UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING
Faculty of Engineering

Assignment 2 (Group)

SECJ2013 DATA STRUCTURE AND ALGORITHM

SEMESTER I, SESSION 2020/2021

Topic: Food Ordering System

Lecturer: Dr. Johanna binti Ahmad

Name	Matric No.
KONG HAO YANG	A19EC0065
LOO ZHI XUEN	A19EC0078
SEE WEN XIANG	A19EC0206

Section: 06

Programme: Bachelor of Computer Science (Software Engineering)

Table of Contents

1.0 Objective 1

2.0 Synopsis 1

3.0 Appendix 3

`Menu.txt` 3

`Item.h` 4

`Item.cpp` 4

`Order.h` 5

`Order.cpp` 6

`Customer.h` 8

`Customer.cpp` 9

`Payment.h` 14

`Payment.cpp` 15

`Customerlinkedlist.h` 16

`Customerlinkedlist.cpp` 17

`Itemlinkedlist.h` 20

`Itemlinkedlist.cpp` 21

`Menulinkedlist.h` 23

`Menulinkedlist.cpp` 24

`Node.h` 28

`Node.cpp` 28

`main.cpp` 29

1.0 Objective

The main objectives of developing the Food Ordering System is:

- To automate food ordering process in a restaurant
- To apply linked list technique in the system

2.0 Synopsis

Introduction

Food Ordering System is a software application which helps a restaurant to control and optimize over their order. This system is mainly designed for customer by making them to be more convenient in ordering food in a restaurant. For the customer's point of view, the customers are able to view menu, order food, view their order and make payment on their own in an easier manner. For the staff's point of view, staff can view all the payment list after the customers have done their payment. Next, staff able to make changes on the menu such as insert the new food item into the menu, delete food item from the menu and update the latest price of specific food item. Staff also could cancel the customers' order upon the request from customer.

This system helps the restaurant to reduce the human resources required because once customers step into restaurant, the whole food ordering process from viewing menu until making payment all is fully automated which can be done by customers themselves. Besides that, this system is able to do all the functions of different roles such as waiters and cashier involved in the restaurant more accurately and in a faster way. The program enables customers to make payment by searching their table number or their name.

In the program, we are implementing linked list technique which the technique will advance the modification of the data management in specific area. In our system, linked list is implemented in menu and ordering part. In menu, staff able to insert, delete and update the food item. Then, in ordering part, customer able to insert and delete their item from the order. At the same time, it brings convenience to both staff and customers as staff able to track and cancel the customers' order easily once the customer wanted to modify their order even though they have submitted their order previously

In summary, the main goal of this Food Ordering System is to maintain the restaurant's functions in a more accurate manner and effective and reduce the use of manual entries.

