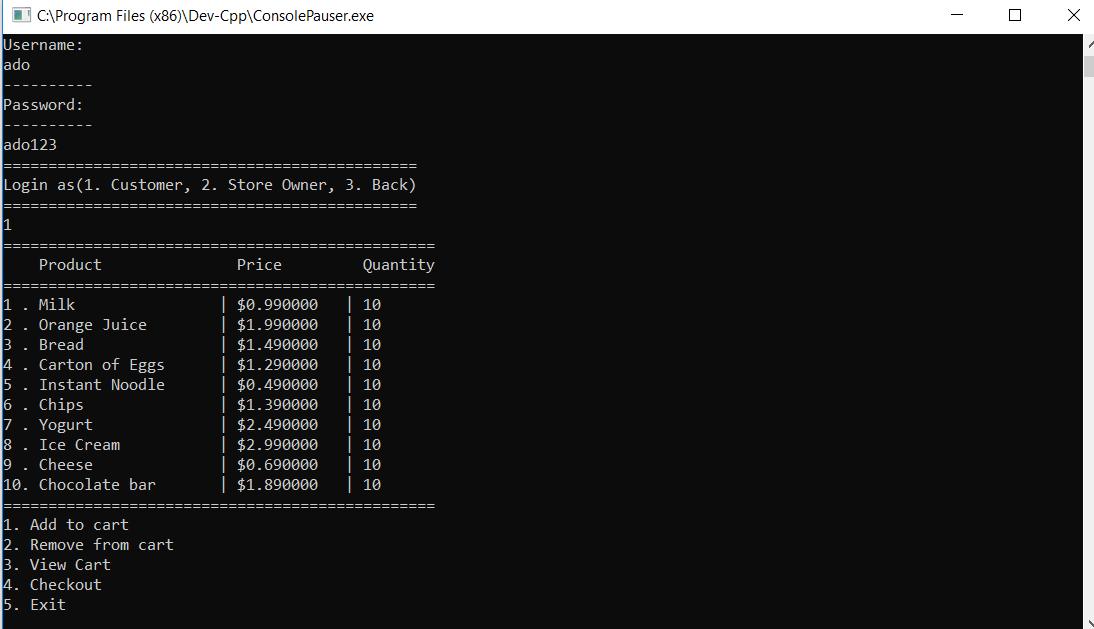
5. After confirming your password, the program then shows you the main interface again, where you can login, register, or exit. Since I already created an account, and I want to shop, I choose log in by inputting 1 (one).

