



EAST WEST UNIVERSITY

## **Project Report**

**Project Title:** Online Food Delivery

Course Name: Data Structures

Course Code: CSE207

Section: 1

### **Submitted To**

Dr. Waselul Haque Sadid, PhD

Assistant Professor, Department of CSE.

### **Submitted By**

Syed Mahmud Ahmed, ID: 2017-2-60-153.

Nilofa Yeasmin, ID: 2017-2-60-151.

Md. Ridwan Sarker Turza, ID: 2017-2-60-135.

**Date of Submission:** 30 August, 2018.

## CSE207 – Data Structures

# Project: Online Food Delivery

### Project Description:

There is a food delivery organization that offers many kinds of foods for home delivery and each client give an order to buy food. Its employees will search for the required food upon customers' request in First Come First Served basis. The food delivery organization has the following requirements:

- a. There are many kinds of foods in the food delivery organization's inventory.
- b. The customers have to organize as FCFS basis.

In this project we had to do the following:

1. Reserve different kinds of foods' detail in linked list.
2. Store the information of customer's order in queue.
3. Store the information of ordered food of per customer in stack.
4. Build report that will show selling reports of foods.

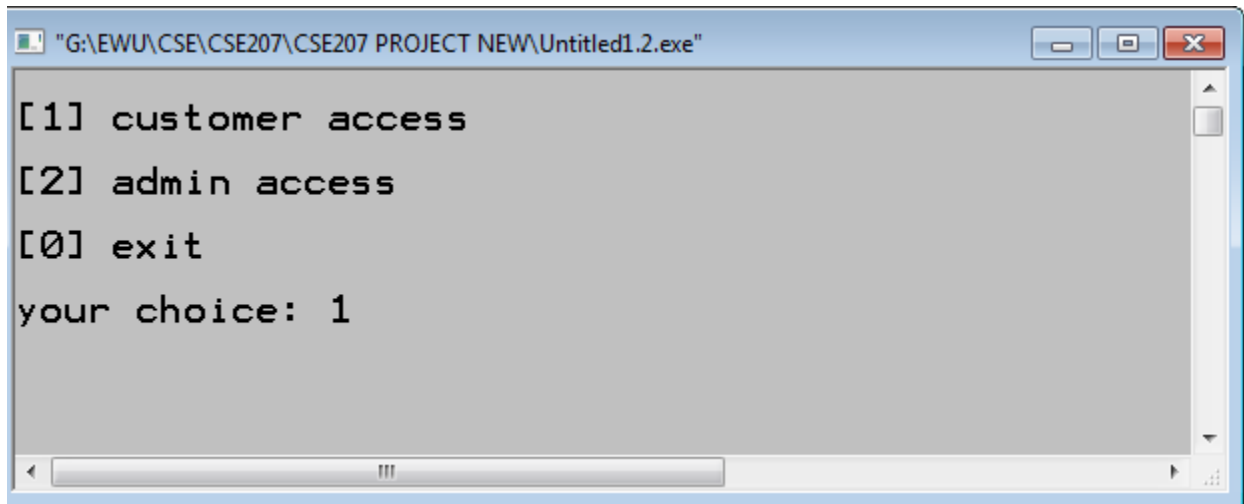
### Method:

We stored the food-menu in a **linked list** where each **node**, dubbed 'item', contained information about a food item.

To store each customer's information we used a **class** named 'customer'. We then stored these 'customer' type objects in a **queue**. Using queues ensured the First Come First Served (**FCFS**) basis as instructed.

We stored each customer's orders in separate **stacks**. For each customer we simply copied selected 'item' nodes from the linked list and pushed them into a stack. We then stored the **top** pointer of that stack in its specific customer object.