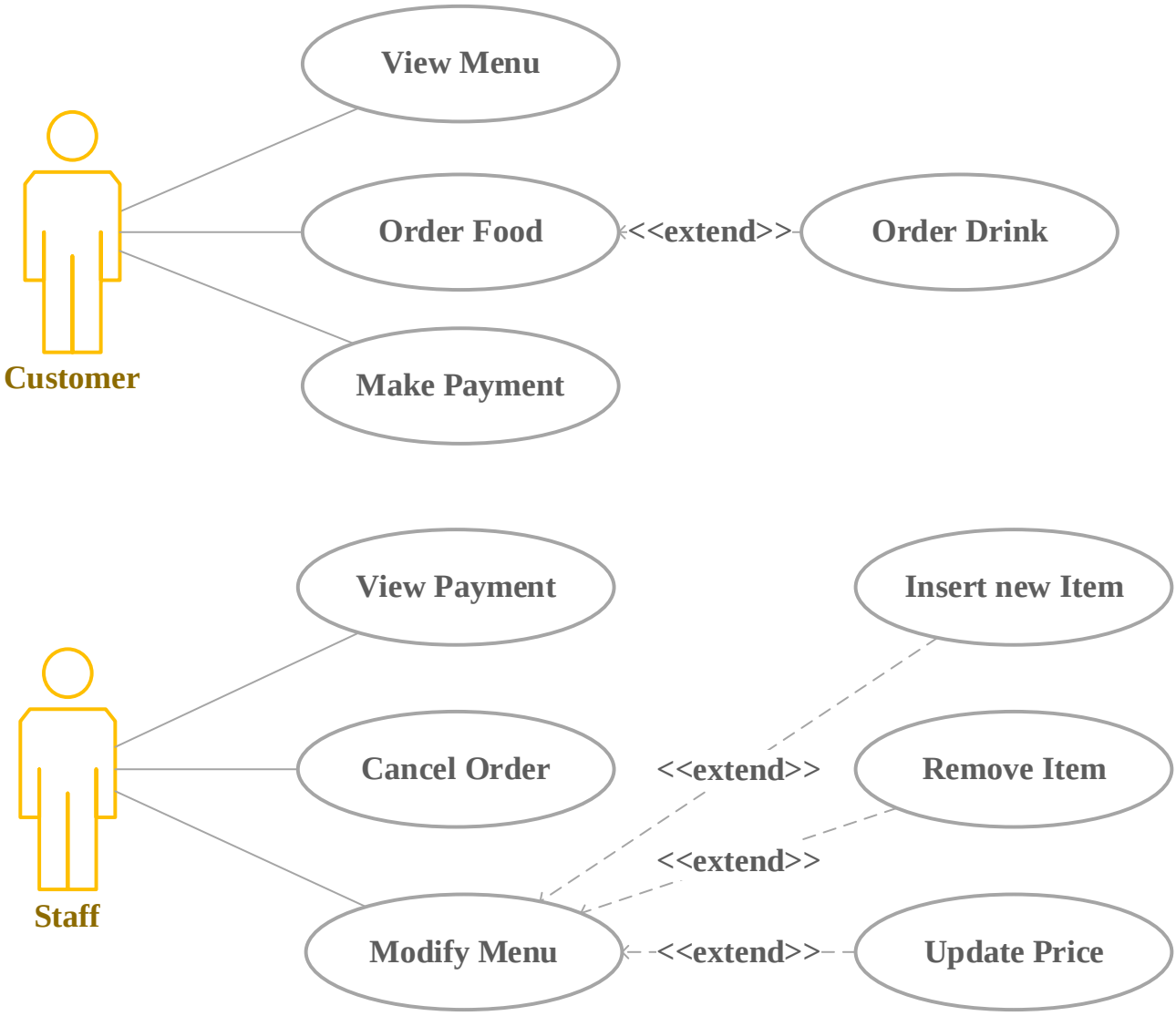

Figure 2.1: Use Case Diagram

System Functionality

Customer

- View the menu before making an order.
- Insert name, table number and item code to place the order.
- Add item to the order and remove item from the order.
- Search order based on their name or table number in order to make payment.
- Check the status of the payment whether the order is paid.

Staff

- View the list of all customer's payments
- Insert, delete the food item and update price of food item inside the menu
- Cancel the customers' order upon request from customer.