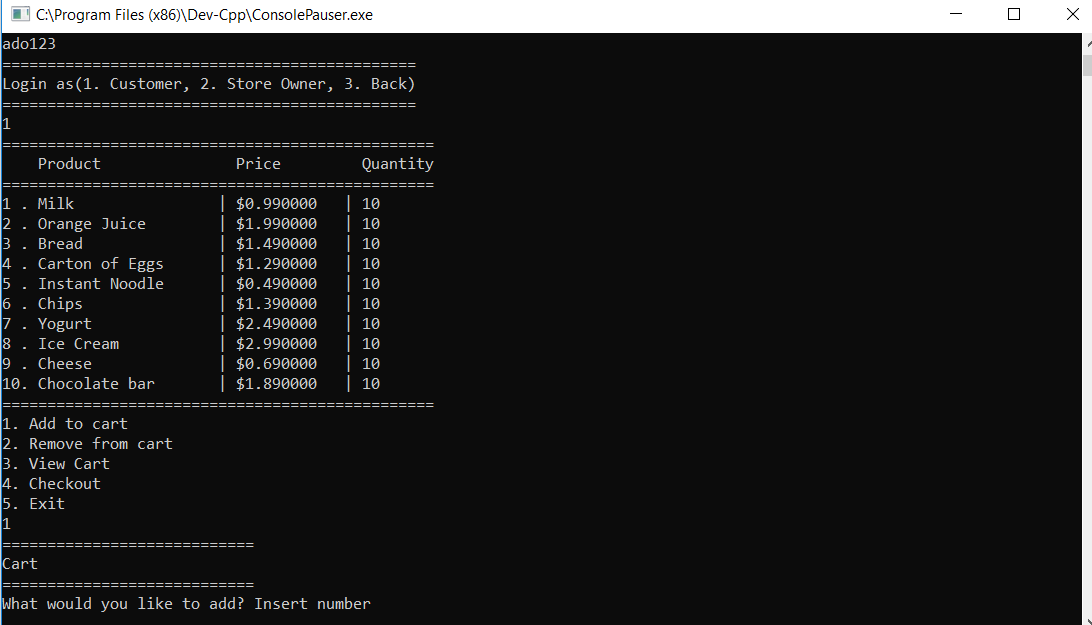


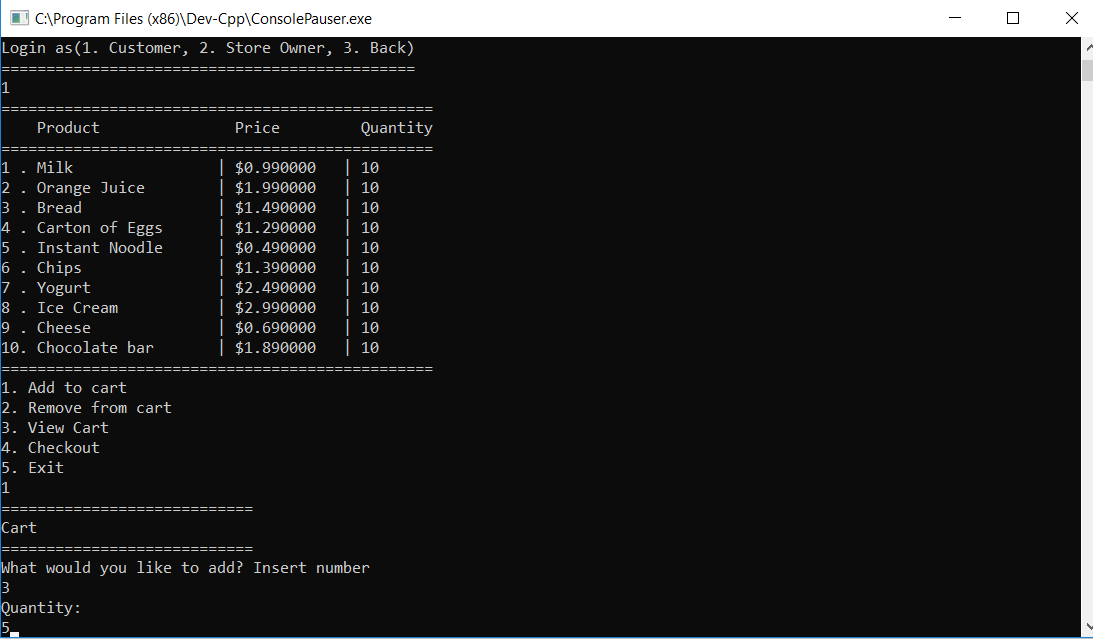
6. The program will prompt me to enter my username and password, so I did.

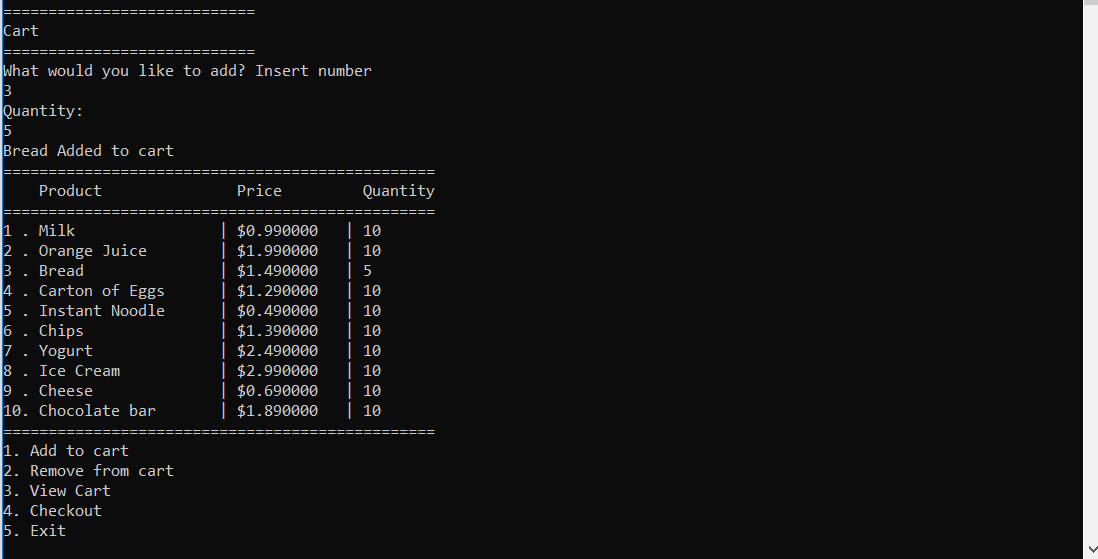


7. I am then asked to log in as a customer or as a store owner/admin. Let’s say I’m not a shop owner, and just want to buy some things from the shop, and so I input 1 (one) for customer.

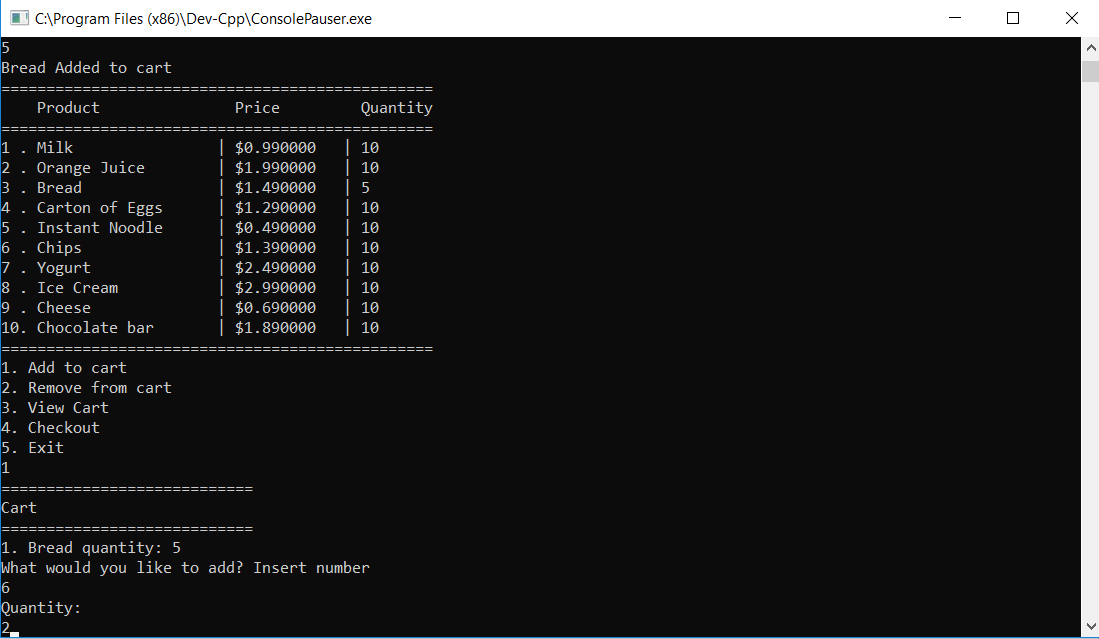
8. The program will show me what the supermarket has in store, their respective prices and quantity.