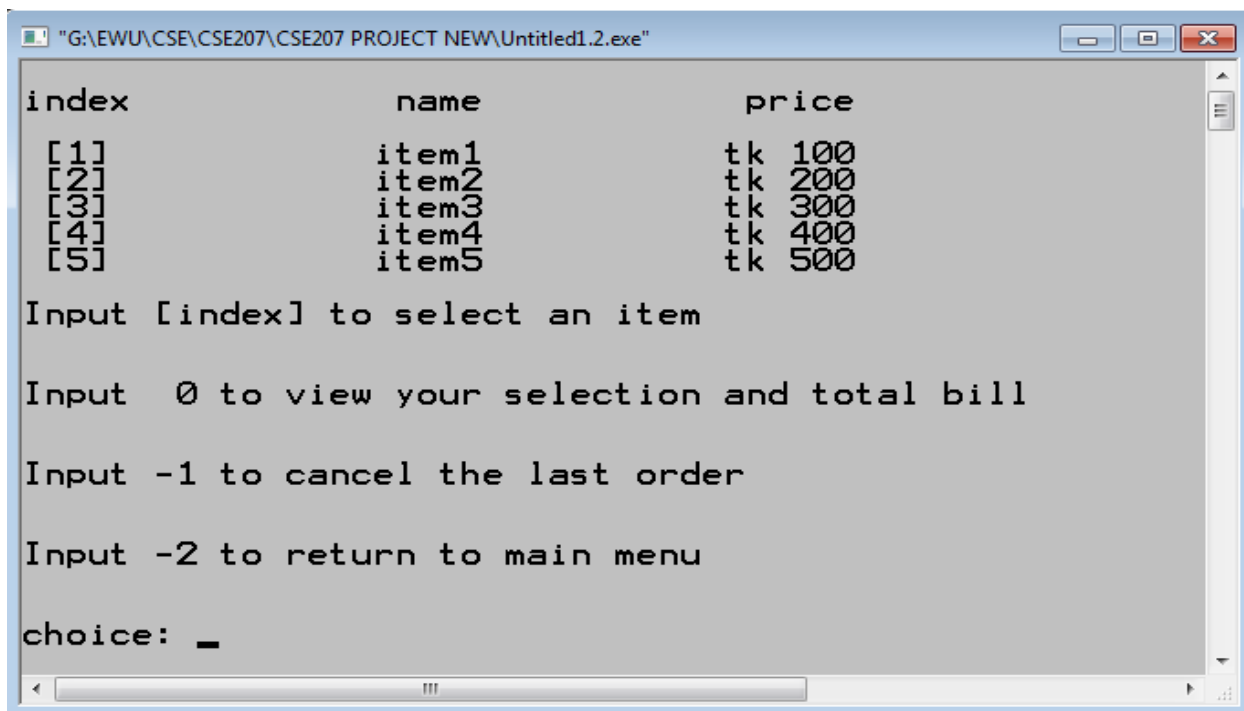
## Program Output:



```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"

[1] customer access
[2] admin access
[0] exit
your choice: 1
```

Once we run the program we are presented with the 'main-menu' which allows for two types of access, namely customer and admin (or employee) access.



```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"

index          name          price
[1]            item1        tk 100
[2]            item2        tk 200
[3]            item3        tk 300
[4]            item4        tk 400
[5]            item5        tk 500

Input [index] to select an item

Input 0 to view your selection and total bill

Input -1 to cancel the last order

Input -2 to return to main menu

choice: _
```

Once we select 'customer access', we are presented with a list of food items available in the online food delivery organization's inventory. The food items and their details are stored in a **file** (menu.txt) and once the program is started all data is read from the file and stored in a **linked list**. Instructions are given at the bottom of the food-menu. To select a food item, one has to simply input its index.

```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"

index          name          price
[1]            item1         tk 100
[2]            item2         tk 200
[3]            item3         tk 300
[4]            item4         tk 400
[5]            item5         tk 500

selected: item #4
specify quantity = 2
order(s) added
Press any key to continue . . .
```

