

**Functioning of Super Market Billing system**

# CSE111 || Data Structures via C++

# Project Team

|  |  |  |  |
| --- | --- | --- | --- |
| **Student ID** | **Student Name** | **Project Evaluation** | |
| 222102487 | Aser Mohamed Ali Mahmoud | Report |  |
| Presentation |  |
| Code |  |
| Total |  |
| 223104648 | Mahmoud Emad Mohamed | Report |  |
| Presentation |  |
| Code |  |
| Total |  |
| 222200017 | Hossam Ayman Elders | Report |  |
| Presentation |  |
| Code |  |
| Total |  |
| 223104683 | Abdallah Mostafa Mohamed | Report |  |
| Presentation |  |
| Code |  |
| Total |  |
| 223102981 | Ahmed Saeed Ahmed | Report |  |
| Presentation |  |
| Code |  |
| Total |  |
| A20001133 | Mohamed hesham Abdalhalim | Report |  |
| Presentation |  |
| Code |  |
| Total |  |

**Under supervision of**

Prof. Mohamed Abdelaziz

TA || Dr. Radwa Taher

Summer-2024

# Abstract:

Supermarkets play a major role in people’s everyday lives. It is their first preference to buy groceries for their daily needs. This project is useful to both supermarket management and also to the consumers who is going to buy the goods. There is a need to have a perfect program to manage the database of available products and the products that are sold, which holds supermarket business together with achieving its goals.

This project introduces a comprehensive framework for managing the complexity of a supermarket structure for visualizing how a supermarket company actually does business. Using data structures, this project enables to shop without faulty billings. This program is written in C++ and runs via a C++ compiler implementing stacks, queues, Arrays and linked lists.

It is more computerized, fast, accurate system which is more convenient to the customers and management of the supermarket when compared to the old manual methods. The customer shall be able to have list of items with specific id numbers without rising confusion in similar products. The super market management can easily add, update and delete the items from the database which makes the program more efficient.

# Aim:

* The aim of the supermarket billing system is to have an efficient way to serve supermarket function fast and accurate.
* No customer wants to wait for their turn for a long time. This happens with an unorganized billing system. Every customer should know the items available in the supermarket, to purchase the products they intend to buy. Membership card holders should be given extra privileges such as a discount coupon of 500 EGP.

# Objective:

* 1. To understand the applications and real time usage of different types of data structures such as array, stack, queue, linked list.
  2. The main objective of the Supermarket billing system is to build a software and help supermarkets to calculate, display bills and serve the customer in a faster and efficient manner.
  3. To construct a strong and efficient algorithm, to develop a program which is editable and can later be used as a module for bigger software mechanism such as hypermarket.
  4. To develop a real time program which is efficient and has a fast processing and also has an industrial application.

# Scope / Applicability:

This project is a traditional supermarket billing system with some added functionality. This system is built for fast data processing and bill generation for supermarket customers. The supermarket billing system is built to help supermarkets calculate and display bills and serve the customer in a faster and efficient manner. The system reduces much of human efforts in calculating bill especially for huge number of products. It provides the list of items without any error. Using data structures, this project enables us to shop without faulty billings. We can have items with specific id number without rising confusion in similar products.

* Saves money and resources of organization and excludes use of paper or sheets in making bill.
* It can detect the product information and their price instantly, that saves time.
* It provides accuracy and faultless billing calculations.
* It is flexible and user-friendly.
* It also displays the purchased products through an electronic bill.

# Introduction:

The supermarket billing system automates the basic functions required for the fast and correct billing which is the most important feature for smooth functioning of a supermarket. The purpose of the system is to store details of all the products that are available in the supermarket and have a record of all the items that are available, sold, remaining amount and that are out of stock. The software is made efficient and fast by using the data structures that we have learnt to minimize the complexity in code. This software is designed to ease the work for administrator (user friendly) and reduce the waiting time for the customer (efficient). The customer is also provided with privileges such as discounts to increase the customer count according to the conditions.

**Purpose**

A process that is entirely software-based and does not need you to go through paperwork and spend hours on such a tedious task. Membership cards and discounts are given to the customers so as to boost the business of supermarket.

The motive behind this system is to make the customer feel free to make his/her choice in a very convenient manner without any paperwork and the need to wait for their turn and spending precious hours waiting in the queue. This is entirely a software-based process and does not need many workers to do manage the work. It makes the system of billing faster, simpler, and more efficient.

## Solution

The system helps to keep track of all the customers and their details who have shopped. The information regarding the products can be easily modified.

Constructing a strong and efficient algorithm and develop a program which is editable and can later be used as a module for bigger software mechanism.

## Conclusion

A supermarket billing System now a days is completely being operated through software mode and has more merits than any other manual system present. It makes the process easier, clear, uses less time, less energy, and more effective.

# Concepts of data structures used:

# *(All types of linear data structures)*

* + 1. Linked list.
    2. Array.
    3. Stack.
    4. Queue.

# Implementation: (Flowchart)

1. **MAIN MENU**

Start

Login Stop

Main Menu

* 1. Stock 2)Customer

3)Membership Customer 4)Admin

5)Exit

6

* 1. **STOCK (LINKED LIST, ARRAY)**

Start

* + 1. Stock

A

1. ADD a new product
2. display all products B
3. MODIFY Existing

product C

1. Delete a particular

product item D

0. Main Menu

Main

E Menu

7

A

Enter product ID Enter product Name Enter product price Enter product quantity

This product is Inserted!

B

Products

Total products in store is : 0

ID

Product Name Price

Quantity

Total products in our store

8

C

Products

Not found Old ID

Enter new ID Enter new Name

Enter new Price Enter newQuantity Total products in our store

Item is modified

D

Products

ID to delete that product

NOT FOUND

Item is Deleted

0 products

### CUSTOMER (STACK, ARRAY, QUEUE, FILES)

QUANTITY

ENTERED NOT AVAILABLE

MAIN MENU

ID to delete that product

NOT FOUND

Item is Deleted



0 products

IF ID AVAILABLE

NOT

AVAILABLE AT STORE

QUANTITY TO BE BE PURCHASED

QUANTITY

AVAILABILIT Y CHECK

* 1. **MEMBERSHIP CUSTOMER (STACK, ARRAY, QUEUE)**

ITEMS PURCHASED

BILL AMOUNT (DISCOUNT IF ELIGIGLE) THANK YOU TO SHOP

CUSTOMER DETAILS TO BE S1T1ORED

CUSTOMER NAME

CUSTOMER

ID OF THE ITEM TO BE PURCHASED

START

NEW MEMBER REGISTER:

ENTER CUSTOMER NAME

PRINT EXISTING PRODUCTS

BILL GENERATION OF AMOUNT AFTER 20 % DISCOUNT

DETAILS OF COUPONS

PRINT AMOUNT+1000

MEMBERSHIP DISCOUNT FROM NEXT TIME

STOP

### ADMIN (ARRAY, QUEUE)

START

ENTER THE OPERATION TO BE

PERFORMED

1

2

DISPLAY ALL

PRODUCTS IN STOCK

CUSTOMER LIST

DEQUEUE

CUSTOMER

GENERATE

MAIN MENU

4

0

3

* 1. **EXIT**

START

EXIT

THANK YOU

STOP

# List of modules:

* Stock
* Customer
* Membership Customer
* Admin

# Module description:

## Stock

1. Add product: Add the product in the store by entering ID, Name, price and Quantity using linked list data structure.
2. Display Products: The available products are displayed in the store along with their prices and quantity using array data structure.
3. Modify Products: If the configuration of a product is found to be faulty, or if the products stock has increased, the product details can be modified.
4. Remove items: If the inventory of the product is unavailable or expired the Stock can remove the product from the list.
5. Back to main menu: move back to main menu to go to customer menu or to exit.

## Customer

1. Assign basket: Once the customer function is selected, it will assign the customer a trolley number using stack data structure.
2. Buy items: Customer should enter the number of items he wants to buy and then all the products customer has chosen will go to the assign basket along with the quantity of each item selected. If the product is out of stock or id entered by

the user is invalid, it directs us back to the main menu. This can be done using array data structure.