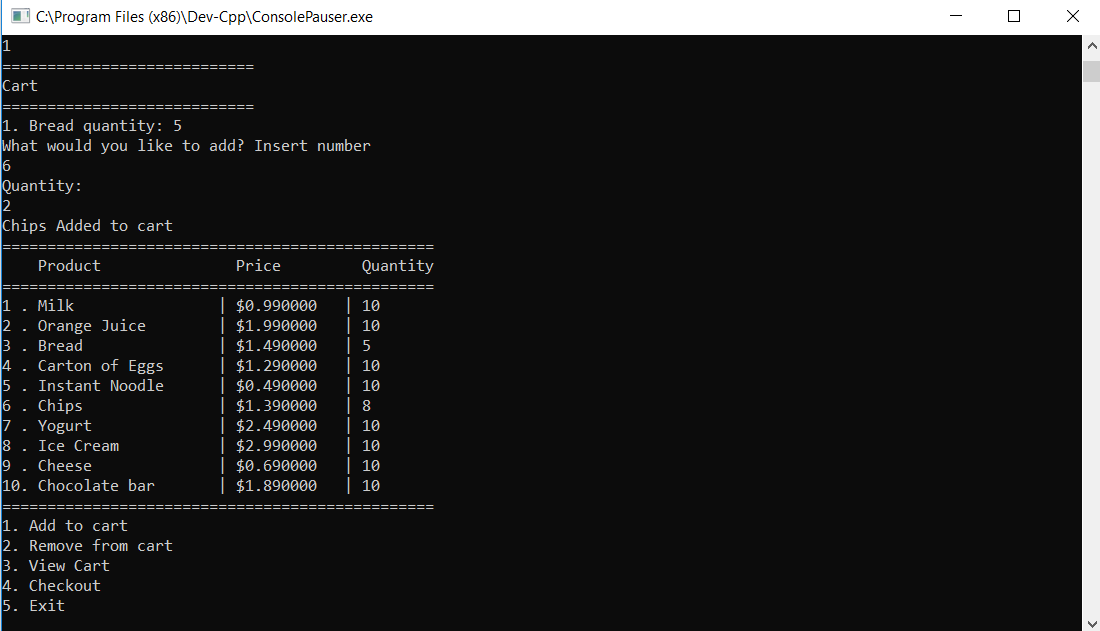
9. Say I want to add something to cart, so I input 1(one).

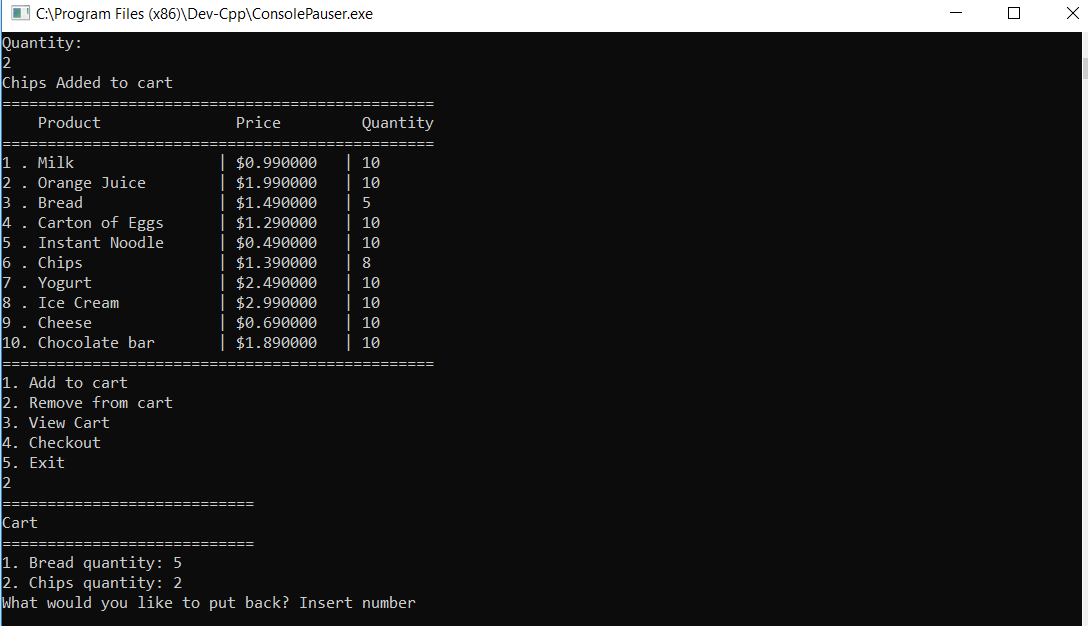
10. Say I want to buy some breads. I input 3 for bread, then the program asks how many do I want. I input 5 for five breads.