Once a food item is selected, we can specify how many quantities of that specific item we wish to order. Once the quantity is specified, orders are added to the customer's selection.

```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"

index          name          price
[1]            item1         tk 100
[2]            item2         tk 200
[3]            item3         tk 300
[4]            item4         tk 400
[5]            item5         tk 500

Input [index] to select an item

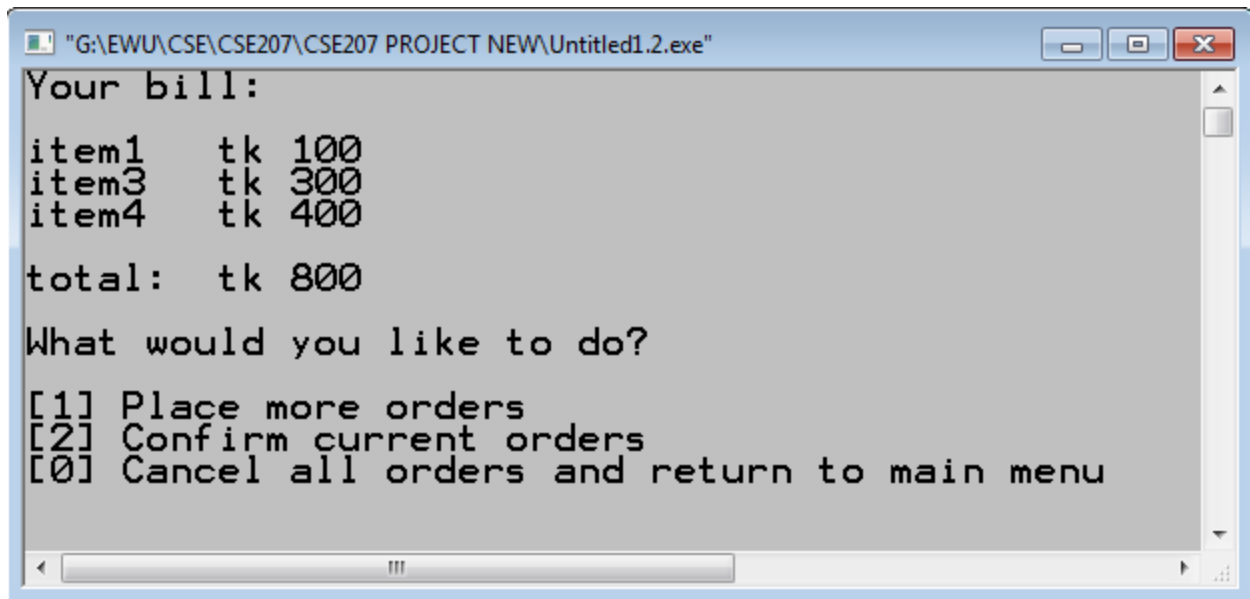
Input  0 to view your selection and total bill

Input -1 to cancel the last order

Input -2 to return to main menu

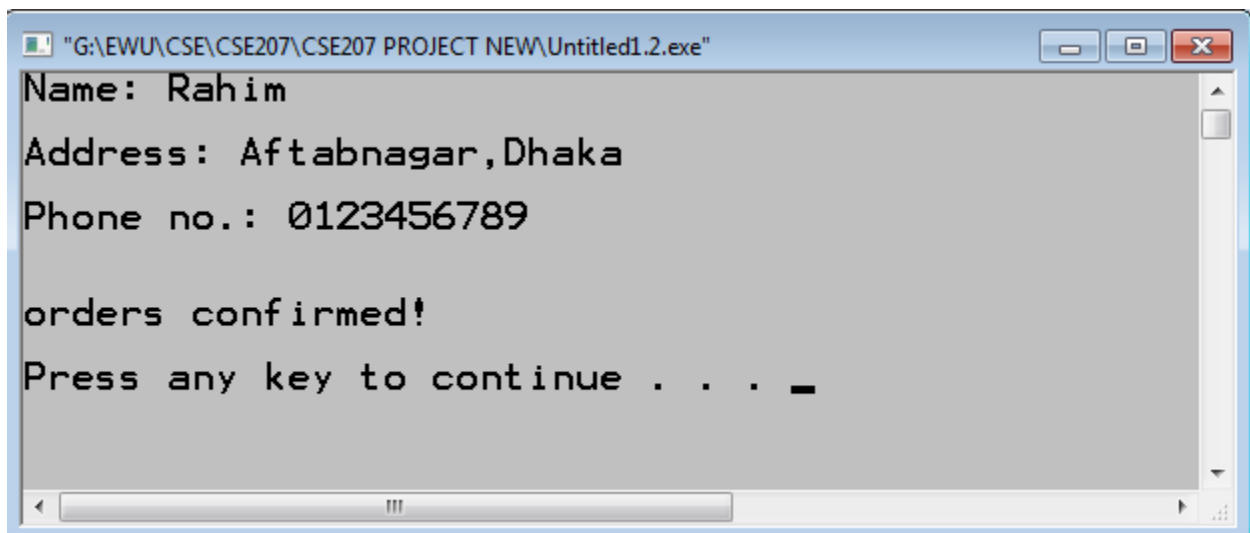
choice: _
```

After ordering we can press **0** to view our selections and bill.