1. Enter customer queue: After customer has selected all the required items, he/she will be added to the customer queue using queue data structure.

## Membership customer

1. Existing Member: If a customer is already having membership card he’s given special privileges like discount and coupons.
2. New Member Register: If a customer wants to register for a new membership card, the additional amount for it will be added to the final amount and will be given discounts for their next purchases.
3. All the other functions and data structures used are same as the customer module.

## Admin

1. Display Products: The available products are displayed in the stock module along with their prices and quantity using array data structure.
2. Customers list: To regulate online traffic, first-come first-serve (FIFO) has to be implemented. This can be done using linked queue data structure. The customer will be put in a queue of all the other buyers for this product. The customer is added using ENQUEUE.
3. Dequeue customer: Once the customer has been assigned the product, he is taken off the waiting list by DEQUEUE Once the customer’s transactions are complete, he has to be removed from the server, can be done using dequeue.

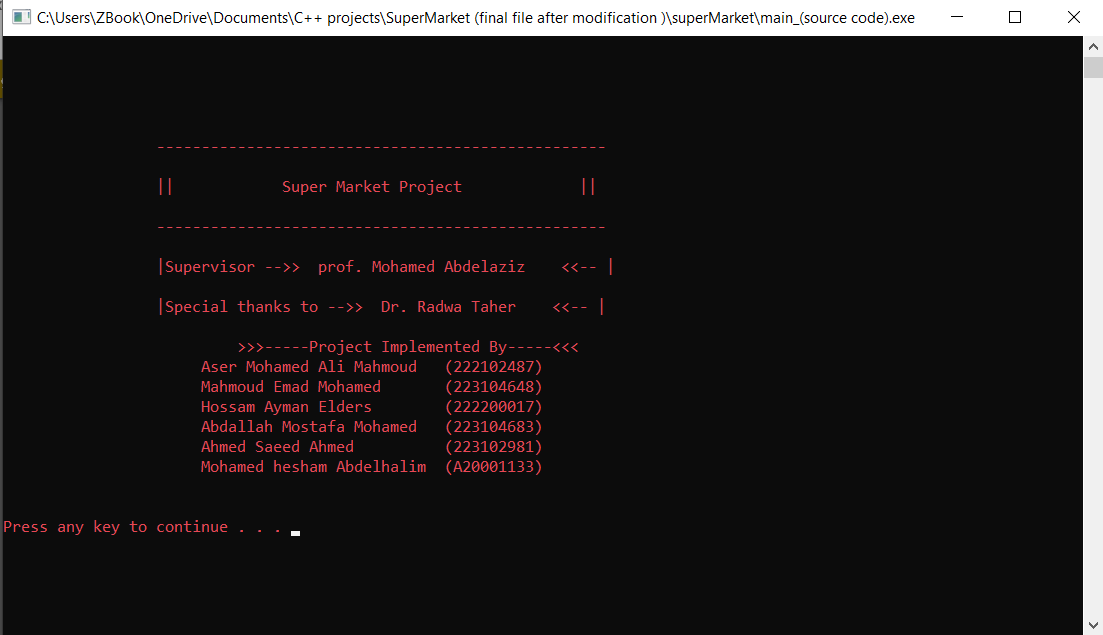
# Supermarket billing system main code:

# #include<iostream> #include<string> #include <sstream> #include <bits/stdc++.h> #include<windows.h> #include"queue1.h" #include"animation.h" #include"stackme.h" // Three header files using namespace std; int search(int); int display(); string check(int); // for checking quantity //////////////////////////////////////////////////////////////////// struct node { int ID; string proName; double prePrice; int quantity; struct node\* next; }; struct customer { int ID; string name; }; struct node \*head=NULL; //////////////////////////////////////////////////////////////////// void beg() { system("cls"); //clear the screen int id,quant; // quant for quantity string name; double pre; // pre for price struct node \*t=new node; struct node \*p=head; int n; cout<<"\t\t\t enter product ID:-"<<endl; bool states=false; do { cin>>id; if(search(id)!=0) { t->ID=id; states=true; }else { cout<<"id is already exist please try another one"<<endl; }} while(states==false); cout<<"\t\t\tEnter product Name:-"; cin>>name; t->proName=name; cout<<"\t\t\tEnter product price:-"; cin>>pre; t->prePrice=pre; cout<<"\t\t\tEnter product quantity:-"; cin>>quant; t->quantity=quant; if(head==NULL) { t->next=head; head=t; } else { while(p->next!=NULL) { p=p->next; } p->next=t; t->next=NULL; } system("cls"); cout<<"\n\n\t\t\t\tThis product is Inserted!\n\n\n"; } /////////////////////////////////////////////////////////////////////////////////////////// int search(int id) // for search item in list { int count=1; struct node \*p=head; while(p!=NULL) { if(p->ID==id) return 0; else{ count++; p=p->next; } } return count; } //////////////////////////////////////////////////////////////////////// void delPro() { system("cls"); display(); int id; struct node \*cur=head; struct node \*pre=head; cout<<"\n\nEnter ID to delete that product:\n\n"; cin>>id; if (head == NULL) { system("cls"); cout<<"List is empty"<<endl; } int pos=0; int count=display(); // for load no of nodes pos=search(id); // for check weather desire node is exist or not if(pos<=count && pos!=0){ while(cur->ID!=id){ // for delete middle area products pre=cur; cur=cur->next; } pre->next=cur->next; system("cls"); cout<<"\n<<item is deleted>>\n"; }else{ cout<<"\n<<<Not found>>\n\n"; } } ////////////////////////////////////////////////////////////////////////////////// void modify() { int id; double pre; // pre for price string pName; int nid; int nq; // pName for new name if (head == NULL) { system("cls"); cout<<"List is empty"<<endl; } else { display(); cout<<"\n\nEnter ID to modify product Name and its price:\n"; cin>>id; struct node \*cur=head; int pos=0; int count=display(); // for load no of nodes pos=search(id); // for check weather desire node is exist or not if(pos<=count) { while(cur->ID!=id) { cur=cur->next; } cout<<"\nOld ID : "<<cur->ID<<endl; cout<<"\nOld Name : "<<cur->proName<<endl; cout<<"\nOld Price : "<<cur->prePrice<<endl; cout<<"\nOld Quantity : "<<cur->quantity<<endl; cout<<endl<<endl; cout<<"Enter new ID:"; bool states=false; do { cin>>nid; if (nid==cur->ID) { cur->ID=nid; states=true; } else if(search(nid)!=0) { cur->ID=nid; states=true; } else {cout<<"id is already exist please try another one"<<endl; }} while(states==false); cout<<"Enter new Name:"; cin>>pName; cur->proName=pName; cout<<"Enter new Price:"; cin>>pre; cur->prePrice=pre; cout<<"Enter new Quantity:"; cin>>nq; cur->quantity=nq; cout<<"\n-------Product Modified-------\n"; } else { cout<<id<<" is <<<Not found>>\n\n"; } } } ////////////////////////////////////////////////////////////////////////////////////// int display() { system("cls"); int c=0; // c for count products struct node \*p=head; cout<<"Existing products are:\n"; cout<<"ID\t\tProduct Name\t\tPrice\t\tQuantity\n"; cout<<"=================================================================|\n"; while(p!=NULL) { cout<<p->ID<<"\t\t"<<p->proName<<"\t\t\t"<<p->prePrice<<"\t\t\t"<<check(p->quantity)<<"\n"; // call check func and pass quantity p=p->next; c=c+1; } cout<<"\nTotal products in our store is : "<<c<<"\n\n\n"; return c; } //////////////////////////////////////////////////////////////////////////////////////// string check(int quant) { // check function int a = quant; stringstream ss; ss << a; string quantity = ss.str(); if(quant<=0) return "out of stock!"; else return quantity; } /////////////////////////////////////////////////////////////////////// void buy() { system("cls"); display(); string products[20]; // for display sold items int pay=0,no,c=0,price,id,i=1; if(head==NULL) { cout<<"\n<<<<There is no items to buy>>>>\n\n"; } else { cout<<"How many items you want to buy!\n"; cin>>no; int count=display(); // for store no of nodes in c while (i<=no) { struct node \*cur=head; int quant,cho; a: // quant for quantity and cho for choice cout<<"Enter id of item that you want to buy: "; int id,pos=0; cin>>id; if(id==-1){system("cls"); return; } pos=search(id); if(pos<=count) { // item is available in store while(cur->ID!=id) { cur=cur->next; } cout<<"How many quantities you want:"; cin>>quant; if(cur->quantity<quant) { cout<<"\n\n\t\t\t----The Quantity You Entered is not available---"<<endl; cout<<"\n\t\t\t-----(Press -1 for Back to Main Menu)------"<<endl; goto a; } products[c]=cur->proName; // this will conatin the items buy names in array; c++; pay=pay+(cur->prePrice\*quant); // calculate Bill cur->quantity=cur->quantity-quant; // change quantity i++; } else { cout<<"\n<<<<<<<<<This item is not available in our store at this time>>>>\n\n"; } } string customer; cout<<"\n\t\t Enter Your Name :"; cin>>customer; enqueue(customer); system("cls"); cout<<"\n\n\n\n\t\t\tYou have bought : "; for(int i=0;i<no;i++) { // show that item you have bought cout<<products[i]<<" "; } if(pay>=3000){ price=pay\*(0.90); // with 10% discount cout<<"\n\nOriginal price : "<<pay; cout<<"\n with 10% discount: "<<price<<"\nThank you for shopping !\n\n"; } else{ price=pay; // with no discount cout<<"\n Bill Amount: "<<price<<"\nThank you for shopping !\n\n"; } { ofstream fout; string line; fout.open("recent\_customer.txt"); cout<<"please enter your Name"<<endl; while (fout) { getline(cin, line); if (line == "-1") cout<<"hi"<<endl; break; fout << line << endl; cout<<"bye"<<endl; } fout.close(); ifstream fin; } } } ///////////////////////////////////////////////////////////////////////////////////////// int membership() { string customer; cout<<"\n\t\t Enter Your User Name :"; cin.ignore(); getline(cin,customer); enqueue(customer); system("cls"); display(); string products[20]; // for display sold items int pay=0,no,c=0,price,id,i=1; if(head==NULL) { cout<<"\n<<<<There is no items to buy>>>>\n\n"; } else { cout<<"How many items you want to buy!\n"; cin>>no; int count=display(); // for store no of nodes in c while (i<=no) { struct node \*cur=head; int quant,cho; a: // quant for quantity and cho for choice cout<<"Enter id of item that you want to buy: "; int id,pos=0; cin>>id; if(id==-1) { system("cls"); } pos=search(id); if(pos<=count) { // item is available in store while(cur->ID!=id) { cur=cur->next; } cout<<"How many quantities you want:"; cin>>quant; if(cur->quantity<quant) { cout<<"\n\n\t\t\t----The Quantity You Entered is not available---"<<endl; cout<<"\n\t\t\t-----(Press -1 for Back to Main Menu)------"<<endl; goto a; } products[c]=cur->proName; // this will conatin the items buy names in array; c++; pay=pay+(cur->prePrice\*quant); // calculate Bill cur->quantity=cur->quantity-quant; // change quantity i++; } else { cout<<"\n<<<<<<<<<This item is not available in our store at this time>>>>\n\n"; } } system("cls"); cout<<"\n\n\n\n\t\t\tYou have bought : "; for(int i=0;i<no;i++) { // show that item you have bought cout<<products[i]<<" "; } price=pay\*(0.80); // with 20% discount as member cout<<"\n\nOriginal price : "<<pay; cout<<"\n with 20% discount: "<<price; } if(pay>=3000){ // coupon cout<<"\n You have won coupon worth EGP 500 "<< "\nThank you for shopping !\n\n"; } else{ cout<<"\n shop more to win gift coupons"<< "\nThank you for shopping !\n\n"; } } ///////////////////////////////////////////////////////////////////////////////////////// int newmembership() { struct customer \*cus = new customer; int cou = 1; cout<<"\n\t\t Enter Your Name :"; getline (cin,cus->name); cus->ID=cou; cou++; enqueue(cus->name); display(); string products[20]; // for display sold items int pay=0,no,c=0,price,id,i=1; if(head==NULL) { cout<<"\n<<<<There is no items to buy>>>>\n\n"; } else { cout<<"How many items you want to buy!\n"; cin>>no; int count=display(); // for store no of nodes in c while (i<=no) { struct node \*cur=head; int quant,cho; a: // quant for quantity and cho for choice cout<<"Enter id of item that you want to buy: "; int id,pos=0; cin>>id; if(id==-1) { system("cls"); } pos=search(id); if(pos<=count) { // item is available in store while(cur->ID!=id) { cur=cur->next; } cout<<"How many quantities you want:"; cin>>quant; if(cur->quantity<quant) { cout<<"\n\n\t\t\t----The Quantity You Entered is not available---"<<endl; cout<<"\n\t\t\t-----(Press -1 for Back to Main Menu)------"<<endl; goto a; } products[c]=cur->proName; // this will conatin the items buy names in array; c++; pay=pay+(cur->prePrice\*quant); // calculate Bill cur->quantity=cur->quantity-quant; // change quantity i++; } else { cout<<"\n<<<<<<<<<This item is not available in our store at this time>>>>\n\n"; } } system("cls"); cout<<"\n\n\n\n\t\t\tYou have bought : "; for(int i=0;i<no;i++) { // show that item you have bought cout<<products[i]<<" "; } price=(pay+1000); // amount + membership (will get discount from next time) cout<<"\n\nOriginal price : "<<pay; cout<<"\n with purchase of membership card: "<<price; //adding price of membership card } cout<<"\n You will get membership discount from next time "<< "\nThank you for shopping !\n\n"; } //////////////////////////////////////////////////////////////////////////////////////// int stock() { system("cls"); int ch; do { // choice for below message cout<<"\t\t============================================"<<endl; cout<<"\t\t| Stock |"<<endl; cout<<"\t\t============================================"<<endl; cout<<"\t\t Enter 1 for ADD a new product "<<endl; cout<<"\t\t Enter 2 to display all products "<<endl; cout<<"\t\t Enter 3 for MODIFY Existing product"<<endl; cout<<"\t\t Enter 4 for Delete a particular product item"<<endl; cout<<"\t\t Enter 0 for Main Menu"<<endl; cout<<"\nEnter Your choice >>>"; cin>>ch; switch(ch){ case 1: beg(); break; case 2: system("cls"); display(); break; case 3: modify(); system("cls"); break; case 4: delPro(); //cout<<"\n-------Product is Deleted-------\n"; break; default: system("cls"); } } while(ch!=0) ; } //////////////////////////////////////////////// void administator() { int ch; system("cls"); do { // choice for below message cout<<"\t\t============================================"<<endl; cout<<"\t\t| Administator Portal |"<<endl; cout<<"\t\t============================================"<<endl; cout<<"\t\t Enter 1 to display all products "<<endl; cout<<"\t\t Enter 2 for Customers List "<<endl; cout<<"\t\t Enter 3 for Dequeue customer"<<endl; cout<<"\t\t Enter 0 for Main Menu"<<endl; cout<<"\nEnter Your choice >>>"; cin>>ch; switch(ch){ case 1: system("cls"); display(); break; case 2: system("cls"); cout<<"|============CUSTOMERS NAMES LIST==============|"<<endl; displayQueue(); break; case 3: system("cls"); cout<<"|============CUSTOMERS NAMES LIST==============|"<<endl; dequeue(); displayQueue(); break; default: system("cls"); } } while(ch!=0) ; } *///////////////////////////////////////////////////////////////////////////* int main() { for(int i=0;i<=51;i++) { push(i); } system("color 0C"); // for console color gotoxy(17,5); cout<<"--------------------------------------------------"<<endl; gotoxy(17,7); cout<<"|| Super Market Project ||"<<endl; gotoxy(17,9); cout<<"--------------------------------------------------"<<endl; gotoxy(17,11); cout<<"|Supervisor -->> prof. Mohamed Abdelaziz <<-- |\n"<<endl; gotoxy(17,13);