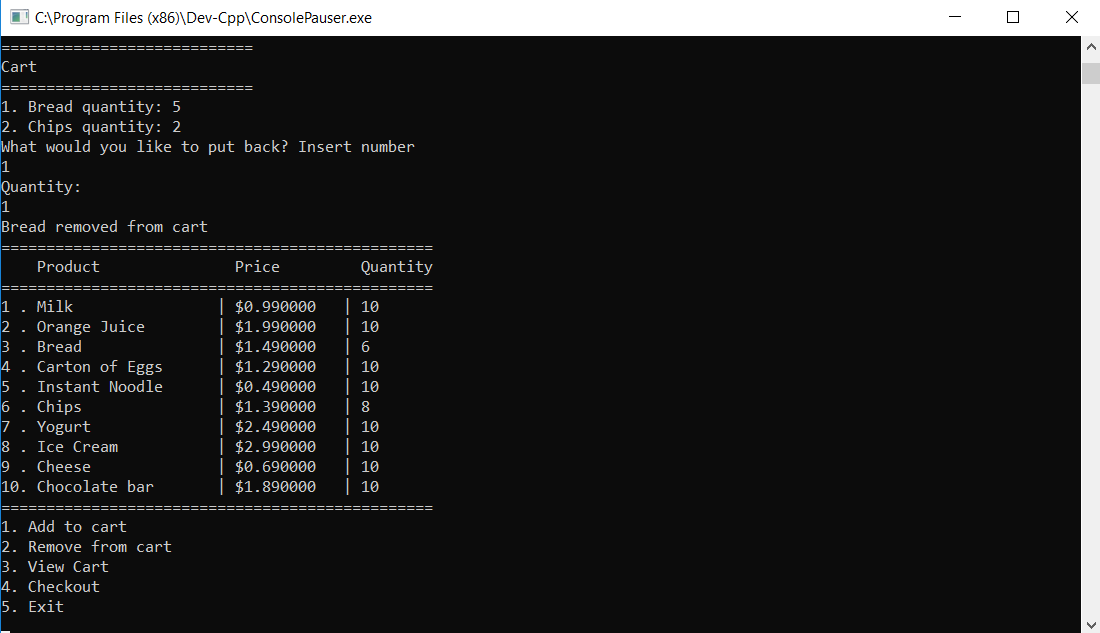
11. We can see now that we have put 5 breads into the cart, when the program displays the store again, bread quantity is only 5, because the default is 10, then subtracted by the 5 we put in cart.

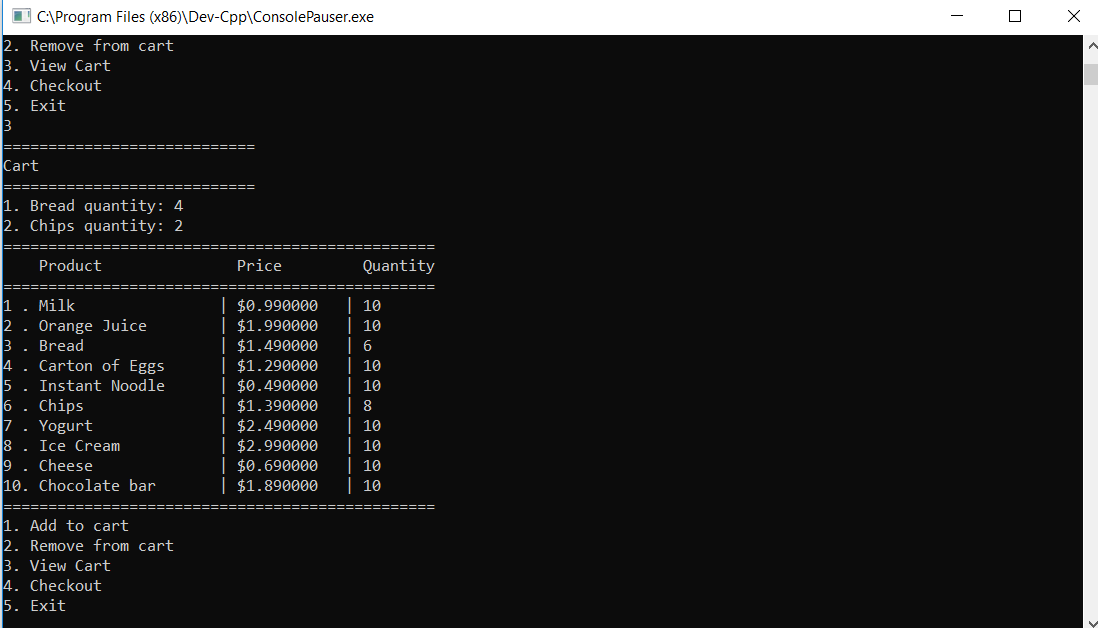
12. Say I want to buy some chips too. I chose 6 for chips, and added 2 chips to the cart.