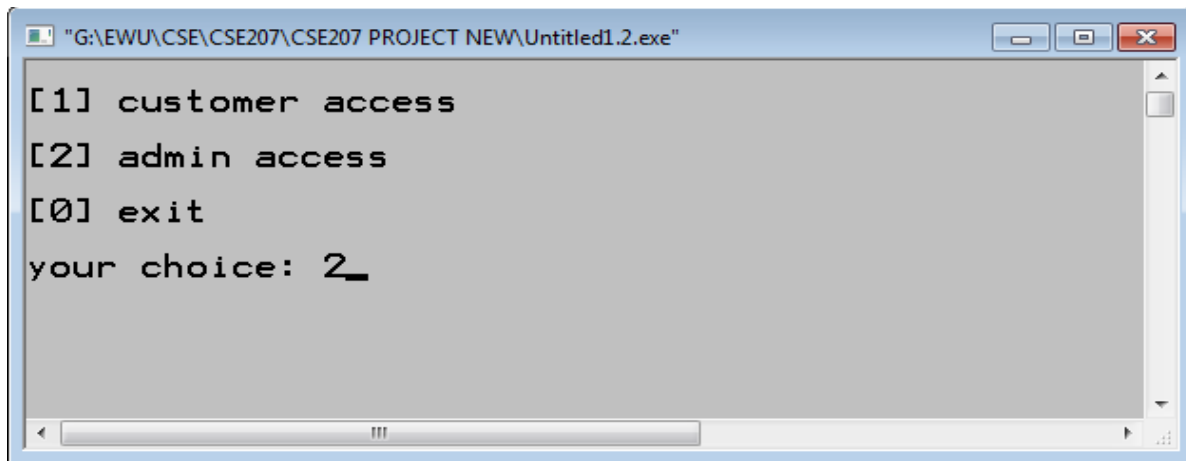
```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"
Your bill:
item1    tk 100
item3    tk 300
item4    tk 400
total:   tk 800
What would you like to do?
[1] Place more orders
[2] Confirm current orders
[0] Cancel all orders and return to main menu
```

From this sub-menu we are given the choice to place more orders, confirm current ones or cancel them. The orders of **each customer** are stored in **separate stacks**. To do this, we created a **class** called 'customer' and as it's attribute we kept a **pointer** to specify the "top" of a customer's stack of orders.



```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"
Name: Rahim
Address: Aftabnagar,Dhaka
Phone no.: 0123456789
orders confirmed!
Press any key to continue . . . _
```

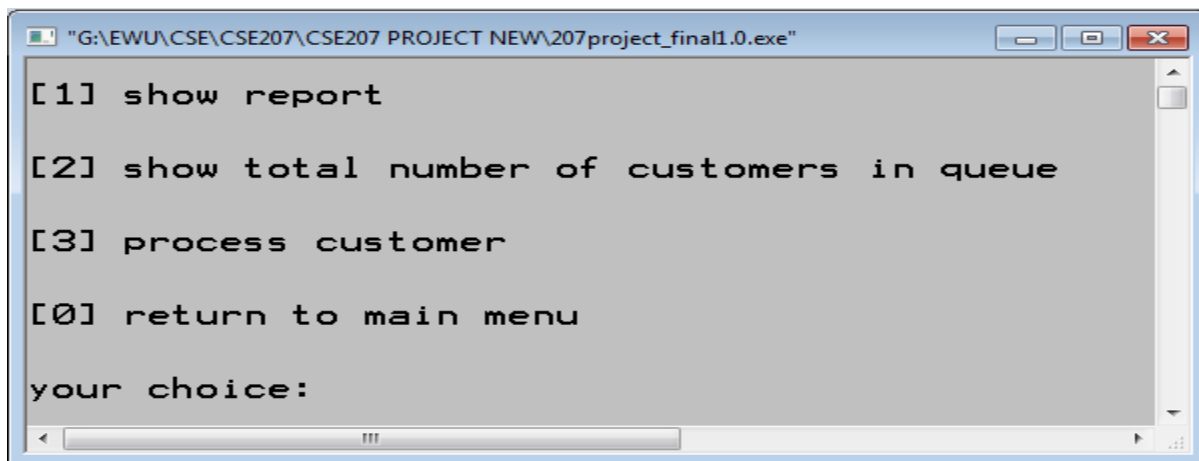
If we choose to confirm our orders, we are requested to fill out additional information about the customer. Once all the information is given, it is stored in an **object** of 'customer' class. The "**top**" pointer of the customer's order-stack is also stored in this object. This object which contains all the information about a customer and his orders is then stored in a **queue**.



```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"

[1] customer access
[2] admin access
[0] exit
your choice: 2_
```

