UTM

UNIVERSITI TEKNOLOGI MALAYSIA

Group Name: BOBOBOY

Group Member:

1. AERON GOH MING LUN A22EC0033
2. TAY SHUN WEI A22EC0110
3. LIN CHONG HUI A22EC0184

Course: DATA STRUCTURE AND ALGORITHMS
Section: 04

Lecturer : Ms. LIZAWATI BINTI MI YUSUF

Table of Contents

Objective

The main objective of BOBOBOY's Restaurant Management System (RMS) is to optimize the order management system with effectiveness features and to minimize the time to get through our ordering system. Our system is created for a full process from ordering to payment session and it can be done via through the system and the customers can get the food in the best condition without wasting any time on waiting or getting used to the system.
All the features, panel and functions are to make sure the customers have good feedback to our restaurant and satisfy not only about the food, but also the whole system and environment of the BOBOBOY restaurant.

Synopsis

In a restaurant, it is most important that RMS should be user friendly and show a detailed menu in a simple way. Besides, the customer's emotion is very important as they want to get the food they ordered as fast as possible. In BOBOBOY restaurant, the RMS contained two main functions in the menu which is sorting and searching. The purpose of implementing these two functions is because we want the customers to look through the menu based on their favorite arrangement such as food name with ascending alphabet arrangement. . After that, if they cannot find what specific food they want on the menu, they can type the details or info about that food on the search panel and it will show the position of that product so next time they can straight away order from the menu without wasting time. Besides, inside the order cart, the customer can add on or delete the item straight away inside the cart so there is no need to go back to the menu to do this action. Inside the cart also got a searching and sorting function which helps customers to get through their order fast and easier. All of these are to make our customers like the ordering system of BOBOBOY restaurants and they feel easy and simple when ordering their desired food.

Problem Analysis

In this project, we will develop a food ordering system in the restaurant. Before implementing the whole system, we need to know the typical ordering system we meet in our daily life and also how it functions from the starting until the food is delivered to customers.

In a typical food ordering system, there should be two main group users, the customers and the admin of the restaurant ordering system. The admin will based on what product had been ordered by the customers and delivered to the kitchen to prepare food ordered. So, for an ordering system, there must be some basic and main functions that need to be performed no matter how we enhance it.

First and foremost, for an ordering system, it must be a menu that showed what kind of dishes and drinks we had offered to the customers. We need to make a list of available options inside the restaurant for the customers to pick what they really desire to have. After that, the option they chose inside the menu should be saved inside one place in the system. This feature is important because it can help the customer to check whether they have misclicked about the food they choose. Besides, this place can also show all the details about the option they chose such as the number they have added, total price and food name or code. They can add or delete the items inside so they no need go back to the menu and reorder. After that, they can confirm their order and get a receipt after payment.

On the other hand, the admin or the kitchen staff would also have several functions that can be used to control the ordering system. Firstly, the order details must be sent to the kitchen staff so they can prepare the food as soon as possible. The kitchen staff should be able to see all the uncompleted orders to make sure that they would not miss out on any customers. The order shown contains the order number and also the order details so kitchen staff can prepare their order in the shortest time. Lastly, after the order is completed and delivered out from the kitchen, the order will be deleted from the list.

After understanding the basic functions of the typical ordering system for both users, we can find some problems inside the typical ordering system and we can make some improvements and implementations to make it more efficient and clear to be used for both main users.

Firstly, for the customers, it should be able to add and delete specific orders inside the cart. In our system, the customers can add their order in the specific places and delete it based on what they wanted. After that, if there are too many orders inside the cart and the customers find it hard to browse through what they had ordered, we provide a searching function inside the system. Customers can search their food based on the name and code we provided inside the cart. The system will show either the food had been ordered or not.

After that, we also add on some features that help the customers to check their payment needs easily. The system provided customers to choose how the receipt they wanted to be displayed based on code, name or foodtype. These three can be shown in the receipt in ascending or descending order. After ordering, the customer is available to check their order status to make sure that their order is sent to the kitchen and prepared or completed. This is to make sure that the customers can have details about their orders so they no need to wait without any information.

For the kitchen staff, we need to solve the problems about the system such as checking the number or order and display the order list. The system will be able to check all the status of order and display the number of orders in the completed list or still preparing list. The details of food such as code, name and unit number can be seen inside the order so the staff no need to make sure about what is the product ordered by the customers. After that, to improve the efficiency of kitchen staff, the completed order should be able to be cleared from the list so that the staff will not miss up any order and only need to focus on orders that have not been completed yet.

After that, for a restaurant, we need to know how many orders are completed in one day. This is easier for the manager of a restaurant to conclude the sales and generate sales reports for everyday.

These are the basic problems that our group members found out and faced throughout the project. All of these problems are solved in an effective way and we also add on the features to make the ordering system more efficient and multi-functional. This project can be developed successfully as we used the data structure concepts stated above, which are the queue data structure and also linked-list for ht menu display.