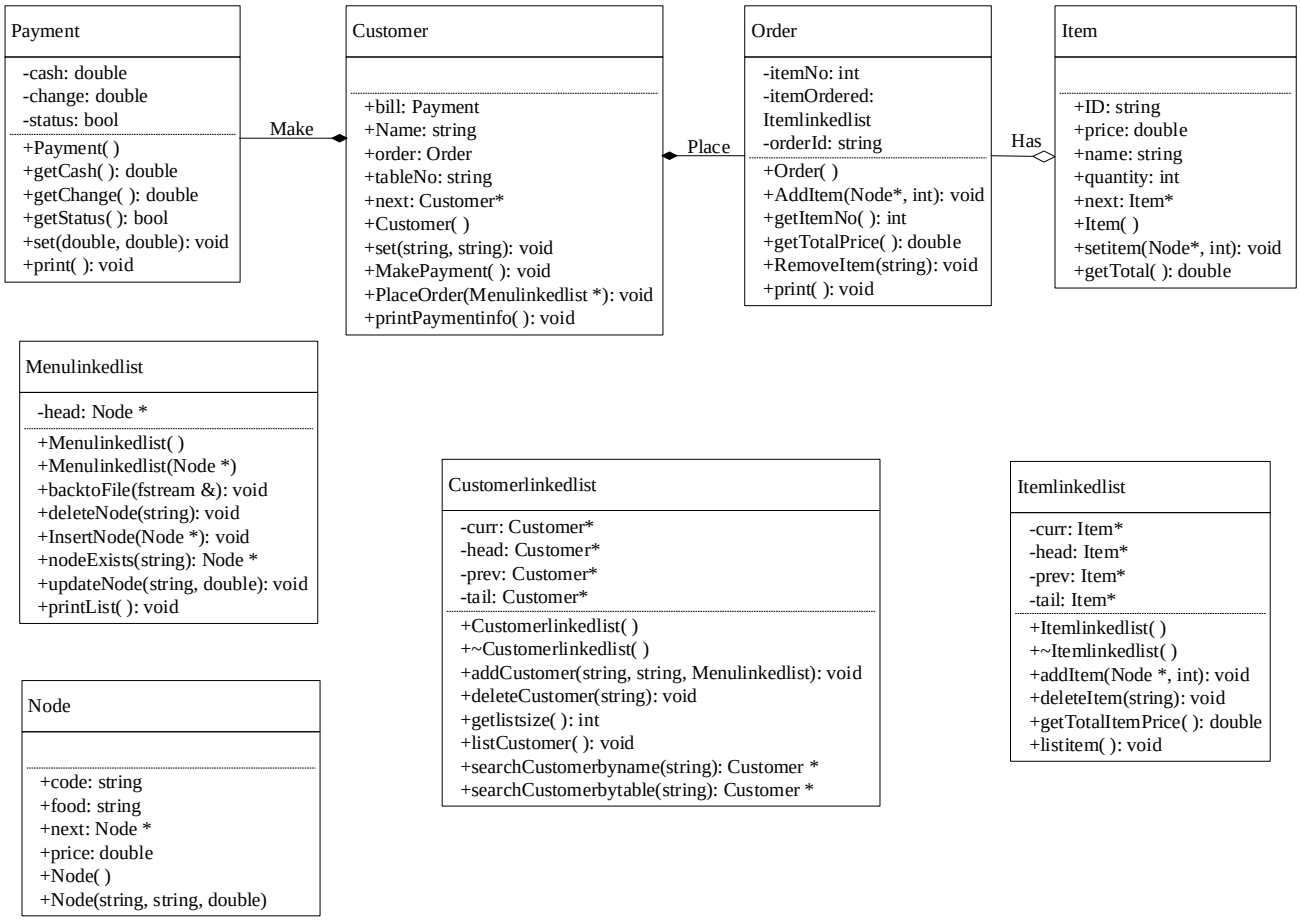

Figure 2.2: UML Diagram of Food Ordering System

3.0 Appendix

Menu.txt

```
N01 NasiLemak 2.50
C01 ChickenRice 3.50
R01 RotiCanai 1.20
F01 FriedRice 3.50
L01 Laksa 4.50
M02 Milo 2.80
C02 Coffee 2.20
S01 SoyBean 1.70
H01 HerbalTea 1.60
M01 MilkTea 8.80
F04 Maggie 4.80
```

Item.h

```
#ifndef _ITEM_H_
#define _ITEM_H_
#include <string>
#include "Node.h"
using namespace std;
class Item{
      public:
            string ID;
            string name;
            double price;
            int quantity;
            Item *next;
            Item ();
            void setitem (Node *,int);
            double getTotal();
};
#endif
```

Item.cpp

```
#include "Item.h"
#include <iostream>
#include "Node.h"
using namespace std;
Item::Item()
{
      ID="";
      price=0;
      name="";
      quantity=0;
      next=NULL;
}
void Item::setitem(Node *food,int item_quantity )
{
      ID=food->code;
      name=food->food;
      price=food->price;
      quantity=item_quantity;
}
double Item::getTotal(){
      return price * quantity;
}
```

Order.h

```
#ifndef ORDER_H
#define ORDER_H
#include "Itemlinkedlist.h"
#include "Node.h"
using namespace std;
class Order
{
      private:
            string orderId;
            int itemNo;
            Itemlinkedlist *itemOrdered;
      public:
            Order ();
            void setID (string);
            string getOrderID();
            int getItemNo ();
            void AddItem (Node *,int);
            void RemoveItem (string);
            void print ();
            double getTotalPrice ();

};
#endif
```

Order.cpp

```
#include "Order.h"
#include "Itemlinkedlist.h"
#include "Node.h"
#include <iostream>
#include <iomanip>
#include <stdlib.h>
using namespace std;

Order::Order()
{
	orderId="";
	itemNo=0;
	itemOrdered=NULL;

}
void Order::setID(string id)
{
	orderId=id;
}
string Order::getOrderID()
{
	return orderId;
}
void Order::AddItem (Node * food, int quantity)
{
	if (itemOrdered==NULL)
	{
		itemOrdered=new Itemlinkedlist ();
		itemOrdered->addItem(food,quantity);
	}
	else
	{
		itemOrdered->addItem(food,quantity);
	}
	itemNo++;
}

void Order::RemoveItem(string id)
{

	if (itemNo==0)
	{
		cout<<"Invalid action !!!"<<endl;
```

```
            cout<<"There is no item added in this order."<<endl;
      }
      else
      {
            itemOrdered->deleteItem(id);
            cout<<"The item has been removed successfully from the
            list."<<endl;
      }
}

int Order::getItemNo()
{
      return itemNo;
}

void Order::print ()
{
      cout<<endl<<endl<<"Order ID: "<<orderId<<endl;
      cout<<"Order List"<<endl;
      cout<<"----------"<<endl;
      cout<<left<<setw(10)<<"No"<<setw(20)<<"ID"<<setw(40)<<"Name"<<setw(2
      0)<<"Unit  Price/RM"  <<  setw(20)<<  "Quantity"<<  setw(20)<<"Total
      Price/RM"<<endl;
      itemOrdered->listitem();
      cout<<endl<<"Subtotal:RM"<<fixed<<setprecision(2)<<
      itemOrdered->getTotalItemPrice()<<endl<<endl;
}

double Order::getTotalPrice(){
      return itemOrdered->getTotalItemPrice();
}
```

Customer.h

```
#ifndef CUSTOMER_H
#define CUSTOMER_H
#include "Payment.h"
#include "Order.h"
#include "Item.h"
#include "Menulinkedlist.h"
#include <iostream>
using namespace std;

class Customer
{
      public:
            string tableNo;
            string Name;
            Order order;
            Payment bill;
            Customer *next;
            Customer ();
            void printPaymentinfo ();
            void set (string, string);
            void PlaceOrder (Menulinkedlist *);
            void MakePayment ();
};
#endif
```