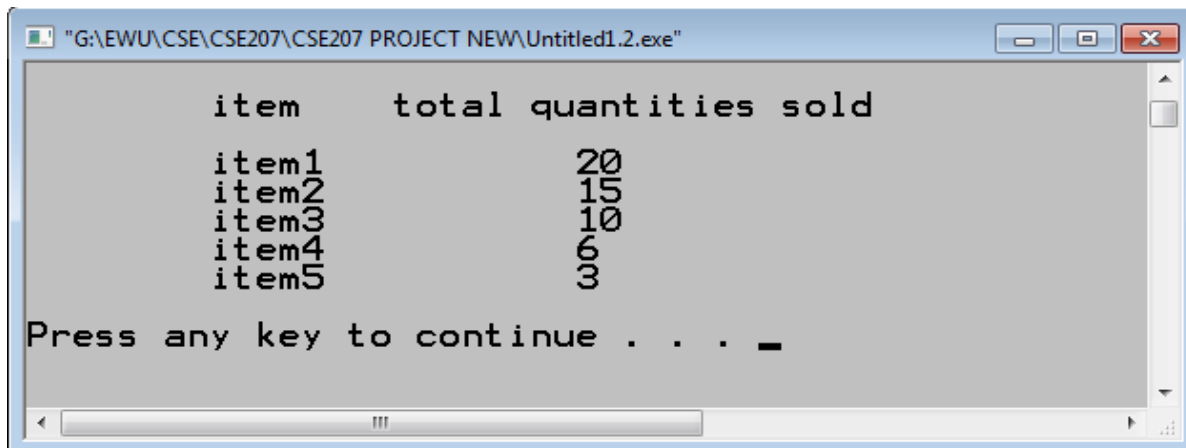
On the other hand, we can choose admin access.



```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\207project_final1.0.exe"

[1] show report
[2] show total number of customers in queue
[3] process customer
[0] return to main menu
your choice:
```

Admin access gives us an option called “show report”.



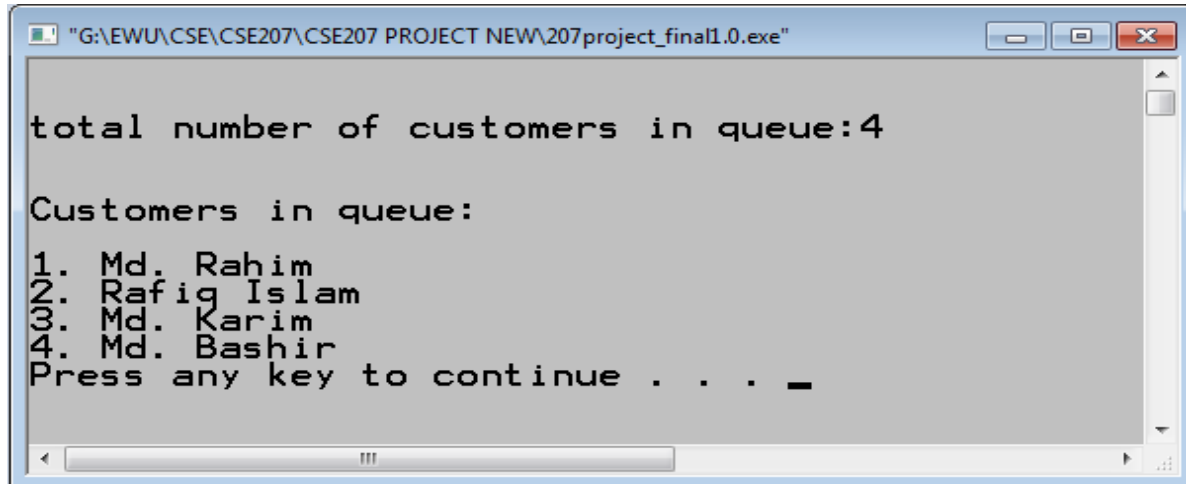
```
"G:\EWU\CSE\CSE207\CSE207 PROJECT NEW\Untitled1.2.exe"

item    total quantities sold
item1   20
item2   15
item3   10
item4    6
item5    3

Press any key to continue . . . _
```

This option lets us view the total quantities of each items sold. When we choose to exit the program, this information is stored in a file to preserve the data.