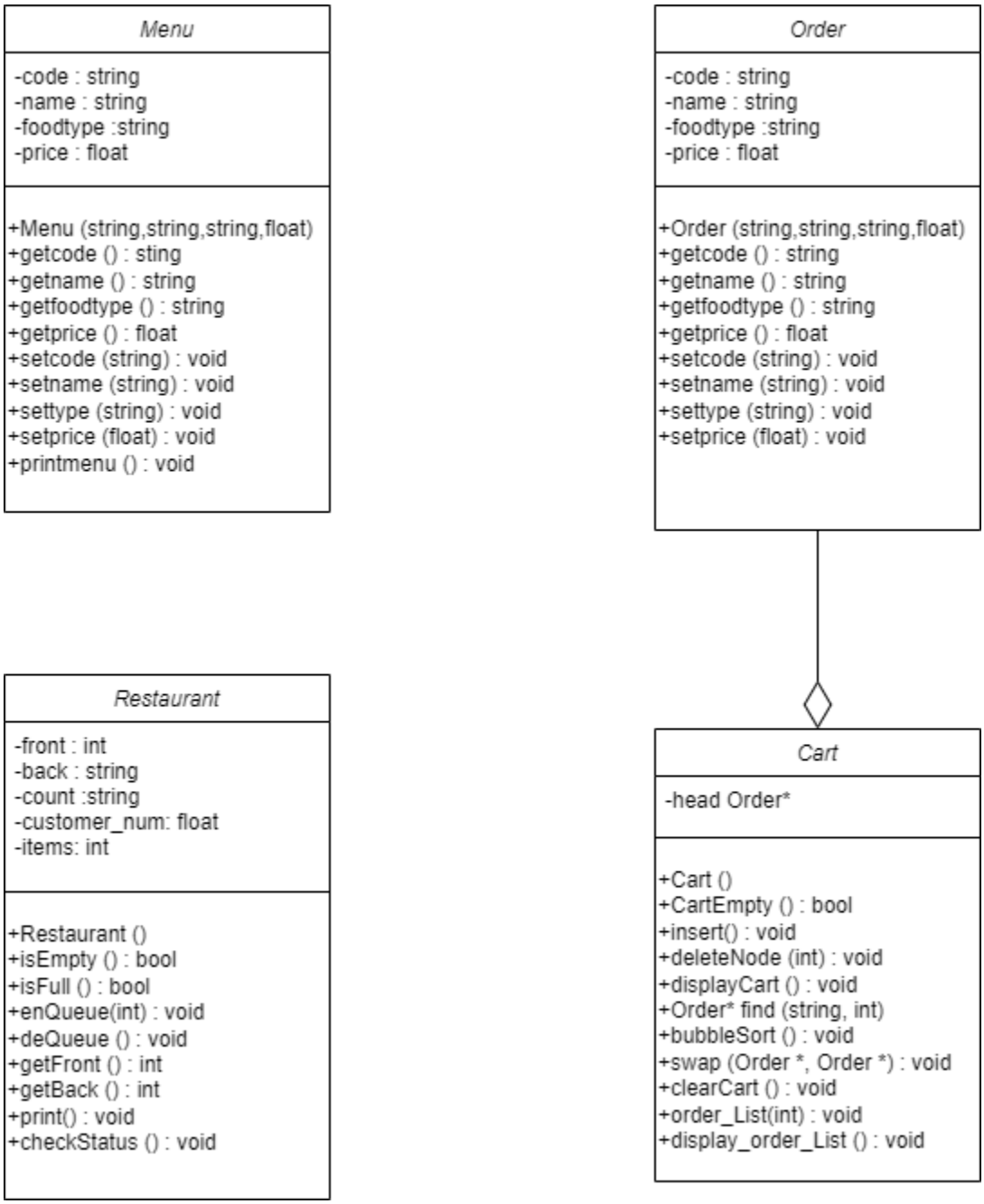
Linked List in Ordering System

In our system, the linked list is used in our system as a cart that can record the order from the customers. We used the linked list data structure for the cart because all the adding and delete functions will be able to be used in the cart. Linked list can control the items inside the cart by adding at front, middle and back. The deletion function also can delete from that three positions based on the implementation of linked list. This helps customers easier to manage their cart and no need to reorder if any misclick occurs. After the confirmation of customers about their food, the order will be sent to the kitchen and payment will come out of the system.

Queue in Ordering System

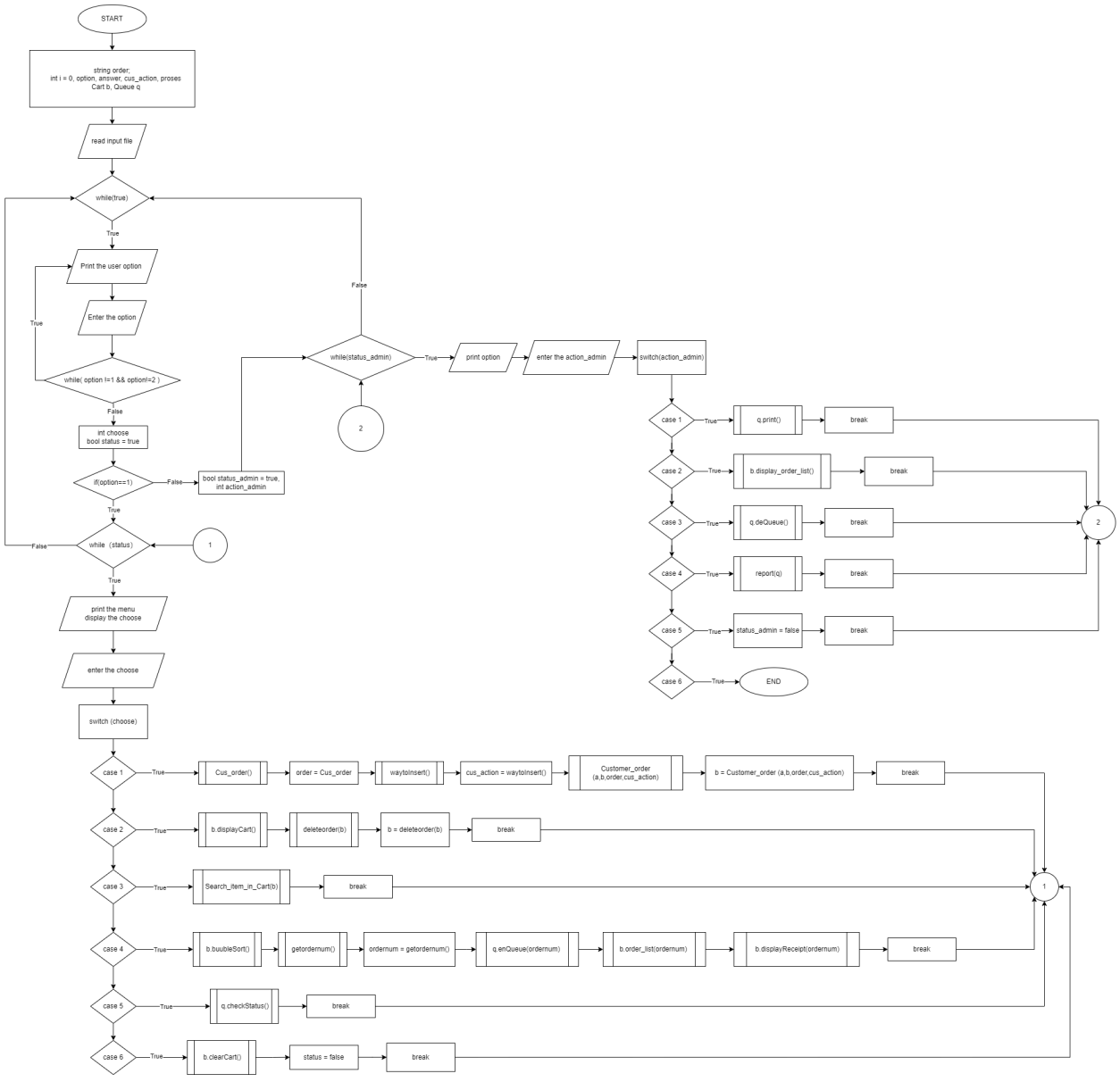
In our system, the queue is used to display the food order based on the order number. The queue is using first in first out which is very suitable for the ordering system. The ordering system is first come first serve which is the same as the queue. After ordering the food, the customers need to type in the order number provided at the restaurant. The system will enqueue the order number into the queue. The system will show the order number in which phase such as preparing phase or ready to serve. This is to help the customers to know the status of their order easily. The customers can sit and wait until their order number shows on the ready to serve phase. After the order has been delivered to the customers , the admin of the system can dequeue the order number from the queue and it will not affect the order number after it. The kitchen staff can focus on the last uncompleted orders in the list.