Customer.cpp

```
#include "Customer.h"
#include "Payment.h"
#include "Order.h"
#include "Item.h"
#include "Menulinkedlist.h"
#include "Node.h"
#include <iostream>
#include <stdlib.h>
#include <iomanip>
using namespace std;

Customer::Customer()
{
	tableNo="";
	Name="";
	next=NULL;
}

void Customer::printPaymentinfo ()
{
	cout<<left<<setw(20)<<tableNo<<setw(40)<<Name<<setw(20)<<order.getTo
talPrice()<<setw(20)<<fixed<<setprecision(2)<<bill.getCash()<<setw(20)<<f
ixed<<setprecision(2)<<bill.getChange();
	if (bill.getStatus())
	{
		cout<<left<<setw(10)<<"Paid"<<endl;
	}
	else
	{
		cout<<left<<setw(10)<<"Unpay"<<endl;
	}
}
void Customer::set(string tn, string n)
{
	tableNo=tn;
	Name=n;
}

void Customer::PlaceOrder ( Menulinkedlist *menu)
{

	int itemquantity;
	string itemid,itemname;
```

```
	double itemprice;
	string tempid;
	int operation;

	do
	{
		cout<<"Choose the below operations"<<endl;
		cout<<"1. Add item into your order \n2. Remove item from your
order \n3. Return to customer menu"<<endl;
		cout<<"Choice: ";
		cin>>operation;
		system("CLS");
		if (operation==1)
		{

		cout<<"Please place your order based on the menu"<<endl;
		menu->printList();
		bool loop=true;
		int index=-1;
			tempid="OR"+tableNo+"/"+Name;

			cout<<endl<<"Your order ID: "<<tempid<<endl;
			order.setID(tempid);
		do
		{

			int pilihan;
			cin.ignore();
		Q:
			cout<<"Item ID: ";
			getline (cin,itemid);
			cout<<"Quantity: ";
			cin>>itemquantity;
			cin.ignore();
			Node *tempnode;
			tempnode=menu->nodeExists(itemid);

			if (tempnode==NULL)
			{
				cout<<"Invalid Item ID...Please enter again"<<endl;
				goto Q;
			}
```

```
                order.AddItem(tempnode,itemquantity);
                E:
                      cout<<"Do you wish to add another item into your order
list? \n 1. Yes \n 2. No"<<endl;
                      cout<<"Choice: ";
                      cin>>pilihan;
                      if (pilihan==1)
                      {
                            loop=true;

                      }
                      else if (pilihan==2)
                      {
                            loop=false;
                      }
                      else
                      {
                           cout<<"Invalid option...Please try again"<<endl;
                            goto E;
                      }

                }
                while (loop);
                system("CLS");
                order.print();

           }
           else if (operation==2)
           {
                if(order.getItemNo()==0)
                {
                      cout<<"There is no item available to be removed from
the order."<<endl;
           }
           else
           {


                cout<<"Please choose the item that you wish to remove from
your order"<<endl;
                order.print();
                bool loop2=true;
                int index2;
                int pilihan;
```

```
                do
                {
                      cin.ignore();
                      cout<<"Item ID: ";
                      getline (cin,itemid);

                      order.RemoveItem(itemid);
                H:
                      cout<<"Do you wish to remove another item into your
order list? \n 1. Yes \n 2. No"<<endl;
                      cout<<"Choice: ";
                      cin>>pilihan;
                      if (pilihan==1)
                      {
                            loop2=true;

                      }
                      else if (pilihan==2)
                      {
                            loop2=false;
                      }
                      else
                      {
                           cout<<"Invalid option...Please try again"<<endl;
                            goto H;
                      }

                }
                while (loop2);
                order.print();
          }
}
          else if (operation==3)
          {
                cout<<"The screen is returning to customer menu"<<endl;
                system("CLS");
          }
                }
                while (operation==1||operation==2);
}

void Customer::MakePayment ()
{
     if (!bill.getStatus()==false)// Checking whether the order has been
```

```
made payment or not
	{
		cout<<endl<<"You have made payment for this order."<<endl;
	}
	else
	{
	double total=0,change,cash;
	cout<<endl<<"Order ID: "<<order.getOrderID()<<endl;
	cout<<"Total amount payable: RM "<<order.getTotalPrice()<<endl;
	cout<<"This machine can only receive \n1. RM 1\n2. RM 5\n3. RM 10\n4.
RM 50\n5. RM 100 \nnotes"<<endl<<endl;

		do
		{

		cout<<"Note inserted: RM";
		cin>>cash;
		if (cash==1||cash==5||cash==10||cash==50||cash==100)
		{
			total=total+cash;
			cout<<"\nTotal amount inserted : RM"<<total<<endl;
		}
		else
		{
			cout<<"Invalid note inserted"<<endl;
		}

		}
		while
((cash!=1||cash!=5||cash!=10||cash!=50||cash!=100)&&(total<order.getTotal
Price()));
	change=total-order.getTotalPrice();
	bill.set(total, change);
	order.print();
	bill.print();
}
}
```

Payment.h

```
#ifndef PAYMENT_H
#define PAYMENT_H
using namespace std;
class Payment
{
      private :
            double cash;
            double change;
            bool status;
      public:
            Payment ();
            void set (double,double);
            bool getStatus();
            double getCash();
            double getChange ();
            void print ();
};
#endif
```

Payment.cpp

```
#include "Payment.h"
#include <iostream>
#include <iomanip>
using namespace std;

Payment::Payment()
{
	cash=0;
	change=0;
	status=false;
}

void Payment::print()
{
	cout<<"Cash: "<<"RM "<<fixed<<setprecision(2)<<cash<<endl;
	cout<<"Change: "<<"RM "<<fixed<<setprecision(2)<<change<<endl;
}
bool Payment::getStatus ()
{
	return status;
}
double Payment::getCash()
{
	return cash;
}
double Payment::getChange()
{
	return change;
}
void Payment::set(double cash,double change)
{
	this->cash=cash;
	this->change=change;
	status=true;
}
```