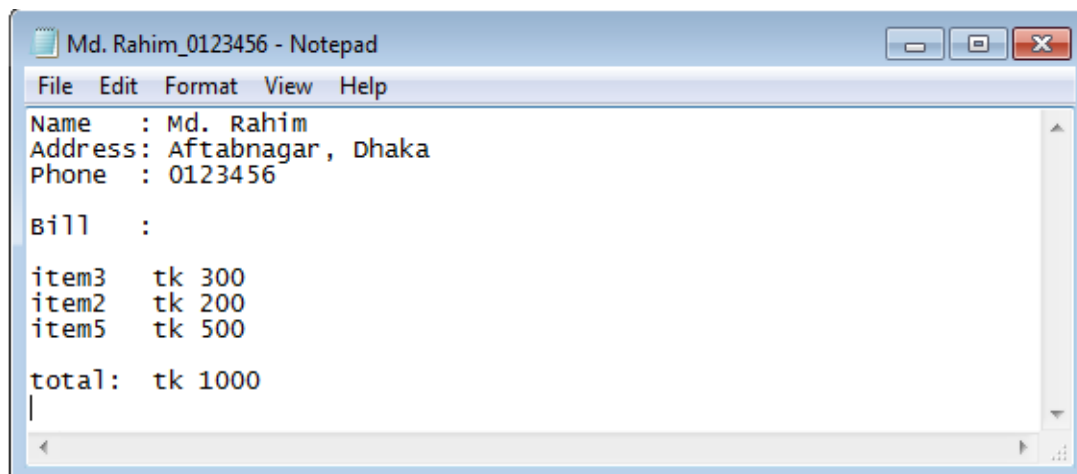
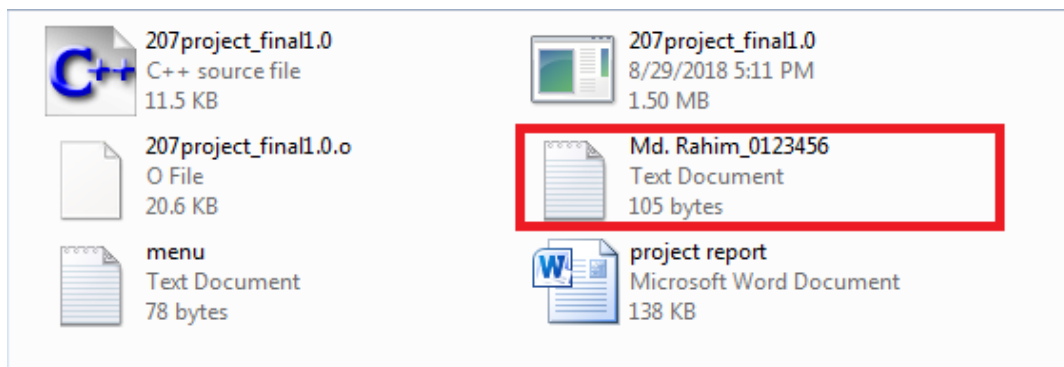
Option [2] shows the total number of customers in queue and their names.



```
total number of customers in queue:4

Customers in queue:
1. Md. Rahim
2. Rafiq Islam
3. Md. Karim
4. Md. Bashir
Press any key to continue . . . _
```

Finally, selecting option [3], 'process customer', will process the order of the customer at the front of the queue. All of his information and bill will be saved in a separate text file and the customer will be removed from the queue.



```
File Edit Format View Help
Name : Md. Rahim
Address: Aftabnagar, Dhaka
Phone : 0123456

Bill :

item3 tk 300
item2 tk 200
item5 tk 500

total: tk 1000
|
```

## Source Code:

```
#include <iostream>

#include <fstream>

#include <stdlib.h>


using namespace std;


struct item
{
    int index, count;

    string name;

    double price;

    struct item *next;
}*menu_head = NULL; //head pointer of the linked list that holds the food menu


class customer
{
public:
    string name, address, phone;

    int order_count;

    double bill;

    item *ostack_top;

    customer *next; //points to the next customer in queue
```



```
customer()
{
    next = NULL;
    ostack_top = NULL;
    bill = 0;
    order_count = 0;
}

void push_order(int, int);
item* pop_order();
}*cust_qfront = NULL, *cust_qrear = NULL; //the front & rear pointers of the customer queue
```

```
int menu_count = 0; //total number of items available
int cust_count = 0; //total number of customers in queue
```

```
void customer_menu();
void admin_menu();
void show_report();
void save_data();
```

```
void enqueue_customer(customer *newcust)
{
```

```

if(cust_qfront == NULL)
{
    cust_qfront = newcust;
    cust_qrear = newcust;
}
else
{
    cust_qrear->next = newcust;
    cust_qrear = newcust;
}
cust_count++;
}

```

```

void dequeue_customer()
{
    customer *temp;

    if(cust_qfront == NULL) //no customers in queue
    {
        system("cls");
        cout<<"\nthere are no customers in queue!\n"<<endl;
    }
    else if(cust_count == 1) //only 1 customer in queue
    {

```

```

    temp = cust_qfront;

    cust_qfront = NULL;

    cust_qrear = NULL;

    delete temp;

    cust_count--;

}

else

{

    temp = cust_qfront;

    cust_qfront = cust_qfront->next;

    delete temp;

    cust_count--;

}

}

void list_cust()

{

    customer *temp = cust_qfront;

    int i = 1;

    while(temp != NULL)

    {

        cout<<i<<" " <<temp->name<<endl;

        temp = temp->next;

        i++;

```

```

    }
}

void customer::push_order(int index, int quantity)
{
    for(int loop = 0; loop < quantity; loop++)
    {
        struct item *temp1 = menu_head;
        struct item *temp2 = new item;

        order_count++;

        while(temp1 != NULL)
        {
            if(temp1->index == index)
            {
                temp2->index = temp1->index;
                temp2->name = temp1->name;
                temp2->price = temp1->price;

                temp1->count++;
                bill = bill+temp1->price;
                break;
            }