Class Diagram

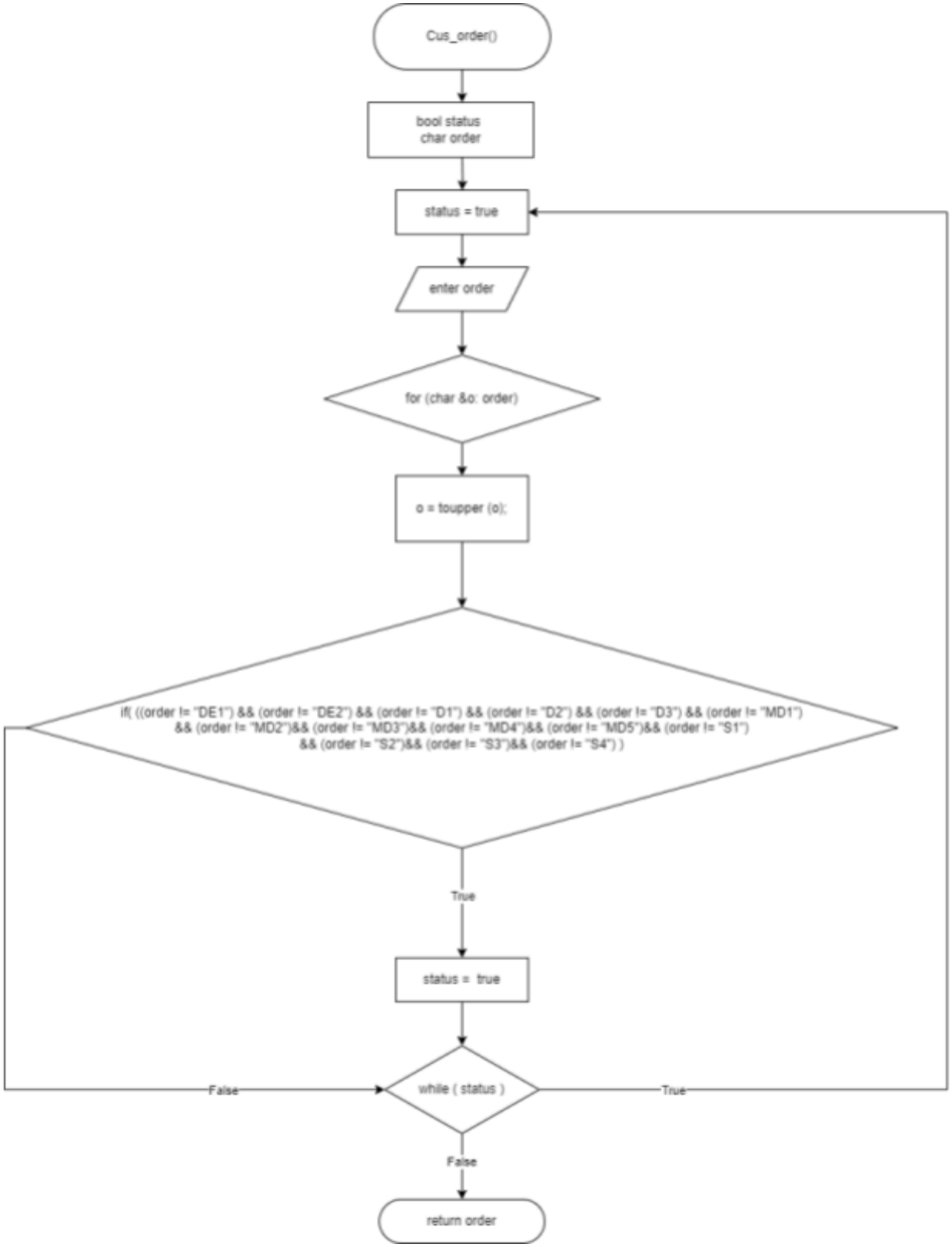


Flowchart

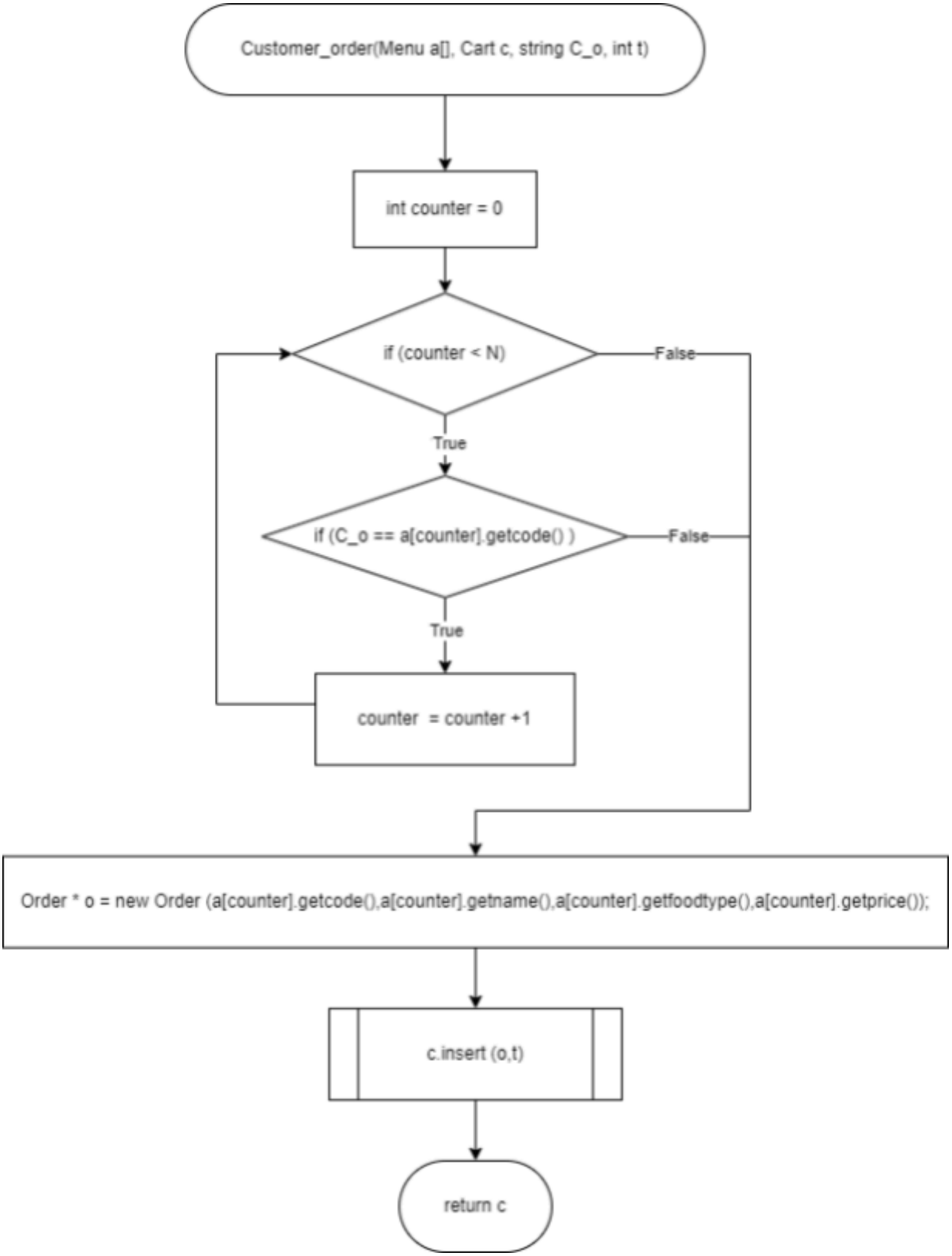
Main body



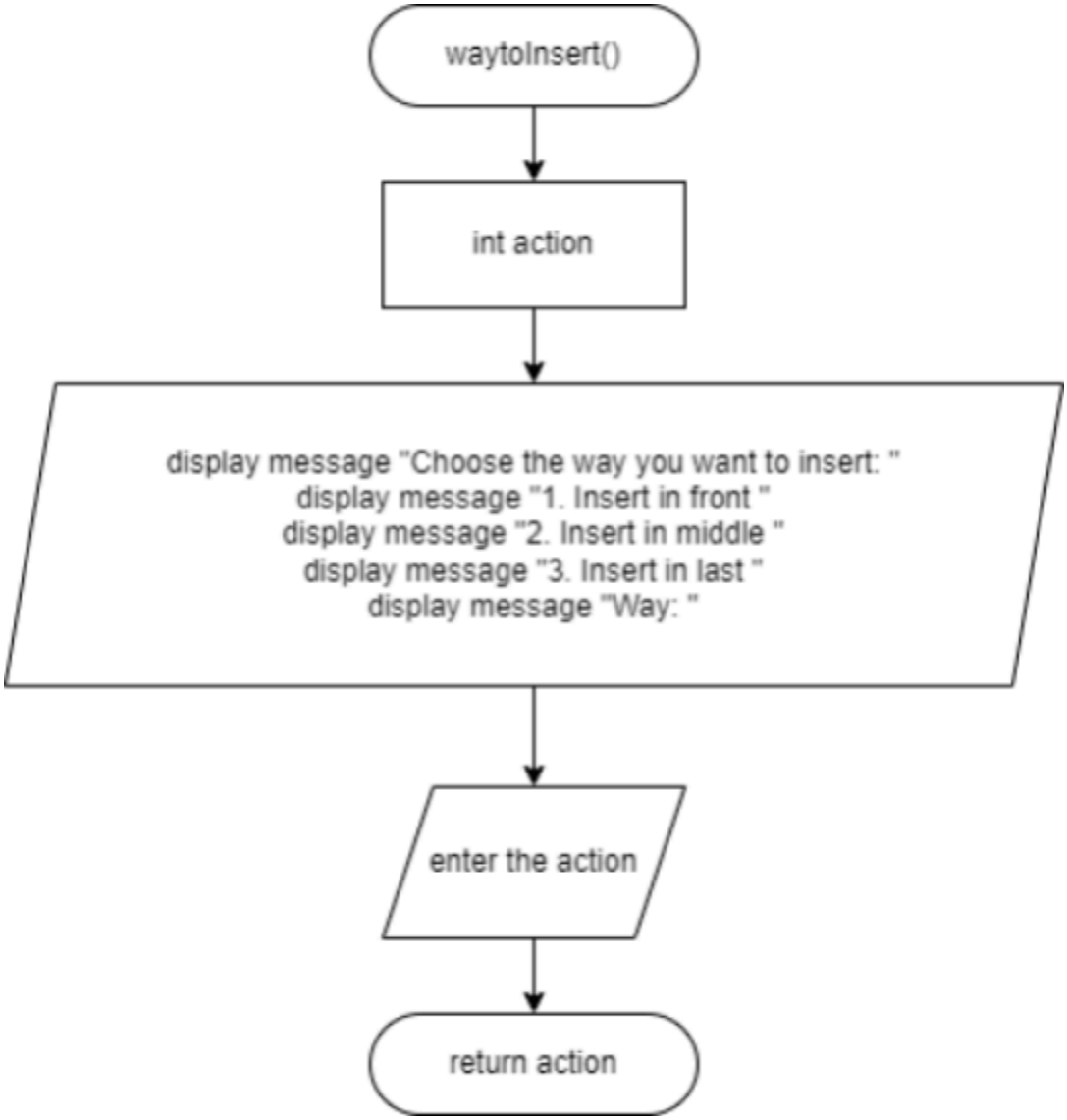
Cus_order()


Cus_order()
bool status
char order
status = true
enter order
for (char &o: order)
o = toupper (o);
if(((order != "DE1") && (order != "DE2") && (order != "D1") && (order != "D2") && (order != "D3") && (order != "MD1")
&& (order != "MD2")&& (order != "MD3")&& (order != "MD4")&& (order != "MD5")&& (order != "S1")
&& (order != "S2")&& (order != "S3")&& (order != "S4"))
True
status = true
while (status)
False
True
False
return order