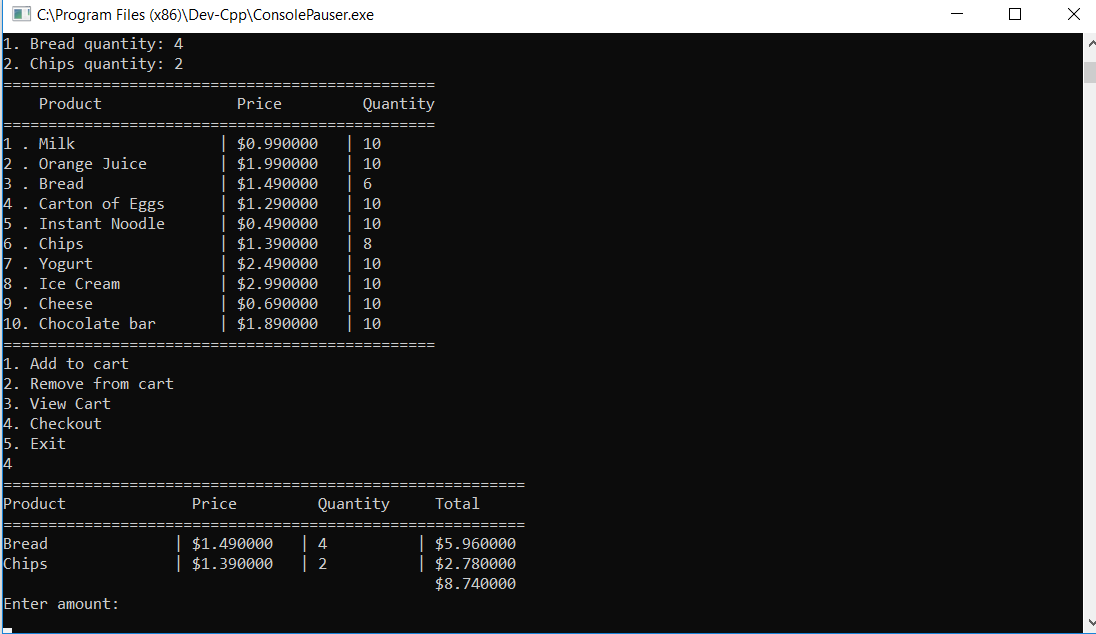
13. Again, now that I put 2 chips in the cart, the store is updated, and only 8 chips are left in the shelf.

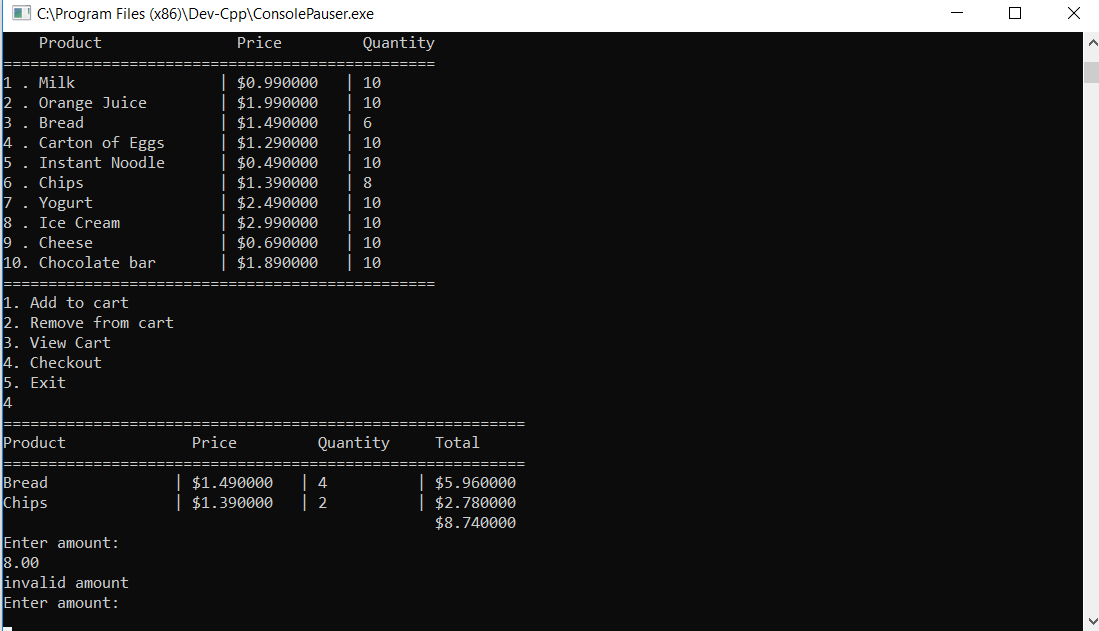


14. Uh-oh, it seems that I put too many bread in the cart, and I want to put one back to the shelf. So I input 2 to remove from cart, and the program will display the contents of the cart, and ask me what I want to put back.

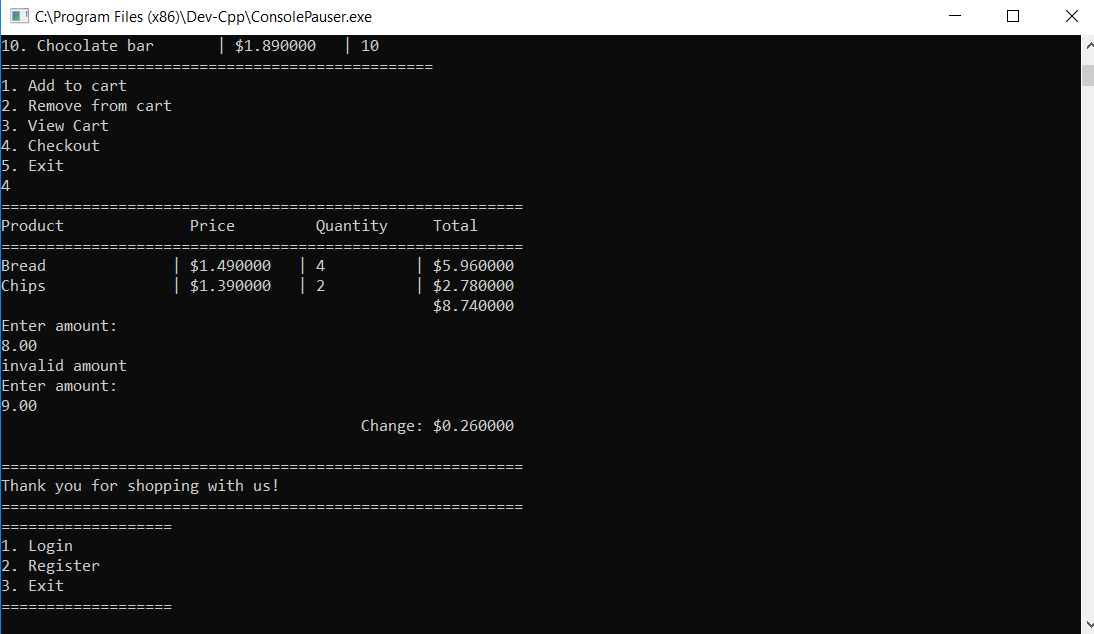
15. So, I input 1 for bread, and the program asks me how many do I want to put back. I only want to put back one, so I input 1. The program then displays the store again, with 6 breads in the shelf since I put one back.