Customerlinkedlist.h

```
#ifndef CUSTOMERLINKEDLIST_H
#define CUSTOMERLINKEDLIST_H
#include <iostream>
#include "Customer.h"
#include "Menulinkedlist.h"
using namespace std;

class Customerlinkedlist
{
      private:
            Customer *head;
            Customer *prev;
            Customer *curr;
            Customer *tail;

      public:
            Customerlinkedlist ();
            ~Customerlinkedlist ();
            void addCustomer (string,string,Menulinkedlist *);
            void deleteCustomer (string);
            Customer * searchCustomerbytable (string);
            Customer * searchCustomerbyname (string);
            void listCustomer ();
            int getlistsize ();
};
#endif
```

Customerlinkedlist.cpp

```
#include "Customerlinkedlist.h"
#include "Menulinkedlist.h"
#include "Customer.h"
#include <iostream>
#include <iomanip>
using namespace std;

Customerlinkedlist::Customerlinkedlist()
{
      head=NULL;
      prev=NULL;
      curr=NULL;
      tail=NULL;
}
Customerlinkedlist::~Customerlinkedlist()
{
           while(head!=NULL){
                      curr=head;
                      head=head->next;
                      delete head;
                 }
}
void         Customerlinkedlist::addCustomer(string         tableno,string
name,Menulinkedlist *menu)
{
         Customer *newcustomer = new Customer;
               newcustomer->set(tableno,name);//   set   customer's   table
number and name
               newcustomer->PlaceOrder(menu);// invoke operations of adding
and removing items from customer's order
                if (head == NULL)// checking whether customer is the first
to be added
               {
                   head = newcustomer;
                   tail = newcustomer;
               }
               else
               {
                   tail->next = newcustomer;
                   tail = newcustomer;
               }
}
```

```
void Customerlinkedlist::deleteCustomer(string name)
{

                curr=head;
                // Search the customer to be deleted
                while ((curr != NULL) && (curr->Name != name)){
                prev = curr;
                curr = curr->next;
                }
                if (curr == NULL)
                cout << "No customer's order to be canceled....'" << endl;
               else{
                if (curr->bill.getStatus())
                {
                      cout<<"Payment has been made for this customer's
order'. This order cannot be canceled."<<endl;
                     }
                     else
                     {


                    if (curr == head) // deletion on the head
                    {
                        head = curr->next;
                        delete curr;
                    }
                    else if (curr == tail) // deletion on the tail
                    {
                        tail = prev;
                        tail->next = NULL;
                        delete curr;
                    }
                    else // item deletion in a middle of 2 nodes
                    {
                        prev->next = curr->next;
                        delete curr;
                    }
                    cout<<"The order is canceled successfully."<<endl;
                }
                }
}
Customer* Customerlinkedlist::searchCustomerbytable(string tablenum)
{
           curr=head;
```

```
                while(curr!=NULL&&curr->tableNo!=tablenum){
                      curr=curr->next;
                }
                return curr;
}
Customer* Customerlinkedlist::searchCustomerbyname(string nama)
{
           curr=head;
                while(curr!=NULL&&curr->Name!=nama){
                      curr=curr->next;
                }
                return curr;
}
void Customerlinkedlist::listCustomer()
{
      curr=head;
      if (curr==NULL)
      {
                cout<<"There is no customer yet."<<endl;
      }
      else
      {
           cout<<left<<setw(20)<<"Table     Number"<<setw(40)<<"Customer's
Name"<<setw(20)<<"Total
Amount/RM"<<setw(20)<<"Cash/RM"<<setw(20)<<"Change/RM"<<setw(10)<<"Status
"<<endl;
                while(curr!=NULL){
                           curr->printPaymentinfo();
                           curr=curr->next;
                      }
      }
}
int Customerlinkedlist::getlistsize()// Returning the size of linked list
{
      int num=0;
      curr=head;
      while (curr!=NULL)
      {
           curr=curr->next;
           num++;
      }
      return num;
}
```

Itemlinkedlist.h

```
#ifndef ITEMLINKEDLIST_H
#define ITEMLINKEDLIST_H
#include "Item.h"
#include "Node.h"
#include <iostream>
using namespace std;

class Itemlinkedlist
{
      private:
            Item *head;
            Item *prev;
            Item *curr;
            Item *tail;
      public:
            Itemlinkedlist ();
            ~Itemlinkedlist ();
            void addItem (Node *,int);
            void deleteItem (string);
            void listitem ();
            double getTotalItemPrice ();

};
#endif
```