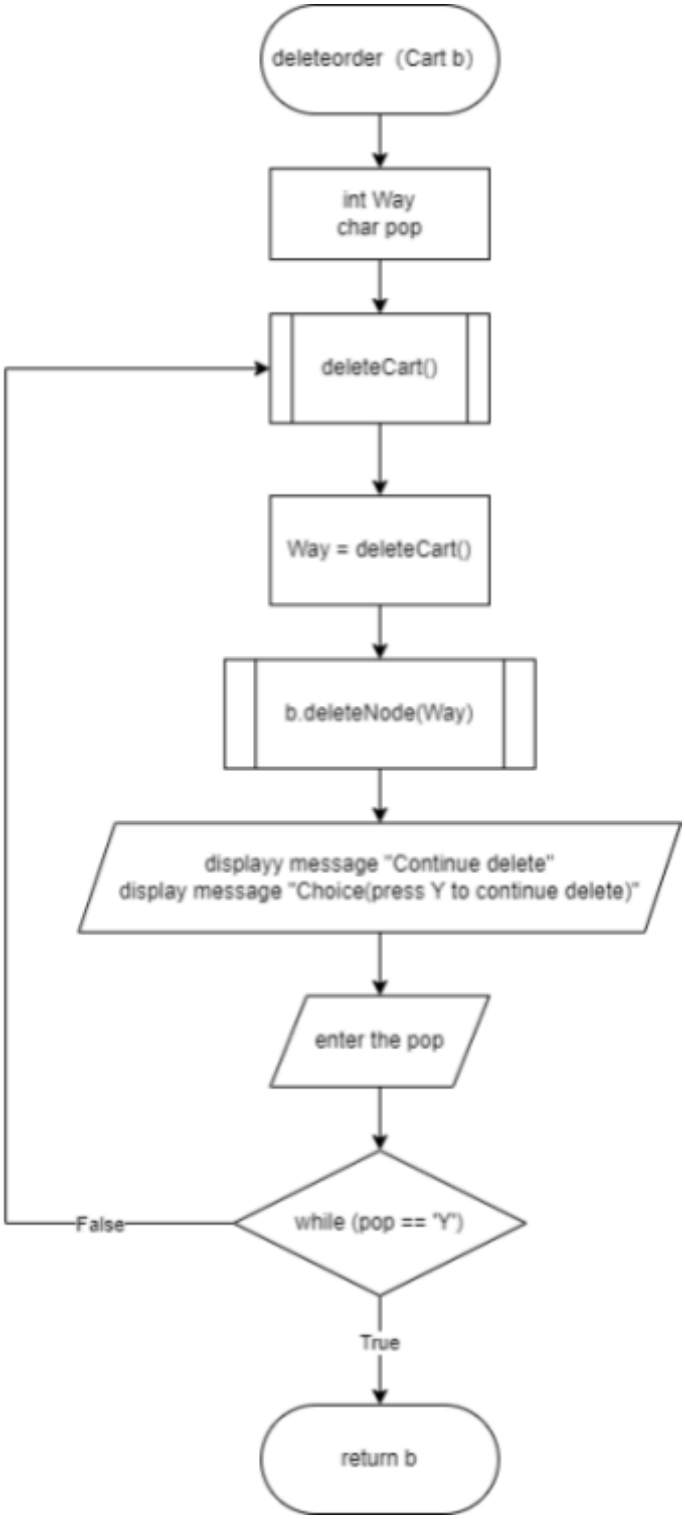
Customer_order



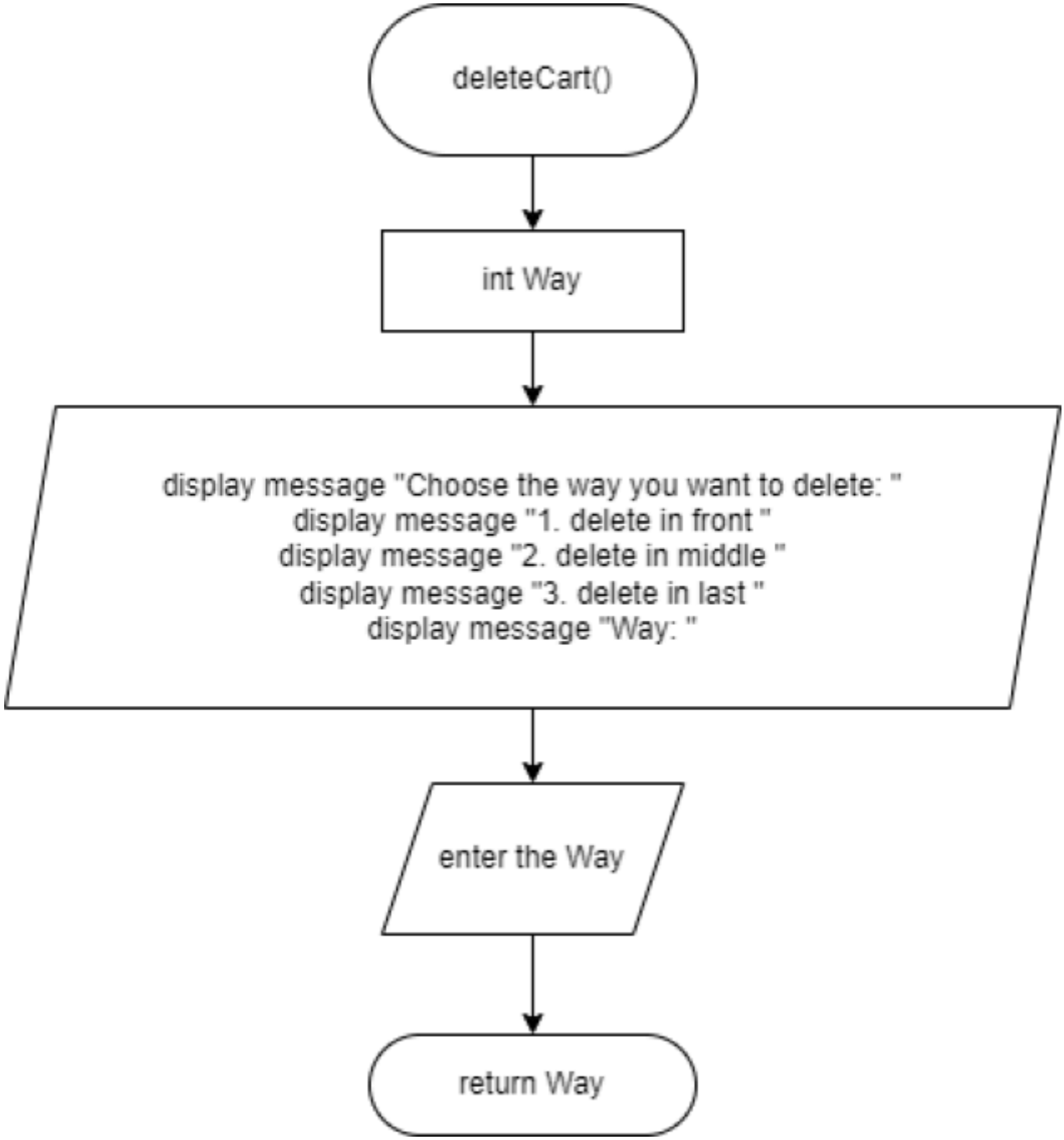
waytoInsert()



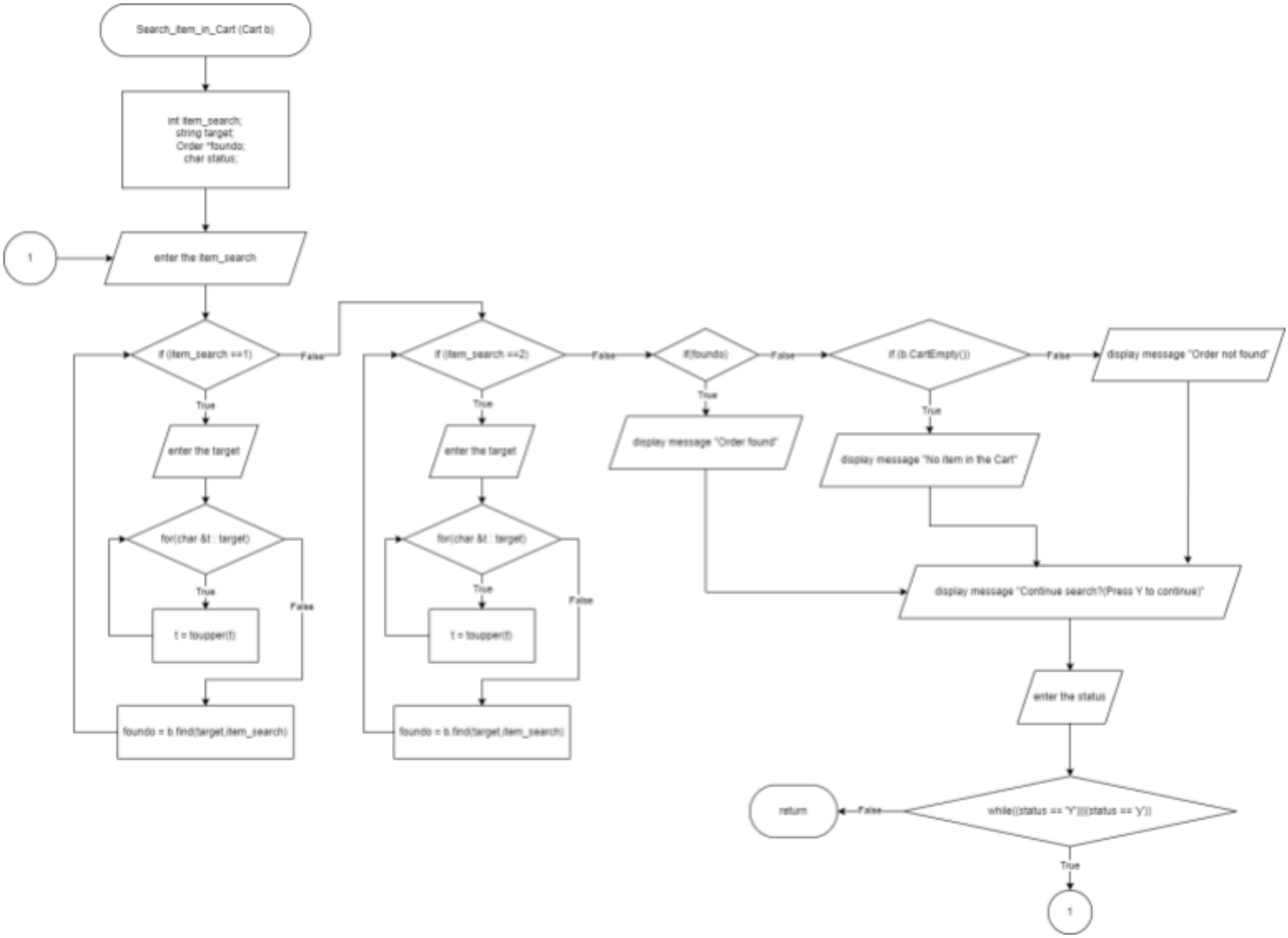
deleteorder()



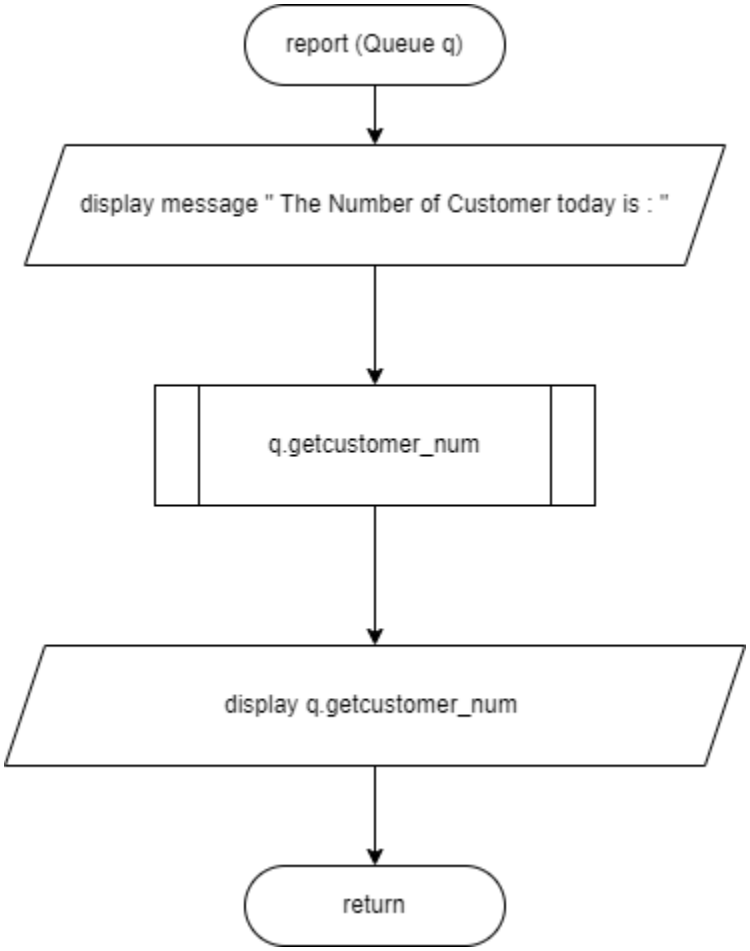
deleteCart()


deleteCart()
int Way
display message "Choose the way you want to delete: "
display message "1. delete in front "
display message "2. delete in middle "
display message "3. delete in last "
display message "Way: "
enter the Way
return Way