16. Before I go to check out, I want to view my cart first, to make sure everything I want to buy is there. So, I input 3 to view cart. The program then displays the content of my cart. I have 4 breads, and 2 bags of chips.

17. Now that everything’s in order, I go to checkout, by inputting 4. The program then displays the checkout interface that adds up the total of everything I have in cart. I am now prompted to input my money. My total is $8.74, so I need to input that amount or more. But say I input a value fewer than that.



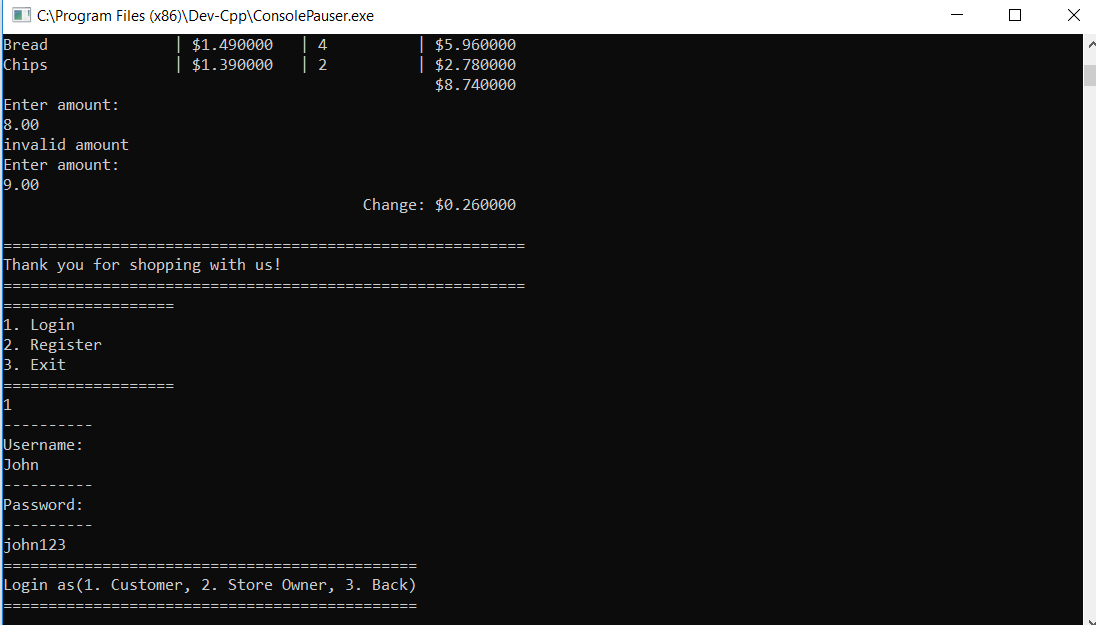
18. My input’s only $8.00, and the program knows that it’s fewer than my subtotal, so it tells me that my input is invalid, and that I need to enter a suitable amount of money.



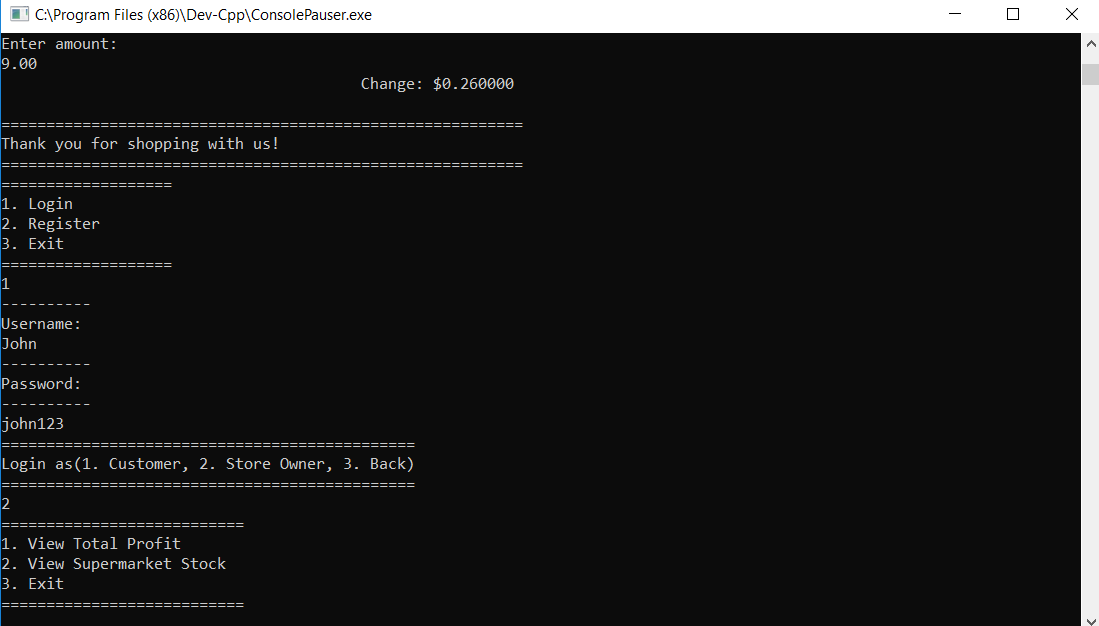
19. So, I enter $9.00, and the program will give me a change of $0.26, by subtracting my input amount with my subtotal. It then displays the main UI, with the options to login, register, or exit.