Itemlinkedlist.cpp

```
#include "Item.h"
#include "Itemlinkedlist.h"
#include "Node.h"
#include <iostream>
#include <iomanip>
using namespace std;

Itemlinkedlist::Itemlinkedlist()
{
      head=NULL;
      prev=NULL;
      curr=NULL;
      tail=NULL;
}
Itemlinkedlist::~Itemlinkedlist()
{
           while(head!=NULL){
                      curr=head;
                      head=head->next;
                      delete head;
                 }
}
void Itemlinkedlist::addItem(Node *food,int quantity)
{
       Item *newitem = new Item;
           newitem->setitem(food,quantity);
               if (head == NULL)// checking whether item is the first to be
added
               {
                   head = newitem;
                   tail = newitem;
               }
               else
               {
                   tail->next = newitem;
                   tail = newitem;
               }
}

void Itemlinkedlist::deleteItem(string id)
{
           curr=head;
                 // Search the item to be deleted
```

```
                while ((curr != NULL) && (curr->ID != id)){
                prev = curr;
                curr = curr->next;
                }
                if (curr == NULL)
                cout << "No item can be deleted from the order" << endl;
                else{

                    if (curr == head) // deletion on the head
                    {
                        head = curr->next;
                        delete curr;
                    }
                    else if (curr == tail) // deletion on the tail
                    {
                        tail = prev;
                        tail->next = NULL;
                        delete curr;
                    }
                    else // item deletion in a middle of 2 nodes
                    {
                        prev->next = curr->next;
                        delete curr;
                    }


                }
}
void Itemlinkedlist::listitem()
{
     int num=0;
          curr=head;
               while(curr!=NULL){

     cout<<left<<setw(10)<<num+1<<setw(20)<<curr->ID<<setw(40)<<curr->nam
e;

     cout<<left<<setw(20)<<fixed<<setprecision(2)<<curr->price<<setw(20)<
<curr->quantity;

    cout<<left<<setw(20)<<fixed<<setprecision(2)<<curr->getTotal()<<endl;
                         curr=curr->next;

                         num++;
```

```
                    }

}
double Itemlinkedlist::getTotalItemPrice()
{
      curr=head;
      double total=0;

      while (curr!=NULL)
      {
            total=total+curr->getTotal();
            curr=curr->next;
      }

      return total;
}
```

Menulinkedlist.h

```
#ifndef MENULINKEDLIST_H
#define MENULINKEDLIST_H
#include "Node.h"
using namespace std;


class Menulinkedlist{
      private:
            Node *head;

      public:
            Menulinkedlist();
            Menulinkedlist(Node *);
            Node* nodeExists (string);
            void InsertNode(Node *);
            void deleteNode (string);
            void updateNode (string, double);
            void printList();
            void backtoFile(fstream &);
};
#endif
```

Menulinkedlist.cpp

```
#include "Menulinkedlist.h"
#include "Node.h"
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;

Menulinkedlist::Menulinkedlist(){
	head = NULL;
}

Menulinkedlist::Menulinkedlist(Node* n){
	head = n;
}

Node* Menulinkedlist::nodeExists(string c){
	Node * temp = NULL;
	Node * ptr = new Node;
	ptr=head;
	while (ptr != NULL){
		if (ptr->code == c){
			temp = ptr;
		}
		ptr = ptr->next;
	}
	return temp;
}

void Menulinkedlist::InsertNode(Node *n){
	Node * ptr1 = new Node;
	ptr1=NULL;
	ptr1=nodeExists(n->code);
	if (ptr1 == NULL){
		Node *curr=head;
		Node *prev=NULL;
		while (curr&&n->code>curr->code)
		{
			prev=curr;
			curr=curr->next;
		}
		Node *newnode=new Node;
		newnode=n;
		if(prev==NULL)
```

```
            {
                  newnode->next=head;
                  head=newnode;
            }
            else
            {
                  newnode->next=prev->next;
                  prev->next=newnode;
            }
      }

      else{
            cout<<n->code<<" is exist in menu. Insertion failed. "<<endl;
      }
}

void Menulinkedlist::deleteNode(string c){

      if (head == NULL){
            cout<<"No item is available to be deleted from the food
menu."<<endl;
      }

      else if (head!=NULL){
            if (head->code == c){
                  Node *tempNod=new Node;
                  tempNod=head;

                  head = head->next;
                  cout<<tempNod->food<<" with the code of
"<<tempNod->code<<" is deleted"<<endl;
                  delete tempNod;
            }

            else{
                  Node *temp = NULL;
                  Node * prevptr = head;
                  Node * currentptr = head->next;

                  while (currentptr!=NULL){
                        if (currentptr->code==c){
                              temp = currentptr;
                              currentptr = NULL;
                        }
```

```
                        else{
                              prevptr = prevptr->next;
                              currentptr = currentptr->next;
                        }
                  }

                  if (temp!=NULL){
                        prevptr->next = temp->next;
                        cout<<temp->food<<" with the code of
"<<temp->code<<" is deleted"<<endl;
                        delete temp;
                  }

                  else{
                        cout<<"The item ID entered is invalid "<<endl;
                  }
            }
      }
}

void Menulinkedlist::updateNode(string c, double p){

       Node* ptr =nodeExists(c);
       if (ptr!=NULL){

            ptr->price = p;
            cout<<"Price for "<<c<< " is updated successfully"<<endl;
       }

       else{
            cout<<c<<" is invalid."<<endl;
       }
}

void Menulinkedlist::printList(){
      if (head == NULL){
            cout<< "The food menu is empty.";
      }

      else{
            Node* temp = head;
            cout<<"Food Menu"<<endl;
```

```
                cout<<left<<setw(20)<<"Item ID"<<setw(40)<<"Item
Name"<<setw(20)<<"Price/RM"<<endl;
           while (temp!=NULL){

     cout<<left<<setw(20)<<temp->code<<setw(40)<<temp->food<<setw(20)<<fi
xed<<setprecision(2)<<temp->price<<endl;
                temp = temp->next;
           }
     }
}

void Menulinkedlist::backtoFile(fstream &infile){
     infile.open("menu.txt",ios::out);
     Node* temp = head;
     while (temp!=NULL){
           infile<< temp->code<<" "<<temp->food <<" "<<temp->price
<<endl;
           temp = temp->next;
     }
     infile.close();
}
```