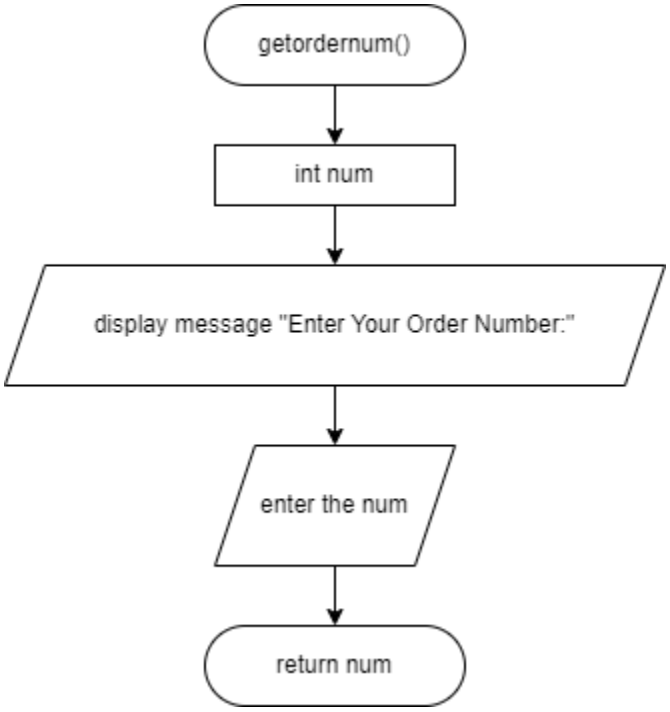
Search_item_in_Cart()



report()


report (Queue q)
display message " The Number of Customer today is : "
q.getcustomer_num
display q.getcustomer_num
return

getordernum()



User Guide

In the home page, the system will show two types of users: customers and the kitchen. For different users, the system has distinct options for them. The users need to select and continue their ordering.

For the customer option, the menu will display the food code, food name and prices for customers. Then the options will be displayed to them. This option is prepared for the customer only. These options' functions are shown below :

1) **Add Order:** The users can add the order to the cart. The system will ask the users to enter the food code. The users can add other orders if they want.

2) **Delete Order:** The users can delete the orders in the cart. The users can delete orders in front, middle or back based on the order arrangement.

3) **Find the item you ordered in Cart:** The users can search the food ordered inside the cart. The users can search for the food based on the food code or name. The system will give feedback that the food name or code is found or not.

4) **Confirm Order:** The users want to confirm their orders. The users need to enter the order number. The system will arrange the order based on the users' needs and display it to the users.

5) **Check the Order Status:** The user can check the order status to ensure that their order is being processed. The users need to enter their order number (from option 4). If the order is completed, it will be shown to the users.

6) **Back to the home page:** The user can go back to the home page after they complete their order.

For the kitchen option, it is for the staff of the restaurant to manage the order and it can only be viewed by the admin. The option will be displayed to them. These options' functions are shown below :

1) **Checking number of orders:** The admin can check the number of orders. The system will display the remaining uncompleted orders.

2) **Display the order list:** The users can enter the order number to look at the order details such as food name.

3) **The order is done:** This option will dequeue the first order number in the order list.

4) **Generate sales report (By Day):** The system will generate the report with the number of customers.

5) **Back to the home page:** The system will go back to the home page.

6) **The system is closed:** The system will be ended.

Source Code

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#define N 14
using namespace std;

class Menu{
	private:
		string code,name, foodtype;
		float price;
	public:
		Menu(string c = " ",string n = " ", string f = " ", float p =0.0 ){
			code = c;
			name = n;
			foodtype = f;
			price = p;
		}
		string getcode(){
			return code;
		}
		string getname(){
			return name;
		}
		string getfoodtype(){
			return foodtype;
		}
		float getprice(){
			return price;
		}
		void setcode(string c){
			code = c;
		}
		void setname(string n){
			name = n;
		}
		void settype(string f){
			foodtype = f;
		}
```

```
		void setprice(float p){
			price = p;
		}
		void printmenu(){
		cout << code << setw(20) << name << setw(20) << foodtype << setw(20) <<
price << endl;
		}
};

class Order{
	Menu menu;
	string code,name, foodtype;
	float price;
	int counter_num;

	public:
		Order *next;
		Order(string c ,string n, string f ,float p){
			code = c;
			name = n;
			foodtype = f;
			price = p;
			counter_num = 1;
			next = NULL;
		}
		string getcode(){
			return code;
		}
		string getname(){
			return name;
		}
		string getfoodtype(){
			return foodtype;
		}
		float getprice(){
			return price;
		}
		void setcode(string c){
			code = c;
		}
```

```
		void setname(string n){
			name = n;
		}
		void settype(string f){
			foodtype = f;
		}
		void setprice(float p){
			price = p;
		}
};

class Cart{
	Order *head;
	public:
		Cart(){
			head = NULL;
		}

		bool CartEmpty(){
			return head == NULL;
		}

		void insert(Order *newNode,int w){
			if(w == 1){
				if(CartEmpty()){
					head = newNode;
				}
				else{
					newNode ->next = head;
					head = newNode;
				}
			}//insert the number in front
			else if (w == 2){
				if(CartEmpty()){
					head = newNode;
				}
				else{
					int position;
					cout << "Enter the position that you want to key in:";
					cin >> position;
```

```
                                        Order *temp = head;
                                        for(int i =1; i<(position-1); i++){   // int count = 1;
while(temp->next!= NULL && count <loc){ temp = temp-> next; count ++
                                                temp = temp->next;
                                        }
                                        newNode->next = temp->next;
                                        temp->next = newNode;
                                }
                        }// insert the number at the middle which the user provide the location
                        else if (w == 3){
                                if(CartEmpty()){
                                        head = newNode;
                                }
                                else{
                                        Order *temp = head;
                                        while(temp->next != NULL){   // make it become null until
the last
                                                temp = temp->next;
                                        }
                                        temp-> next = newNode;
                                }
                        }// insert the number at the last
                }

                void deleteNode(int w){
                        Order *temp = head;
                        if(w == 1){
                                if(CartEmpty()){
                                        cout << "Does not have order can be delete" << endl;
                                }
                                else{
                                        if(temp->next != NULL){
                                        head = head->next;
                                        temp ->next = NULL;
                                        }
                                        delete temp;
                                }
                        }// delete the first number
                        else if (w == 2){
                                if(CartEmpty()){
```