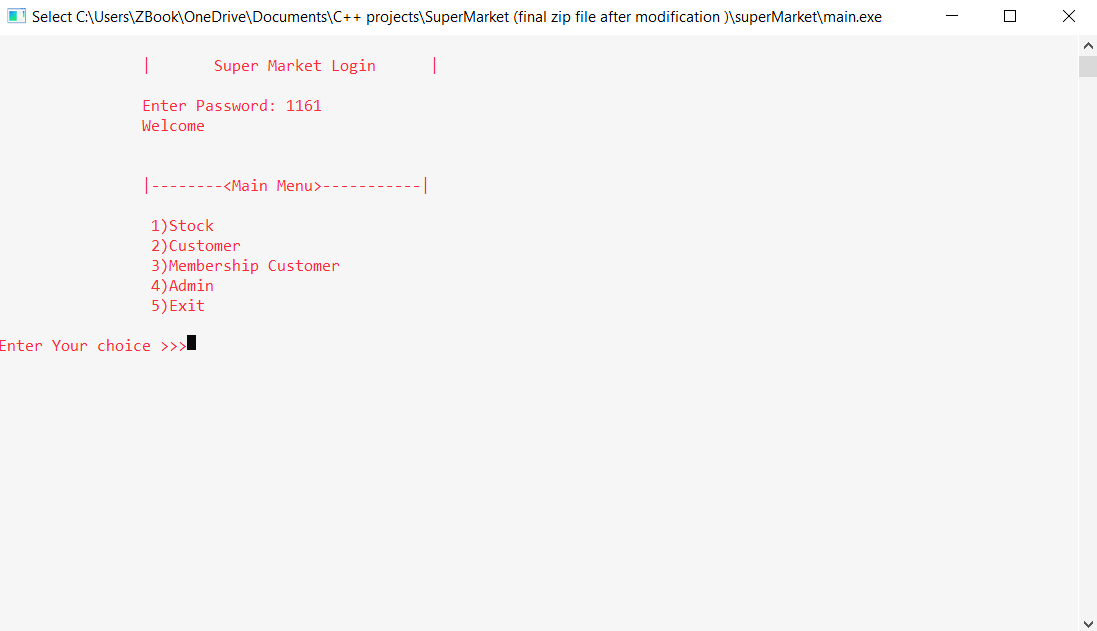
# cout<<"|Special thanks to --->>Dr. Radwa Taher <<--|\n"<<endl;

# gotoxy(17,15); cout<<" >>>-----Project Implemented By-----<<<"<<endl; gotoxy(22,16); cout<<"Aser Mohamed Ali Mahmoud (222102487)"<<endl; gotoxy(22,17); cout<<"Mahmoud Emad Mohamed (223104648)"<<endl; gotoxy(22,18); cout<<"Hossam Ayman Elders (222200017)"<<endl; gotoxy(22,19); cout<<"Abdallah Mostafa Mohamed (223104683)"<<endl; gotoxy(22,20); cout<<"Ahmed Saeed Ahmed (223102981)"<<endl; gotoxy(22,21); cout<<"Mohamed hesham Abdalhalim (A20001133)"<<endl; gotoxy(22,22); cout<<""<<endl<<endl; system("pause"); system("cls"); system("color Fc"); /////////////////////////// int ps,profit=0; cout<<"\n\t\t| Super Market Login |\n"; cout<<"\n\t\tEnter Password: "; cin>>ps; if(ps==1161) { cout<<"\t\tWelcome \n\n"; /////////////////////////// int ch; while(ch!=5){ // choice for below message cout<<"\n\t\t|--------<Main Menu>-----------|"; cout<<"\n\n"; cout<<"\t\t 1)Stock \n"; cout<<"\t\t 2)Customer \n"; cout<<"\t\t 3)Membership Customer \n"; cout<<"\t\t 4)Admin \n"; cout<<"\t\t 5)Exit \n"; cout<<"\nEnter Your choice >>>";cin>>ch; switch(ch){ case 1: stock(); break; case 2: cout<<endl<<endl; bpop(); system("pause"); buy(); break; case 3: int choice; while(choice!=2){ cout<<"\t\t 1)Existing Member \n"; cout<<"\t\t 2)New Member Register \n"; cout<<"\nEnter Your choice >>>";cin>>choice; switch(choice){ case 1: membership(); break; case 2: newmembership(); break; } break; } break; case 4: administator(); break; case 5: cout<<"\n\n\t\t\t\t\tThank You\t\t\t\t"; break; } } return 0; } else{ cout<<"\t\tWrong password \n\n"; } }

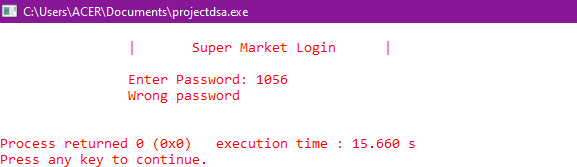
**Results and Discussion** (Working and Outputs of code):

As soon as we run the code, we get the details of this project and further we can continue into working by entering any key :

To secure the details we set a password to verify that only eligible person can access it :

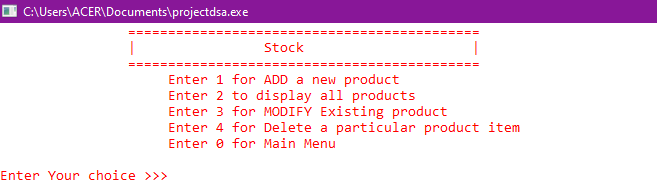


If a non-eligible person tries to access the program automatically terminate:

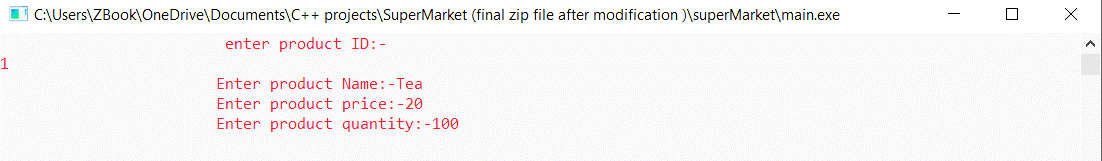


**MODULE 1 (STOCK):**

If we select option (1) then it leads us to stock module operations as follows:

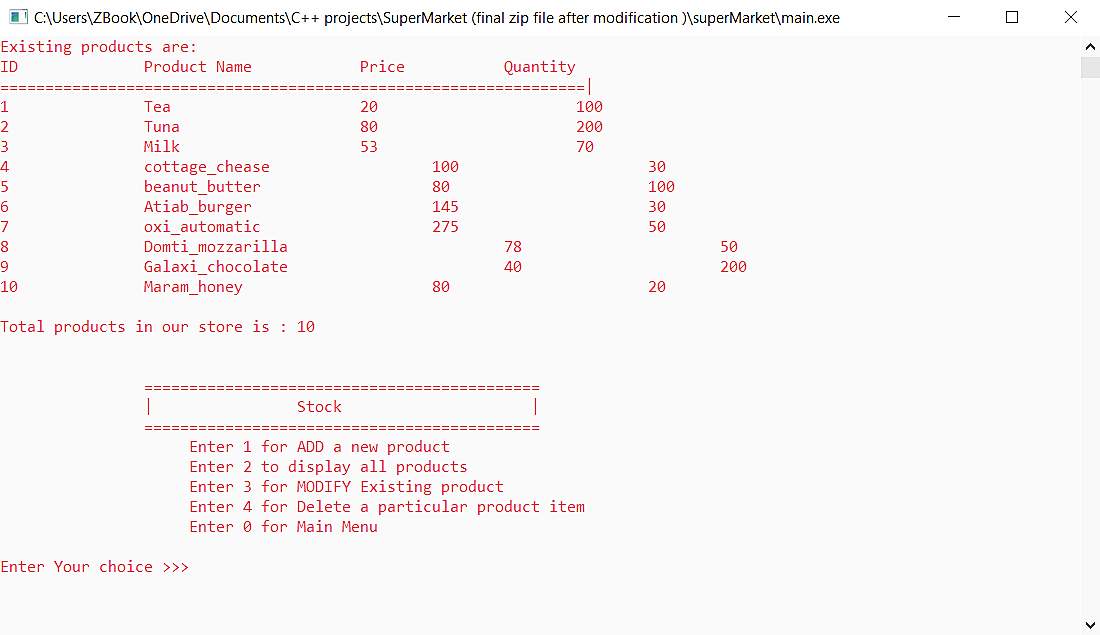


Addition of products to stock:

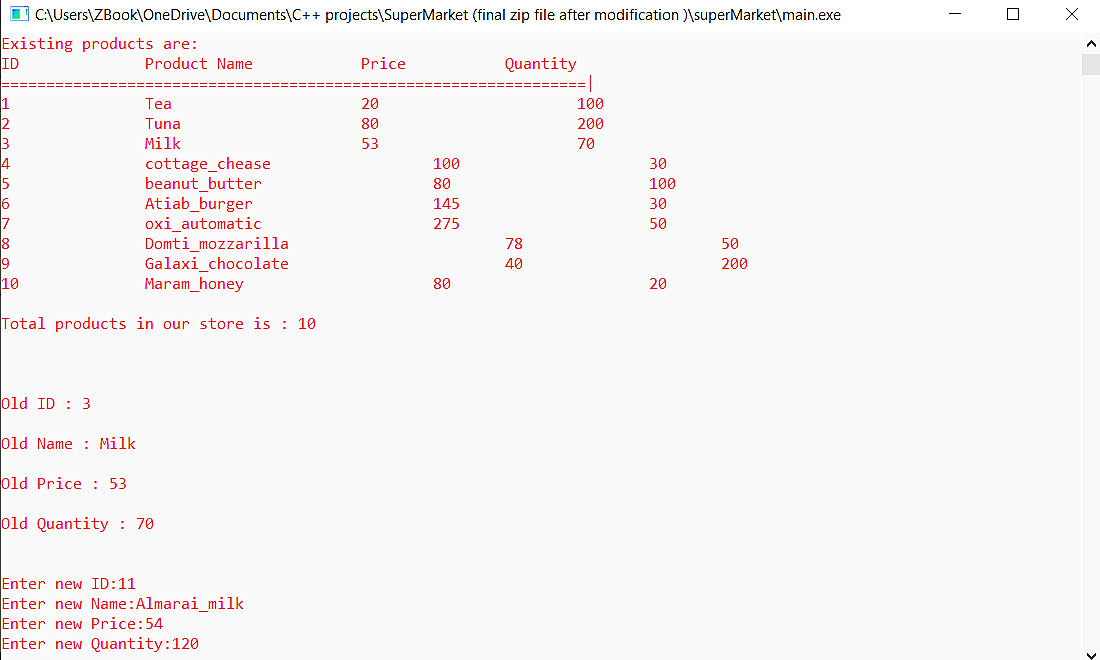


If the product is inserted then it prints “product is inserted” message. It again shows all operations in stock module:

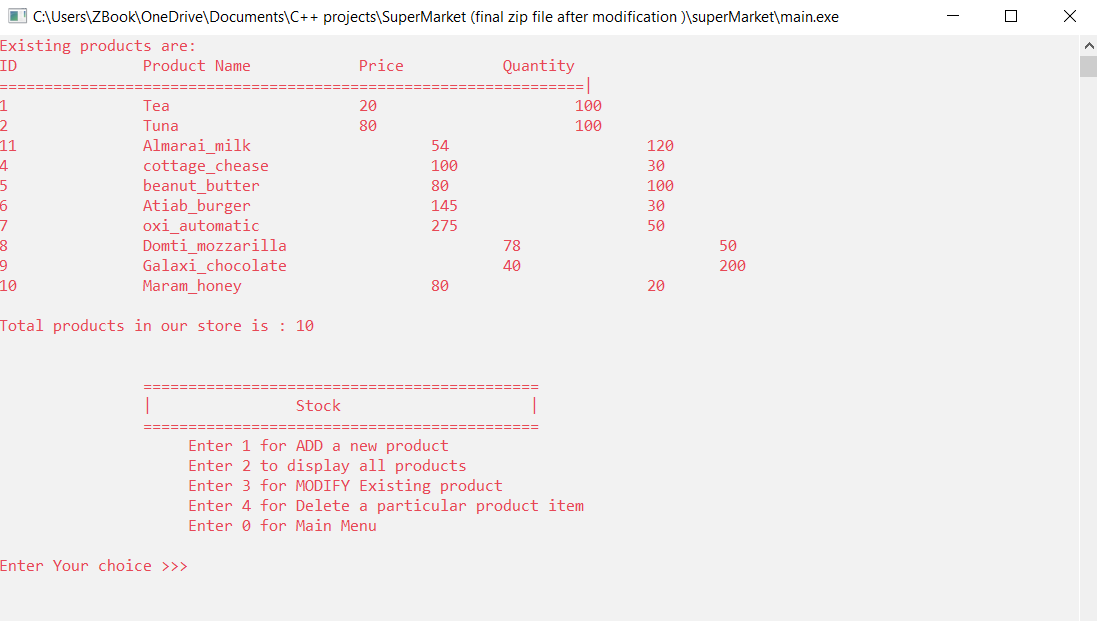


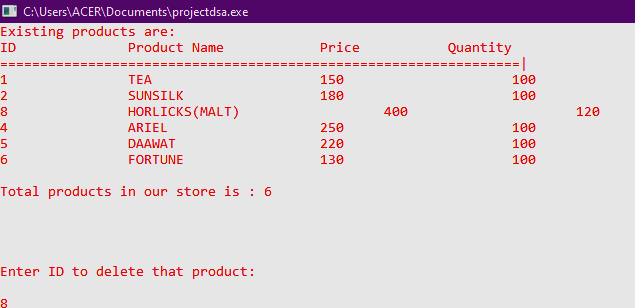
All the products that were inserted in the stock can be retrieved and displayed by this operation:

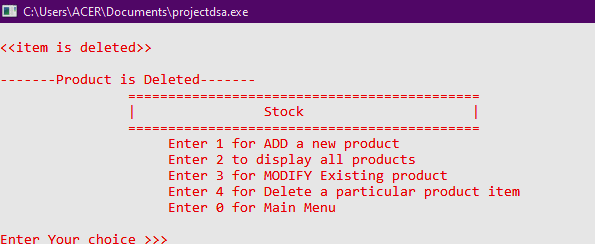
Modification of any details of the inserted product can be done by this operation:



Modifications can be observed and verified by again displaying products in the stock &

if you have to remove any product from the stock, deletion operation can be used:

If we have to delete (Domti\_mozzarilla) from the stock then we have to enter this product’s id (8):