```

```
        temp1 = temp1->next;
    }

    if(ostack_top == NULL)
    {
        temp2->next = NULL;
        ostack_top = temp2;
    }
    else
    {
        temp2->next = ostack_top;
        ostack_top = temp2;
    }
}
```

```
item* customer::pop_order()
{
    struct item *temp;
    temp = ostack_top;

    if(temp != NULL)
    {
        system("cls");
```

```

        cout<<"You haven't placed any orders yet!\n"<<endl;
        return NULL;
    }
else
{
    temp = temp->next;

    struct item *pop = new item;

    pop = temp;

    bill = bill - pop->price; ///updating bill total

    delete temp;
    return pop;
}

}

void insert_item(int index, string name, double price, int count)
{
    if(index < 0)
    {
        cout<<"Invalid index"<<endl;
        return;
    }

```

```

int currIndex      =      1;

    item* currNode =      menu_head;

while (currNode!=NULL && index > currIndex) {

    currNode      =      currNode->next;

    currIndex++;

}

    if (index > 0 && currNode == NULL)
{
    cout<<"Invalid index"<<endl;

    return;

}

    item* newNode = new item;

    newNode->index = index;

    newNode->name = name;

    newNode->price = price;

    newNode->count = count;

    if (index == 0) {

        newNode->next      =      menu_head;

        menu_head      =      newNode;

```

```

    }

    else {

        newNode->next    =    currNode->next;

        currNode->next    =    newNode;

    }

}

```

```

void print_foodlist()

{

    struct item *temp = menu_head;


    cout<<"\nindex\t\t name\t\t price\t\n"<<endl;

    while(temp != NULL)

    {

        cout << " [" << temp->index+1 << "]\t";

        cout << "\t" << temp->name  << "\t";

        cout << "\t\t" << temp->price  << "\t";

        cout << endl;

        temp = temp->next;

    }

}

```

```

void process_customer() //processes the order of the customer at the front of the queue

                        //and writes his information in a separate .txt file

```



```

{
    system("cls");

    ofstream outfile;

    string filename;

    customer *cust = cust_qfront;


    filename = cust->name + "_" + cust->phone + ".txt";
    outfile.open(filename.c_str());


    outfile << "Name  : " << cust->name << endl;
    outfile << "Address: " << cust->address << endl;
    outfile << "Phone  : " << cust->phone << endl<< endl;
    outfile << "Bill   : " << endl;
    outfile<< endl;


    item *temp = cust->ostack_top;
    while(temp != NULL)
    {
        outfile<<temp->name<<"\ttk " <<temp->price<< endl;

        temp = temp->next;
    }

    outfile<<"\ntotal:\ttk " <<cust->bill;

    outfile<< endl;