```
                cout << "Does not have order can be delete" << endl;
            }
            else{
                int position;
                cout << "Enter the position that you want to key in:";
                cin >> position;
                Order *temp1;
                for(int i =1; i<position; i++){   // int count = 1;
while(temp->next!= NULL && count <loc){ temp = temp-> next; count ++
                    temp1 = temp;
                    temp = temp->next;
                }
                temp1->next = temp->next;
                temp->next = NULL;
                delete temp;
            }
        }// delete the number at the middle which the user provide the location
        else if(w == 3){
            if(CartEmpty()){
                cout << "Does not have order can be delete" << endl;
            }
            else{
                Order *temp1;
                while(temp->next != NULL){
                    temp1 = temp;
                    temp = temp -> next;
                }
                temp1->next = NULL;
                delete temp;
            }
        }// delete the last
    }
    void displayCart(){
        Order *temp;
        temp = head;
        if(temp == NULL){
            cout << "Cart is empty" << endl;
            return;
        }
```

```
                cout << "************************ CART
******************************" << endl << endl;
                cout << "-------------------------------------------------------------" << endl;
                cout << "Code" << setw(18) << "Name" << setw(20) << "Type" <<
setw(20) << "Price(RM)" << endl;
                cout << "-------------------------------------------------------------" << endl;
                while(temp){
                        cout << temp->getcode() << setw(20) << temp->getname() <<
setw(20) << temp->getfoodtype() << setw(20) << temp->getprice() << endl;
                        temp = temp -> next;
                }
                 cout << endl;
        }
        void displayReceipt(int order_num){
                fstream outfile("Receipt.txt",ios :: out);
                Order *temp;
                temp = head;
                float totalprice;
                if(temp == NULL){
                        outfile << "Cart is empty" << endl;
                        cout << "Cart is empty" << endl;
                }
                outfile << "Your Order Number is: " << order_num << endl;
                outfile << "********************** Receipt
******************************" << endl << endl;
                outfile << "-------------------------------------------------------------" << endl;
                outfile << "Code" << setw(18) << "Name" << setw(20) << "Type" <<
setw(20) << "Price(RM)" << endl;
                outfile << "-------------------------------------------------------------" << endl;
                cout << "********************** Receipt
******************************" << endl << endl;
                cout << "-------------------------------------------------------------" << endl;
                cout << "Code" << setw(18) << "Name" << setw(20) << "Type" <<
setw(20) << "Price(RM)" << endl;
                cout << "-------------------------------------------------------------" << endl;
                while(temp){
                        cout << temp->getcode() << setw(20) << temp->getname() <<
setw(20) << temp->getfoodtype() << setw(20) << temp->getprice() << endl;
                        outfile << temp->getcode() << setw(20) << temp->getname() <<
setw(20) << temp->getfoodtype() << setw(20) << temp->getprice() << endl;
```

```
                totalprice += temp->getprice();
                temp = temp -> next;
            }
            outfile << "The Total Price is " << setw(43) << totalprice << endl;
             cout << endl;
             outfile << endl;
        }
        Order* find(string target,int n){
            Order *current = head;
            while(current){
                if(n == 1){
                    if(current -> getcode() == target){
                        return current;
                    }
                }
                else if(n == 2){
                    if(current -> getname() == target){
                        return current;
                    }
                }
                current = current ->next;
            }
            return NULL;
        }
        void bubbleSort(){
int swapped,way;
Order *ptr1;
Order *lptr = NULL;

cout << "Select Receipt display type: " << endl;
   cout << "1. Arrange the Receipt by following the code (ascending order)" << endl;
   cout << "2. Arrange the Receipt by following the code (descending order)" << endl;
   cout << "3. Arrange the Receipt by following the name (ascending order)" << endl;
   cout << "4. Arrange the Receipt by following the name (descending order)" << endl;
   cout << "5. Arrange the Receipt by following the foodtype (ascending order)" << endl;
   cout << "6. Arrange the Receipt by following the foodtype (descending order)" << endl;
   cout << "Your choice:";
   cin >> way;
// Checking for empty list
if (head == NULL)
```

```
        return;
    if(way == 1){
    do {
        swapped = 0;
        ptr1 = head;

        while (ptr1->next != lptr) {
            if (ptr1->getcode() > ptr1->next->getcode()) {
                swap(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}
    else if(way == 2){
    do {
        swapped = 0;
        ptr1 = head;

        while (ptr1->next != lptr) {
            if (ptr1->getcode() < ptr1->next->getcode()) {
                swap(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}
    else if(way == 3){
    do {
        swapped = 0;
        ptr1 = head;

        while (ptr1->next != lptr) {
            if (ptr1->getname() > ptr1->next->getname()) {
                swap(ptr1, ptr1->next);
                swapped = 1;
```

```
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}
    else if(way == 4){
    do {
        swapped = 0;
        ptr1 = head;

        while (ptr1->next != lptr) {
            if (ptr1->getname() < ptr1->next->getname()) {
                swap(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}
    else if(way == 5){
    do {
        swapped = 0;
        ptr1 = head;

        while (ptr1->next != lptr) {
            if (ptr1->getfoodtype() > ptr1->next->getfoodtype()) {
                swap(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}
        if(way == 6){
    do {
        swapped = 0;
        ptr1 = head;
```

```
        while (ptr1->next != lptr) {
            if (ptr1->getfoodtype() < ptr1->next->getfoodtype()) {
                swap(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);
}
}

// Helper function to swap two nodes in the linked list
void swap(Order *a, Order *b) {
    string temp_code = a->getcode();
    string temp_name = a->getname();
    string temp_foodtype = a->getfoodtype();
    float temp_price = a->getprice();

    a->setcode(b->getcode());
    a->setname(b->getname());
    a->settype(b->getfoodtype());
    a->setprice(b->getprice());

    b->setcode(temp_code);
    b->setname(temp_name);
    b->settype(temp_foodtype);
    b->setprice(temp_price);
}

void clearCart(){
        while(head != NULL){
                Order * temp = head;
                head = head->next;
                delete temp;
                }
        }

        void order_list(int ordernum){
```

```
        Order *temp;
        temp = head;
        string number_as_string = to_string(ordernum);
        ofstream outputfile("Order_"+number_as_string+".txt");
        if(outputfile.is_open()){
                //cout << "File can open " << endl;
                while(temp){
                        outputfile << temp->getname() << endl;
                        temp = temp->next;
                        //cout << "file can output" << endl;
                }
        }
        else{
                cout << "file cannot open" << endl;
        }
}

        void display_order_list(){
        string name,number;
        cout << "Enter the Order Number :";
        cin >> number;
        ifstream inputfile("Order_"+number+".txt");
        cout << "The Order list :" << endl;
        if(inputfile.is_open()){
                while(getline(inputfile,name)){
                        cout << name << endl;
                }
        }
        else{
                cout << "file cannot open" << endl;
        }
}
};

class Retaurant{
        int front;
        int back;
        int count;
        int customer_num;
        int items[N];
```

```
public:
    Retaurant(){
        front = 0;
        count = 0;
        back = N-1;
        customer_num = 0;
    }
    bool isEmpty(){
        return (count == 0);
    }
    bool isFull(){
        return (count == N);
    }
    void enQueue(int d){
        if(isFull()){
            cout << "Sorry, the queue is full" << endl;
        }
        else{
           back = (back + 1) % N;
           items[back] = d; // items[++ back] = d
           count ++;
           customer_num++;
      }
    }
    void deQueue(){
        if(isEmpty()){
            cout << "Sorry, no item in the queue" << endl;
        }
        else{
            cout << "The customer Order "<< items[front] <<" has been done"
<< endl;
            front = (front + 1) % N;
            count --;
        }
    }
    int getFront(){
        return items[front];
    }
    int getBack(){
        return items[back];
```