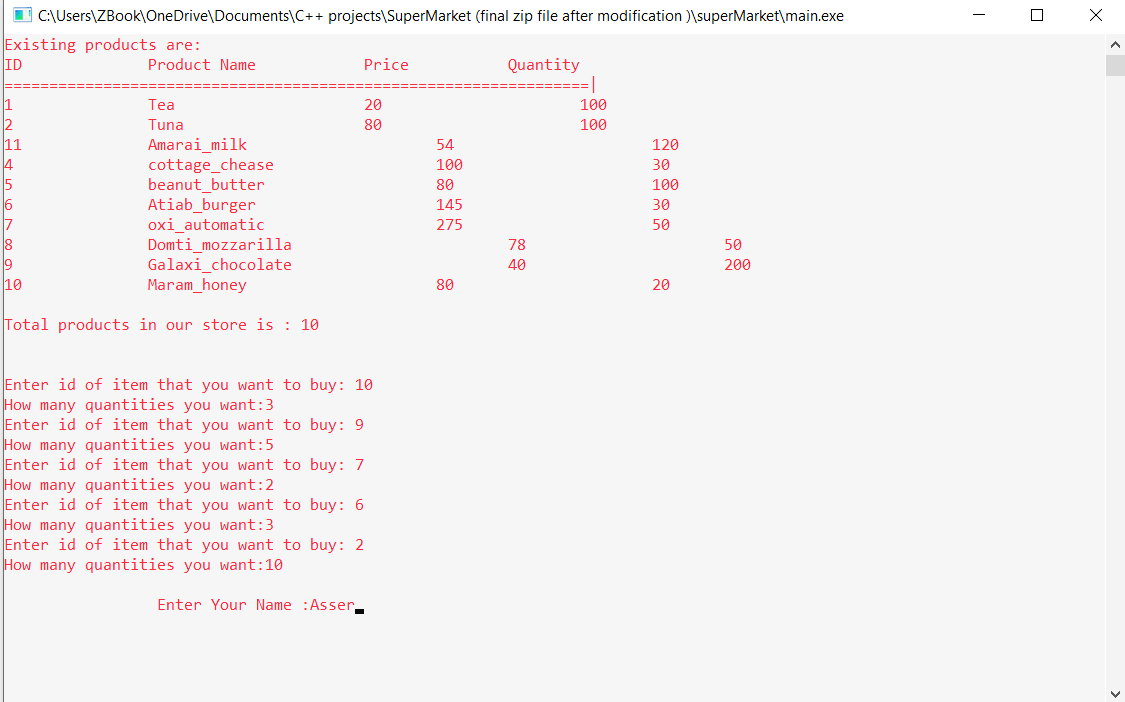
Item deleted:

To return to main menu enter 0 .

### MODULE 2 (CUSTOMER):

First a trolley is allocated to the customer by using stack:

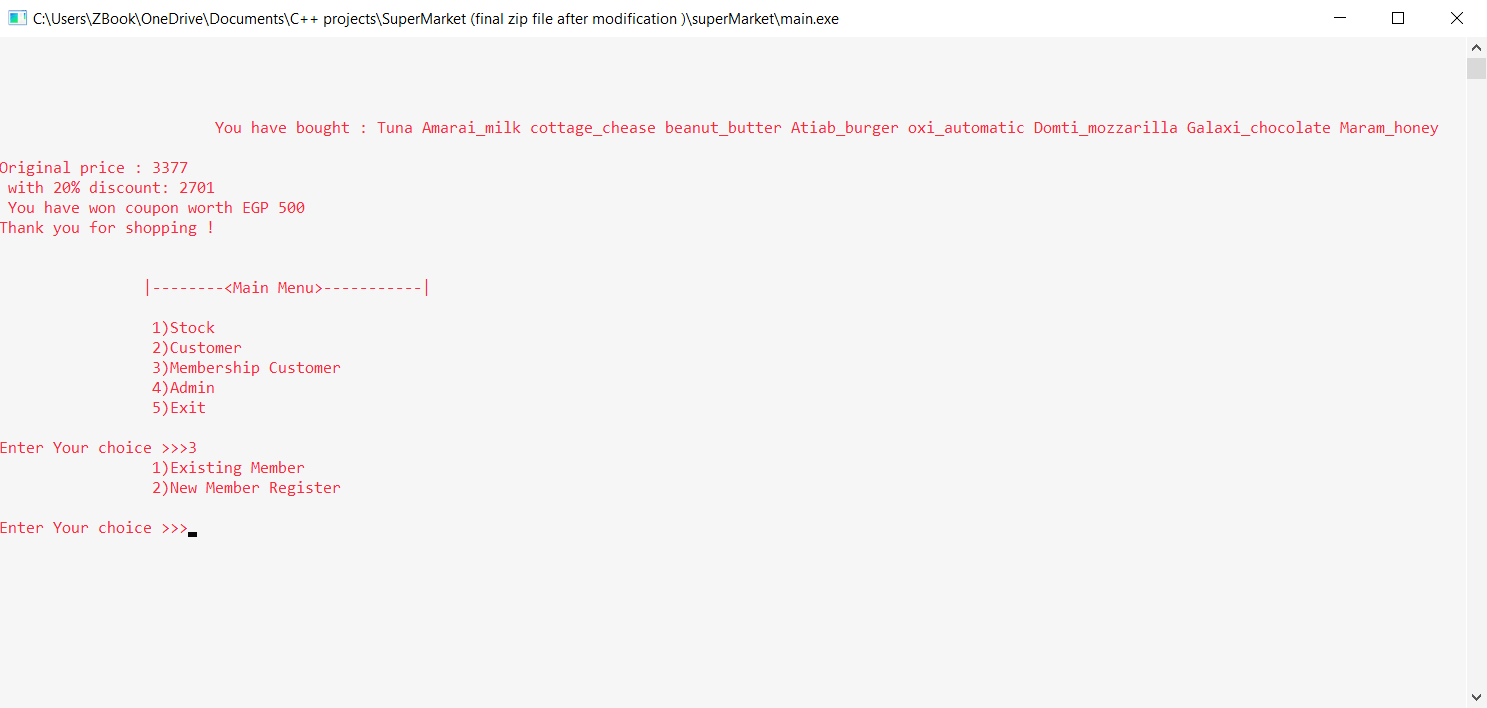
Enter no. of products to be bought, id’s of that product, customer name & if id is not valid it shows that product is unavailable:



Final amount and enter customer details:

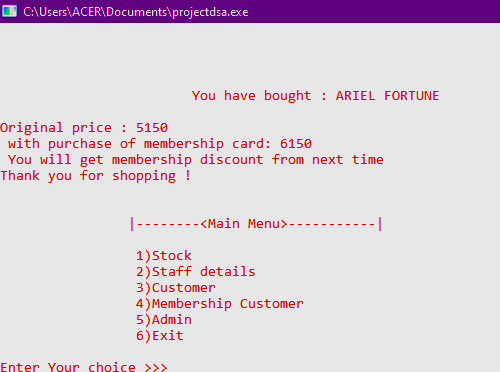


### MODULE 3 (MEMBERSHIP CUSTOMER):

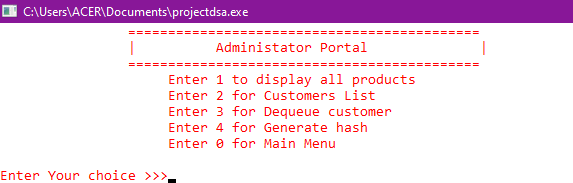
**Case 1 (existing membership card) :** Shopping amount greater than 3000 EGP will get gift coupons of worth 500 EGP. Also, will get a compulsory discount of 20% :