Now that I have bought something from the store, the store owner, John wants to see his income for this session, and also check his stocks.

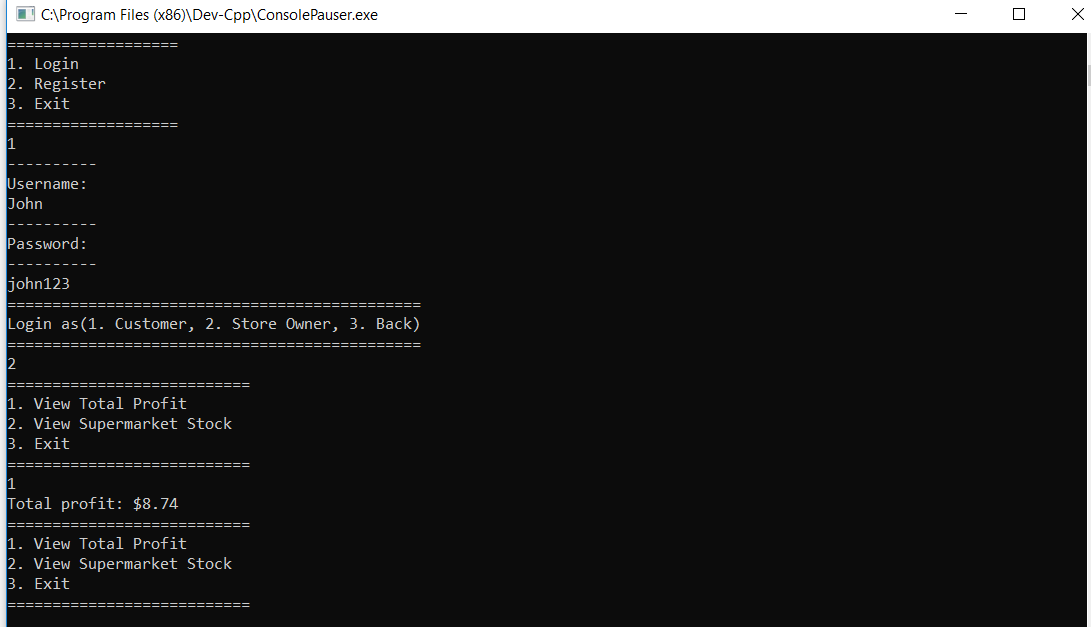
So, John will log in as admin, his username is John, and his password is john123.



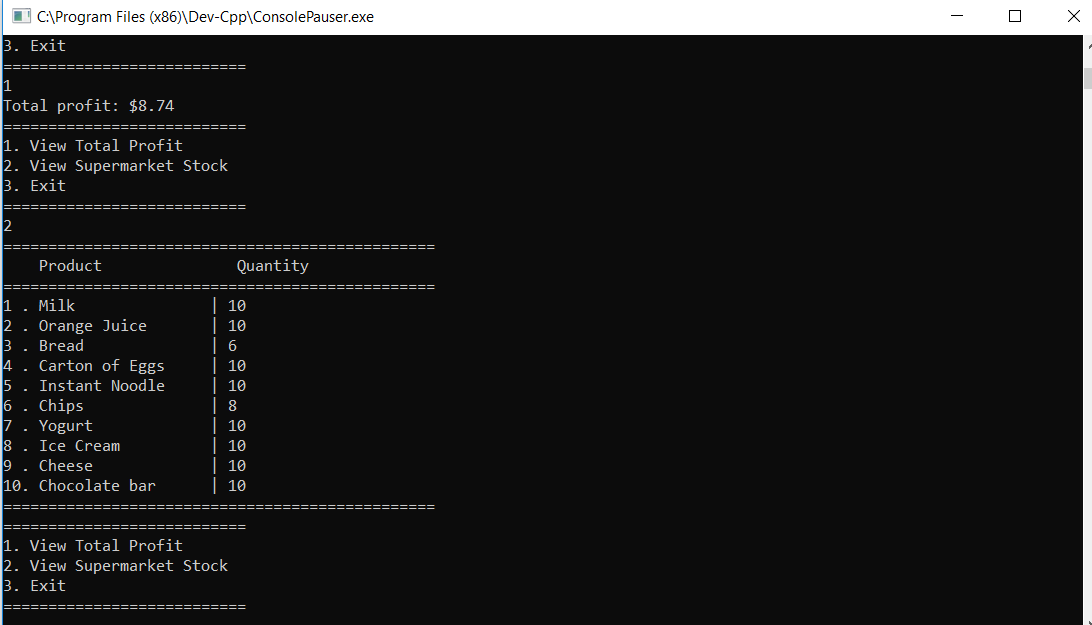
20. John will select 2 to log in as store owner.