Node.h

```
#ifndef NODE_H
#define NODE_H
#include <iostream>
using namespace std;

class Node{

public:
	string code;
	string food;
	double price;
	Node *next;

	Node();
	Node(string, string, double);
};

#endif
```

Node.cpp

```
#include "Node.h"
#include <iostream>

using namespace std;
Node::Node(){
	code = "";
	food = "";
	price = 0.0;
	next = NULL;
}

Node::Node(string c, string f, double p){
	code = c;
	food = f;
	price = p;
};
```

main.cpp

```
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <fstream>
#include <string>
#include "Order.h"
#include "Item.h"
#include "Customer.h"
#include "Payment.h"
#include "Customerlinkedlist.h"
#include "Itemlinkedlist.h"
#include "Node.h"
#include "Menulinkedlist.h"
using namespace std;

string user_interface = "Welcome to food ordering system. ";
string category_interface="Please choose your category.\n1. Staff\n2. Customer\n";
string customer_menu = "1 (View menu) \n2 (Place Order) \n3 (Make Payment)\n4 (Return to user menu)";
string menu_sorting = "1 (Sort ascendingly based on alphabet) \n2 (Sort descendingly based on alphabet) \n3 (Sort ascendingly based on price) \n4 (Sort descendingly based on price)";


// Inserting new item into food menu
void case1appendCode(Node * n1, Menulinkedlist *s){
	string code, food;
	bool check=false;
	double price;
	cout<<"<<<Attention>>>"<<endl;
	cout<<"No spacing is allowed for food name"<<endl;
	cout << "Enter information of item to be inserted into menu" << endl;
	cout << "Enter the code: ";
	getline(cin,code);
	do
	{
		cout << "Enter the food name: ";
	getline(cin,food);

	for (int j=0;food[j]!='\0';j++)
	{
		if (food[j]==' ')
```

```
            {
                  cout<<"Spacing is detected...Please insert again"<<endl;
                  check=true;
                  break;
            }
            else
            {
                  check=false;
            }
      }
      }
      while (check);

      cout << "Enter the price: RM ";
      cin >> price;
      n1 -> code = code;
      n1 -> food = food;
      n1 -> price = price;
      s->InsertNode(n1);
}
// Deleting item from the food menu
void case2deleteCode(Menulinkedlist *s){
      string code1;
      cout << "Delete Code Operation "<<endl;
      s->printList();
      cout<<endl<<"Enter code of the menu to be deleted " << endl;
      cout<<"Code: ";
      getline(cin,code1);
      s->deleteNode(code1);
}
// Updating the price of food item
void case3updatePrice(Menulinkedlist *s){
      string code1;
      double price;
      cout << "Update Price Operation" << endl;
      s->printList();
      cout<<endl<<"Enter the food code you wanna to update: ";
      getline(cin,code1);
      cout<<endl<<"Enter the new price for "<<code1<<": ";
      cin >> price;
      s->updateNode(code1, price);
}

void AddCustomer (Customerlinkedlist *customerlist,Menulinkedlist *menu)//
```

```
Adding a customer when placing a new order
{
      string Name,tableNo;
            cout<<"Please enter your name and table number"<<endl;
                  cout<<"Name: ";
                  getline(cin,Name);
                  cout<<"Table Number: ";
                  getline(cin,tableNo);
                  system("CLS");
                  customerlist->addCustomer(tableNo,Name,menu);
}
void DeleteCustomer (Customerlinkedlist *customerlist)
{
      string deletename;
      cout<<"Canceling customer order can only be done with manager's
permission."<<endl;
      cout<<"Customer's orders that already made payment are not allowed to
be deleted"<<endl;
      cout<<"Please enter customer's' name for the order that wishes to be
deleted"<<endl;
      cout<<"Customer's Name: ";
      getline(cin,deletename);
      customerlist->deleteCustomer(deletename);


}


int main(){

      int category;
      bool loopcategory=true;
      Customerlinkedlist *list=new Customerlinkedlist;
      Menulinkedlist *menu=new Menulinkedlist;
      fstream infile;
      cout<<user_interface<<endl<<endl;
      system ("PAUSE");
      system("CLS");
      int count=0;



      if (!infile)
      {
```

```
      cout<<"The input file does not exist."<<endl;
}
else
{
           infile.open("menu.txt",ios::in);
string msg;
while (getline(infile,msg)){
      count++;
}

infile.close();


infile.open("menu.txt",ios::in);
string code,food;
double price;
for(int i=0; i<count; i++){
      Node *add = new Node;
      infile>>code>>food>>price;
      add -> code = code;
     add -> food = food;
     add -> price = price;
     menu->InsertNode(add);
}

infile.close();
do
{

cout<<"USER MENU"<<endl;
cout<<"---------"<<endl;
cout<<category_interface<<endl;
cout<<"Or\n3. Exit program"<<endl;
cout<<"Option: ";
cin>>category;
cin.ignore();
system("CLS");
if (category==1)
{
      int staff_operation;
      do
      {
            cout<<"Staff Menu"<<endl;
            cout<<"----------"<<endl;
```

```
                cout<<"1. View customer's payment\n2. Cancel customer's
order\n";
                cout<<"3. Modify food menu \n4. Return to user menu"<<endl;
                cout<<"Option: ";
                cin>>staff_operation;
                cin.ignore();
                system("CLS");
                if (staff_operation==1)
                {
                      list->listCustomer();
                      cout<<endl<<endl;

                }
                else if (staff_operation==2)
                {
                      DeleteCustomer(list);
                      cout<<endl<<endl;