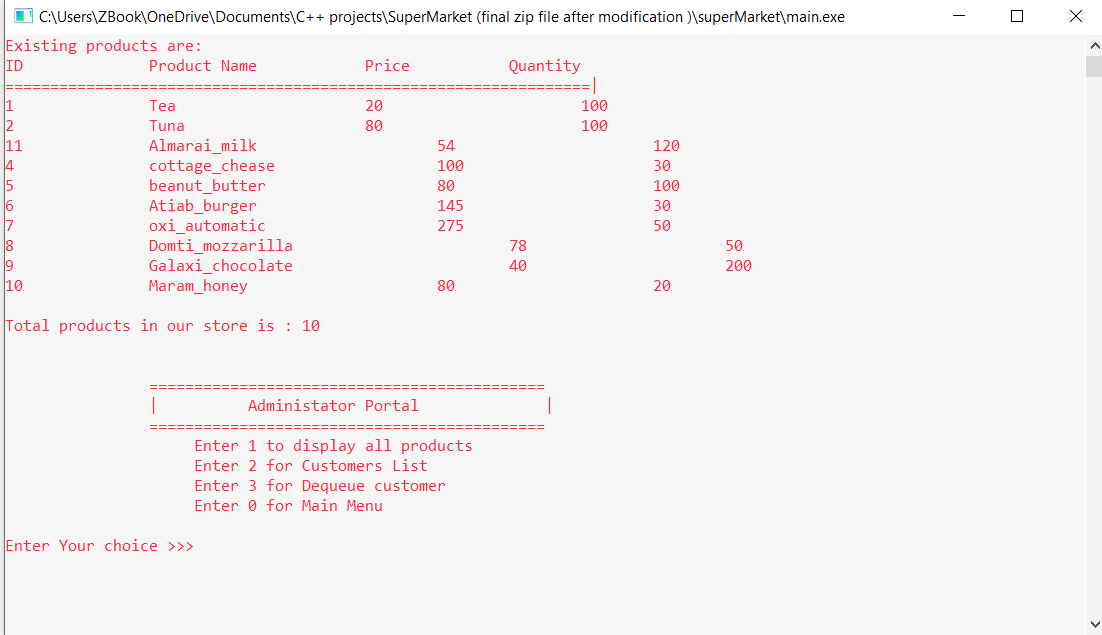
Purchasing less than 3000 will not have any gift coupons :

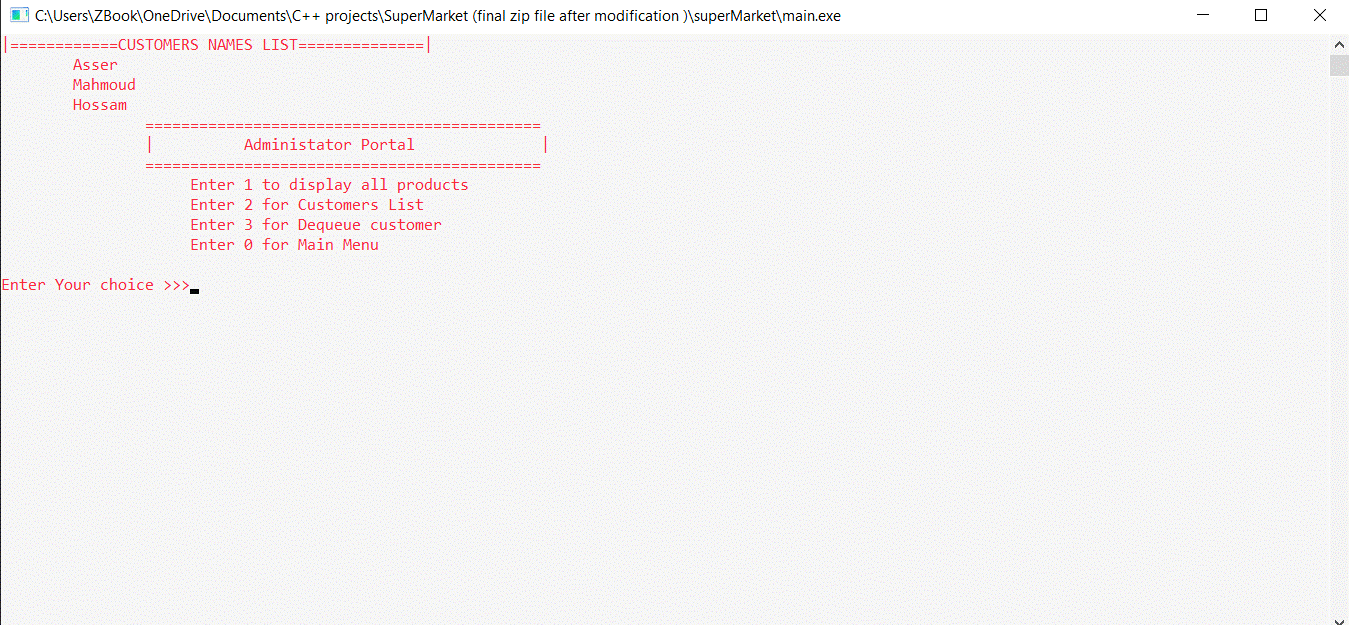
**Case 2 (new membership card) :**

Bill to be paid by new membership card holder is amount purchased +1000 . Also discounts and coupons will be given for their next shopping:

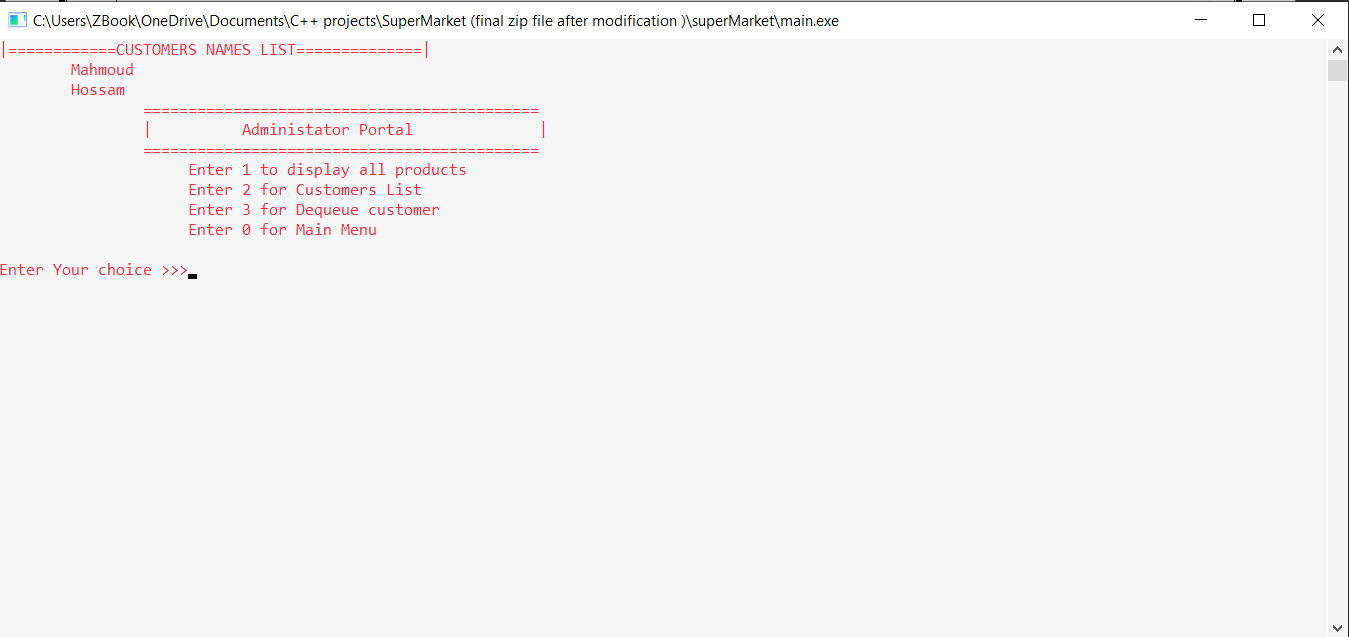
### MODULE 4 (ADMIN):

Admin portal opens as soon as we select this module:

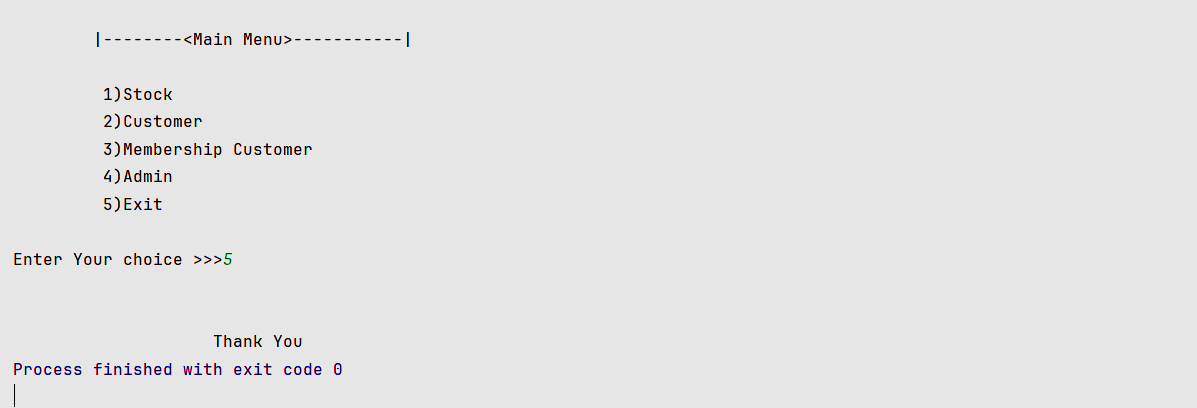
Display of all the products in the stock, because admin should know so as to add items if they are out of stock:

Customer’s list:

If there is no customers, then it prints “queue is empty”:

Dequeue any customer from the line (waiting):

**Module 5 :** Enter (5) to exit and stop the program:



**Link of the code** (uploaded on google drive):

<https://drive.google.com/file/d/14dUyGkz_jKpCUVCreuLqyiZHZ9nhf7KU/view?usp=sharing>

# Time complexity:

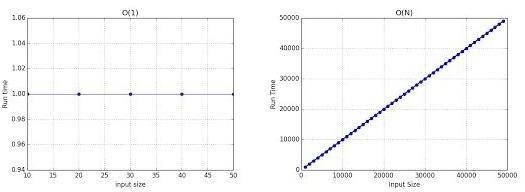
Time complexity is a very important aspect. Individuals prefer faster services due to the rapid development of technology. This demands faster operations from this program compared to other programs which were developed for the same purpose. Hence, time complexity has been calculated and the same has been explained below.

**Stack:**

The worst case of the insertion (trolley allocation) and deletion (deleting the trolley allocated after the customer has shopped) operations of stacks is O(1).

## Linked list:

The worst case of the insertion and deletion operations of linked lists is O(1). The search operation is O(n).

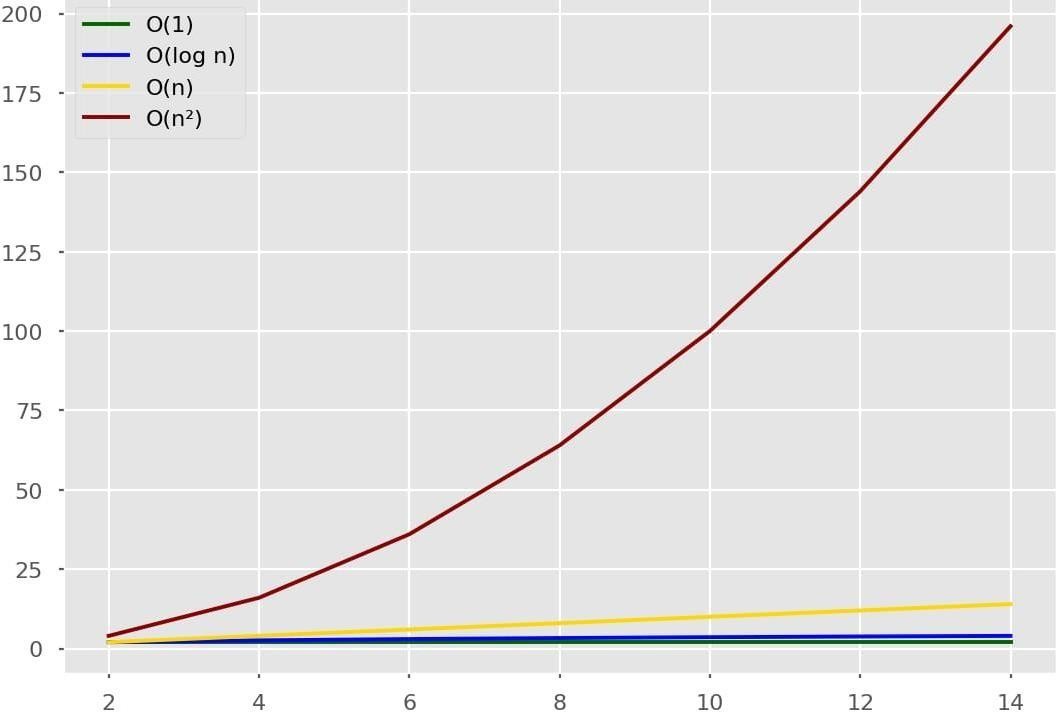


## Array:

The worst case of the insertion and deletion operations of array is O(n). The search operation is O(n).

## Queues:

The worst case of the insertion and deletion operations of queue is O(1). The search operation is O(n).



# References

1. <https://www.bing.com/ck/a?!&&p=e8605020cddb8ccaJmltdHM9MTcyMzQyMDgwMCZpZ3VpZD0wZGU3M2I1NC00MDgyLTYxNTUtMjg2My0yZmVjNDE1ZjYwMWEmaW5zaWQ9NTM3OQ&ptn=3&ver=2&hsh=3&fclid=0de73b54-4082-6155-2863-2fec415f601a&psq=data+structures+via+c%2b%2b+the+most+modern+referrences&u=a1aHR0cHM6Ly9jczYxLnNlYXMuaGFydmFyZC5lZHUvc2l0ZS8yMDIxL1NlY3Rpb24xLw&ntb=1>
2. Pointers in C and C++. (2020, July 14). Journal Dev. Retrieved from: https://[www.journaldev.com/30481/pointers-in-c-and-c-plus-plus](http://www.journaldev.com/30481/pointers-in-c-and-c-plus-plus)
3. Vanessa Cross.(September 26, 2017). How to Run a Supermarket Business. Retrieved from: [https://bizfluent.com/how-6808955 runsupermarket-business.html](https://bizfluent.com/how-6808955%20runsupermarket-business.html).
4. Reema, Thareja.(2014). Data structures using C, 2nd edition. Oxford University Press.
5. Narasimha, Karumanchi.(2014). Data Structures and Algorithms Made Easy, 2nd ed. Atlantic Publishers and Distributers.
6. Frank, Steeneken & Dave, Ackley. (January 2012). A Complete Model of Supermarket Business. Retrieved from: https://[www.bptrends.com/publicationfiles/01-](http://www.bptrends.com/publicationfiles/01-03-2012-) [03-2012-](http://www.bptrends.com/publicationfiles/01-03-2012-) ARTSupermarket%20Article-steeneken-Ackley%20111226.pdf. BP Trends
7. Krishnamoorthy, R., & Kumaravel, G. I. (2010). Data structures using C. New Delhi: Tata McGraw-Hill education Private.
8. Sahni, Sartaj. (2005). Data Structures, Algorithms and Applications in C++,2nd ed. Universities Press.
9. Wegner, Peter; Reilly, Edwin D. (2003-08-29). Encyclopedia of Computer Science. Chichester, UK: John Wiley and Sons.
10. Sara Baase, Allen Van Gelder. (1999). Computer Algorithms, Introduction to Design and Analysis, 3rd edition. Wesley Longman Publishing.
11. Thomas, H & Cormen Charles, E & Leiserson Ronald, L&Rivest Clifford, Stein. (May 2001). Introduction\_ to\_ Algorithm. The MIT Press, McGraw-Hill Book Company.
12. Ellis, Horowitz.(1983). Fundamentals of Data Structures in C, 2nd Edition. Computer Science Press. retrieved from: https:[//ww](http://www.geeksforgeeks.org/pointers-andreferences-in-)w.[geeksforgeeks.org/pointers-andreferences-in-](http://www.geeksforgeeks.org/pointers-andreferences-in-)c/
13. Lipschutz, S. (101-). Data structures (SOS) (Revised first edition). McGraw- Hill Education.