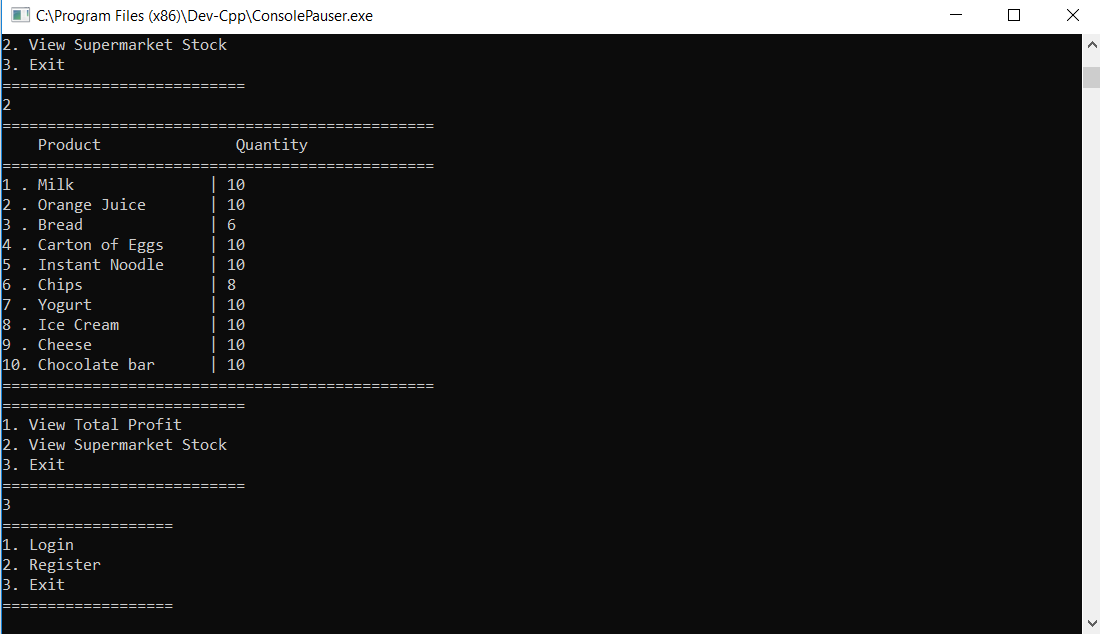
21. As store owner, John can view his total profit, or view his shop’s stocks.



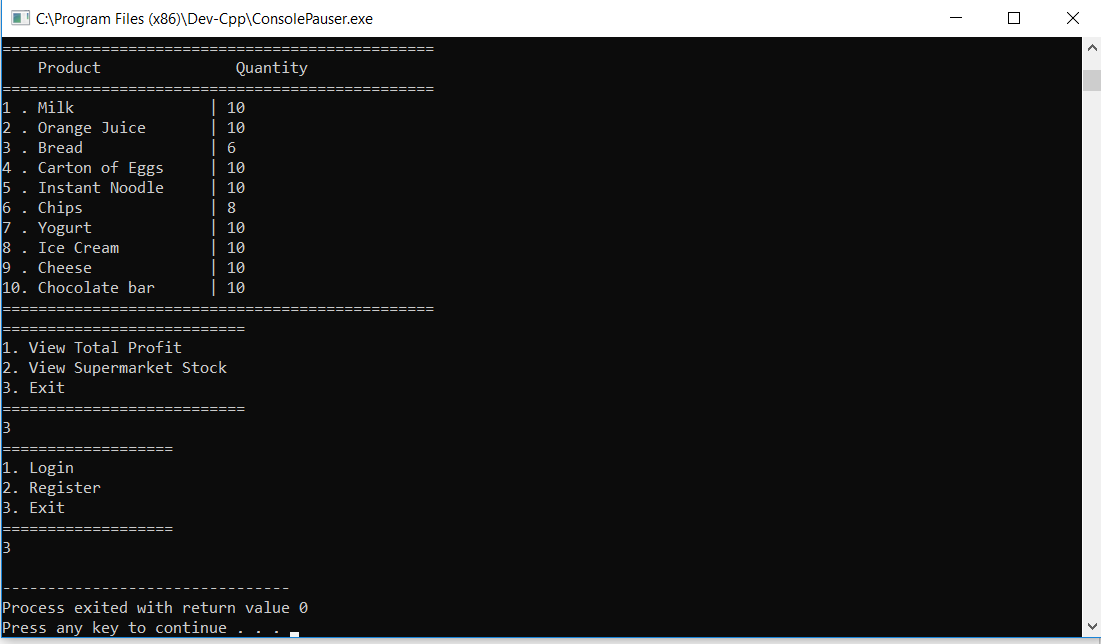
22. John wants to check his profits for the day first, so he inputs 1. The program then displays his total profit. Today, only ado went to his store to buy things. His total purchase was $8.74, so, John’s total profit for today is just that.



23. John then wants to check his leftover stocks for the day. He inputs 2, and the program then displays whatever he has at the store. The default quantity for everything is 10. Since ado has bought 4 pieces of bread and 2 chips, John knows that his stock of those two items has decreased.



24. Now that John is up to date with his store’s situation, he exits admin mode by inputting 3.



25. To completely exit the program, John then inputs 3, which means that the program as a whole will end and is free to exit.

**B. Time Complexity**

**Cart.cpp**

displayCart() = O(N)

addToCart() = O(N)

removeFromCart() = O(N)

printBill() = O(N)

**Supermarket.cpp**

display() = O(N)

displayUser() = O(N)

sm\_option() = O(N2)

**C. Program and Demo Link**

**Link to the GIT website**

<https://github.com/robertreden24/final-project-supermarket-c->

**Link to the demo video**

https://drive.google.com/open?id=1vQnNGWs-C5Ttwes12RoSmaQqKGy7\_uEC