```

```
    outfile.close();

    dequeue_customer();

    cout<<"\norders have been processed!"<<endl;

    system("pause");

    return;
}
```

```
void main_menu()
{
    int choice;

    system("cls");

    cout<<endl;

    cout<<"[1] customer access\n"<<endl;

    cout<<"[2] admin access\n"<<endl;

    cout<<"[0] exit\n\n";

    cout<<"your choice: ";

    cin>>choice;

    switch(choice)
    {
    case 1:

        system("cls");

        customer_menu();
```

```
        break;
    case 2:
        system("cls");
        admin_menu();
        break;
    case 0:
        system("cls");
        save_data();
        exit(1);
    }
}
```

```
void customer_menu()
{
    customer *cust = new customer;
    int choice;
    int quantity = 0;
    while(1)
    {
        system("cls");
        print_foodlist();
        cout<<"\nInput [index] to select an item\n"<<endl;
        cout<<"\nInput 0 to view your selection and total bill\n"<<endl;
        cout<<"\nInput -1 to cancel the last order\n"<<endl;
```

```
cout<<"\nInput -2 to return to main menu\n"<<endl;
```

```
cout<<"\nchoice: ";
```

```
cin>>choice;
```

```
cout<<endl;
```

```
if(choice > 0 && choice <= menu_count)
```

```
{
```

```
    system("cls");
```

```
    print_foodlist();
```

```
    cout<<"\nselected: item #"<<choice<<endl;
```

```
    cout<<"\nspecify quantity = ";
```

```
    cin>> quantity;
```

```
    cust->push_order(choice-1, quantity);
```

```
    cout<<"\norder(s) added\n"<<endl;
```

```
    system("pause");
```

```
}
```

```
else if(choice == 0)
```

```
{
```

```
    if(cust->order_count == 0)
```

```
    {
```

```
        system("cls");
```

```
        cout<<"\nNo orders taken.\n"<<endl;
```

```
    }
```

```
else
```

```
{  
  
    system("cls");  
  
    item *temp;  
  
    temp = cust->ostack_top;  
  
    cout<<"Your bill:\n"<<endl;  
    while(temp != NULL)  
    {  
        cout<<temp->name<<"\ttk "<<temp->price<<endl;  
        temp = temp->next;  
    }  
  
    cout<<"\ntotal:\ttk "<<cust->bill;  
    cout<<endl<<endl;  
  
    cout<<"What would you like to do?\n"<<endl;  
    cout<<"[1] Place more orders"<<endl;  
    cout<<"[2] Confirm current orders"<<endl;  
    cout<<"[0] Cancel all orders and return to main menu"<<endl;  
    cout<<"\nYour choice:";  
  
    int choice;  
  
    cin>>choice;  
  
    cout<<endl;
```

```
if(choice == 1) continue;
```

```
else if(choice == 2)
```

```
{
```

```
    system("cls");
```

```
    cout<<"Name: ";
```

```
    cin.ignore();
```

```
    getline(std::cin, cust->name);
```

```
    cout<<endl;
```

```
    cout<<"Address: ";
```

```
    getline(std::cin, cust->address);
```

```
    cout<<endl;
```

```
    cout<<"Phone no.: ";
```

```
    getline(std::cin, cust->phone);
```

```
    cout<<endl;
```

```
    enqueue_customer(cust); ///enqueue customer
```

```
    cout<<"\norders confirmed!\n"<<endl;
```

```
    system("pause");  
    return;  
}
```

```
else if(choice == 0)  
{  
    system("cls");  
    cout<<"\norders canceled!\n"<<endl;  
    system("pause");  
    return;  
}
```

```
else  
{  
    cout<<"\ninvalid selection\n"<<endl;  
}  
}
```

```
    system("pause");  
}  
else if(choice == -1)  
{  
    struct item *pop;  
    pop = cust->pop_order();
```

```

        if(pop != NULL) cout<<"\nitem removed: "<<pop->name<<endl<<endl;

        system("pause");
    }

    else if(choice == -2)
    {
        system("cls");

        return;
    }

    else
    {
        system("cls");

        cout<<"\nInvalid selection\n"<<endl;

        system("pause");
    }
}

return;
}

```

```

void admin_menu()

```

```

{
    while(1)
    {
        system("cls");

        cout<<"\n[1] show report\n"<<endl;
    }
}

```



```

cout<<"\n[2] show total number of customers in queue\n"<<endl;

cout<<"\n[3] process customer\n"<<endl;

cout<<"\n[0] return to main menu\n"<<endl;

cout<<"\nyour choice: ";

int choice;

cin>>choice;


if(choice == 1) show_report();


else if(choice == 2)
{
    system("cls");

    cout<<"\n\ntotal number of customers in queue:" << cust_count<<endl;

    cout<<endl;

    if (cust_count > 0 )
    {
        cout<<"\nCustomers in queue:\n"<<endl;

        list_cust();

    }

    system("pause");
}

else if(choice == 3 && cust_count > 0) process_customer(); //processes the order of the
customer at the front of the queue

else if(choice == 3 && cust_count == 0)

```

```

{
    system("cls");

    cout<<"\nerror: there are no customers in queue to process!\n"<<endl;

    system("pause");
}

else if(choice == 0) return;

else
{
    cout<<"\ninvalid selection"<<endl;

    system("pause");
}
}
}

```

```

void show_report()

```

```

{
    item *temp = menu_head;

    cout<<"\n\titem"<<"\ttotal quantities sold\n"<<endl;

    while(temp != NULL)
    {
        cout<<"\t"<<temp->name<<"\t\t"<<temp->count<<endl;
    }
}

```

```
        temp = temp->next;
    }
    cout<<endl;
    system("pause");
}
```

```
void load_data()
{
    ifstream infile("menu.txt");
    if(infile.is_open())
    {
        int index, count;
        string name;
        double price;

        infile >> index >> name >> price >> count;
        while(true)
        {
            insert_item(index, name, price, count);
            menu_count++;
            infile >> index >> name >> price >> count;
            if(infile.eof()) break;
        }
        infile.close();
    }
}
```

```

    }
}

void save_data()
{
    ofstream outfile("menu.txt");

    item *temp = menu_head;

    if(outfile.is_open())
    {
        while(temp != NULL)
        {
            outfile<<temp->index<<" "<<temp->name<<" "<<temp->price<<" "<<temp->count<<
endl;

            temp = temp->next;
        }

        outfile.close();
    }
}

int main()
{
    load_data();

    while(true) main_menu();

    return 0;
}

```