```
		}
		int getcustomer_num(){
			return customer_num;
		}
		void print(){
			if(isEmpty()){
				cout << "No order is placed" << endl;
			}
			else{
			if(front > back){
				for(int i = front; i<N; i++){
					cout << "The Number Order " << items[i] << " is placed"
<< endl;
				}
				for(int i = 0; i <= back; i++){
					cout << "The Number Order " << items[i] << " is placed"
<< endl;
			   }
			}
			else{
				for(int i = front; i <= back; i++){
					cout << "The Number Order " << items[i] << " is placed"
<< endl;
			   }
			}
		}
		cout << endl;
	}
	void checkStatus() {
	int expectedNumber;
	cout << "Enter Your Order Number: ";
	cin >> expectedNumber;
  bool orderFound = false;

  for (int i = front; i <= back; i++) {
    int currentNumber = items[i];
    if (currentNumber == expectedNumber) {
      orderFound = true;
      cout << "Order Number " << currentNumber << " is in progress." << endl;
      break;  // Exit the loop once the order is found
```

```
        }
    }

    if (!orderFound) {
        cout << "Order Number " << expectedNumber << " completed." << endl;
    }
}

};

string Cus_order(){
        string order;
        do{
                cout << "What order do you want to place?(based on the code given)" << endl;
                cout << "Order:";
                cin >> order;
                for(char &o : order){
                        o = toupper(o);
                }
                if((order != "DE1") && (order != "DE2") && (order != "D1") && (order !=
"D2") && (order != "D3") && (order != "MD1") && (order != "MD2")&& (order !=
"MD3")&& (order != "MD4")&& (order != "MD5")&& (order != "S1")&& (order != "S2")&&
(order != "S3")&& (order != "S4")){
                        cout << "Invalid code for order, Please order again" << endl;
                }
        }while((order != "DE1") && (order != "DE2") && (order != "D1") && (order != "D2")
&& (order != "D3") && (order != "MD1") && (order != "MD2")&& (order != "MD3")&&
(order != "MD4")&& (order != "MD5")&& (order != "S1")&& (order != "S2")&& (order !=
"S3")&& (order != "S4"));
        return order;
}

int waytoInsert(){
        int action;
        cout << "Choose the way you want to insert: " << endl;
        cout << "1. Insert in front" << endl;
        cout << "2. Insert in middle" << endl;
        cout << "3. Insert in last" << endl;
        cout << "Way: ";
        cin >> action;
```

```
	return action;
}

Cart Customer_order(Menu a[],Cart c,string C_o, int t){   // C_o = customer order
		int counter;
		for(counter = 0;  counter < N; counter++){
			if(C_o == a[counter].getcode()){
				break;
			}
		}
		Order * o = new Order
(a[counter].getcode(),a[counter].getname(),a[counter].getfoodtype(),a[counter].getprice());
		c.insert(o,t);
		return c;
}

int deleteCart(){
	int Way;
	cout << "Choose the way you want to delete: " << endl;
	cout << "1. Delete in front" << endl;
	cout << "2. Delete in middle" << endl;
	cout << "3. Delete in last" << endl;
	cout << "Way: ";
	cin >> Way;
	return Way;
}

Cart deleteorder(Cart b){
	int Way;
	char pop;
		do{
			Way = deleteCart();
			b.deleteNode(Way);
			cout << "Continue delete?" << endl;
			cout << "Choice(press Y to continue delete):";
			cin >> pop;
		}while(pop == 'Y');
		return b;
}
```

```
void Search_item_in_Cart(Cart b){
	int item_search;
	string target;
	Order *foundo;
	char status;
	do{
		cout << "Seacrh the Cart by using:" << endl;
		cout << "1. Code" << endl;
		cout << "2. Name" << endl;
		cout << "Choice: ";
		cin >> item_search;
		if(item_search == 1){
			cout << "Enter the code:";
			cin >> target;
			for(char &t : target){
			t = toupper(t);
			}
			foundo = b.find(target,item_search);
		}
		else if(item_search == 2){
			cout << "Enter the name: ";
			cin >> target;
			for(char &t : target){
			t = toupper(t);
			}
			foundo = b.find(target,item_search);
		}
		if(foundo){
			cout << "Order found" << endl;
		}
		else{
			if(b.CartEmpty()){
				cout << "No item in the Cart" << endl;
			}
			else{
			cout << "Order not found" << endl;
			}
		}
		cout << "Continue search?(Press Y to continue)";
		cin >> status;
```

```
	}while((status == 'Y')||(status == 'y'));
}

int getordernum(){
	int num;
	cout << "Enter Your Order Number: ";
	cin >> num;
	return num;
}

void report(Retaurant q){
	cout << "The Number of Customer today is : " << q.getcustomer_num() << endl;
}

int main(){
   string menu_code, menu_name, menu_type,order;
	float menu_price;
	Menu a [N];
	int i = 0,cus_action,option,ordernum;
	Cart b;
	Retaurant q;
	ifstream file("input.txt.txt");
	/*
	if(!file){
		cout << " Error opening file" << endl;
	}
	else{
		cout << "File can run" << endl; // use for testing the file
	}
	*/
	while(getline(file,menu_code,',')){
		getline(file,menu_name,',');
		getline(file,menu_type,',');
		file >> menu_price;
		file.ignore();
		a[i].setcode(menu_code);
		a[i].setname(menu_name);
		a[i].settype(menu_type);
		a[i].setprice(menu_price);
		i++;
```

```
	}
	while(true){
	do{
		system("CLS");
		cout << "Login your character " << endl;
		cout << "1. Customer" << endl;
		cout << "2. Kitchen" << endl;
		cout << "Option: ";
		cin >> option;
	}while(option != 1 && option != 2);
	int choose;
	bool status = true;
	if(option == 1){
	while(status){
		system("CLS");
		cout << "************ Welcome to BOBOBOY Restaurant's Ordering System
************" << endl << endl;
		cout << "-----------------------------------------------------------" << endl;
		cout << "Code" << setw(18) << "Name" << setw(20) << "Type" << setw(20) <<
"Price(RM)" << endl;
		cout << "-----------------------------------------------------------" << endl;
		for(int i = 0; i <N; i++){
			a[i].printmenu();
		}
		cout << "Process:" << endl;
		cout << "1. Add Order" << endl;
		cout <<"2. Delete Order" << endl;
		cout << "3. Find the item you ordered in Cart" << endl;
		cout << "4. Confirm Order" << endl;
		cout << "5. Check the Order Status" << endl;
		cout << "6. Back to the home page" << endl;
		cout << "Your Choice:";
		cin >> choose;
		switch(choose){
			case 1 :	order = Cus_order();
						cus_action = waytoInsert();
						b = Customer_order(a,b,order,cus_action);
						break;
			case 2 :	b.displayCart();
						b = deleteorder(b);
```

```
                                        break;
                case 3 :        Search_item_in_Cart(b);
                                        break;
                case 4 :        b.bubbleSort();
                                        ordernum = getordernum();
                                        q.enQueue(ordernum);
                                        b.order_list(ordernum);
                                        b.displayReceipt(ordernum);
                                        system("pause");
                                        break;
                case 5 :        q.checkStatus(); system("pause"); break;
                case 6 :        b.clearCart(); status = false; break;
        }
    }
    }
    else{
        bool status_admin = true;
        int action_admin;
        while(status_admin){
            system("CLS");
            cout << "Process: " << endl;
            cout << "1. Checking number of order" << endl;
            cout << "2. Display the order list" << endl;
            cout << "3. The order is done " << endl;
            cout << "4. Generate sales report(By Day) " << endl;
            cout << "5. Back to the home page " << endl;
            cout << "6. The system is closed" << endl;
            cout << "Your choice: ";
            cin >> action_admin;
            switch(action_admin){
                case 1 : q.print(); system("pause"); break;
                case 2 : b.display_order_list(); system("pause"); break;
                case 3 : q.deQueue(); system("pause"); break;
                case 4 : report(q); system("pause"); break;
                case 5 : status_admin = false; break;
                case 6 : cout << "System closed" << endl; return 0;
            }
        }
    }
}}
```