                }
                else if (staff_operation==3)
                {
                      int optionnum;


      do{
            cout << "What operation do you want to perform? Select Option
number." << endl;
            cout << "1. Insert new item into the food menu" << endl;
            cout << "2. Remove item from the food menu" << endl;
            cout << "3. Update price of item in the food menu" << endl;
            cout << "4. View updated food menu" << endl;
            cout<<"5. Exit"<<endl;

            cout<<"Option: ";
            cin >> optionnum;
            cin.ignore();
            Node * n1 = new Node();
            //Node n1;
            system("CLS");
            switch (optionnum) {
            case 1:
                  case1appendCode(n1,menu);
                  break;
```

```
        case 2:
                        case2deleteCode(menu);
                        break;
            case 3:
                        case3updatePrice(menu);
                        break;

      case 4:
                  cout<< "This is the latest menu of KLW restaurant"<<endl;
                  menu->printList();

                  break;
            case 5:
                  break;
      default:
                  cout << "Please enter the available option number. " <<
endl;
            }
            system("PAUSE");
            system("CLS");
   } while (optionnum != 5);

   menu->backtoFile(infile);
                  }

            else if (staff_operation==4)
            {
                  cout<<"Returning to user menu....."<<endl;
                        system("CLS");
            }
      }
      while (staff_operation!=4);


      }
      else if (category==2)
      {



      A:
            cout<<"CUSTOMER MENU"<<endl;
            cout<<"-------------"<<endl;
```

```
	cout<<customer_menu<<endl;
	int command;
	cout<<"Insert your command: ";
	cin>>command;
	system("CLS");
	while (command == 1){
		menu->printList();

		cout<<"Press 0 to return to customer Menu : ";
			cin>>command;
			if (command==0)
			{
				system("CLS");
				goto A;

			}


		system("CLS");
	}
	while (command==2)
	{
		cin.ignore();
		AddCustomer(list,menu);
		goto A;
	}
	while (command==3)
	{
			int option,index_match,operation2;
			string key;
			string tempArray[list->getlistsize()];
		V:
			cout<<"1. (Search by table number)\n2. (Search by
customer's name)"<<endl;
			cout<<"Option: ";
			cin>>option;
			cin.ignore();
		if (option==1)
			{
				Customer *tempcustomer;
				cout<<"Please enter your table number"<<endl;
				cout<<"Table No: ";
				getline(cin,key);
				system("CLS");
```

```
                        tempcustomer=list->searchCustomerbytable(key);



                        if (tempcustomer==NULL)
                        {
                              cout<<"No  result  is  found.\nPlease  try
again"<<endl;
                              goto V;
                        }
                        else
                        {
                              tempcustomer->MakePayment();
                              cout<<endl<<"Do you wish to"<<endl;
                             cout<<"1. Make another payment"<<endl<<"2.
Return to customer menu"<<endl;
                              cout<<"Option: ";
                              cin>>operation2;
                              system("CLS");
                              if (operation2==1)
                              {
                                    goto V;
                              }
                              else
                              {
                                    goto A;
                              }
                        }
                  }
      else if (option==2)
      {

            cout<<"Please enter your name"<<endl;
                        cout<<"Name: ";
                        getline(cin,key);

                        Customer *tempcustomer;
                        tempcustomer=list->searchCustomerbyname(key);



                        if (tempcustomer==NULL)
                        {
                              cout<<"No  result  is  found.\nPlease  try
again"<<endl;
```

```
                                    goto V ;
                              }
                              else
                              {
                                    tempcustomer->MakePayment();
                                    cout<<"Do you wish to"<<endl;
                                   cout<<"1. Make another payment"<<endl<<"2.
Return to customer menu"<<endl;
                                    cout<<"Option: ";
                                    cin>>operation2;
                                    system("CLS");
                                    if (operation2==1)
                                    {
                                          goto V;
                                    }
                                    else
                                    {
                                          goto A;
                                    }
                              }
      }
}
                if (command==4)
                {
                      cout<<"Returning to user menu....."<<endl;
                      system("CLS");
                }

}
else if (category==3)
{
      loopcategory=false;
}
else
{
      cout<<"Invalid input.....Please try again"<<endl<<endl;
      system("PAUSE");
      system("CLS");
}
}
while (loopcategory);
}
      return 0;
}
```