# Week 3 Assignment – Food Ordering and Delivery System

## Food Ordering and Delivery System

This project models a **Food Ordering and Delivery System** using Object-Oriented Programming (OOP) concepts. It employs **linked lists**, **dynamic arrays**, and **circular linked lists** to manage various aspects of the system, including restaurants, menu items, orders, and delivery of orders.

**Features of the System**

1. **Manage Restaurants**:

Add, update, and view restaurants.

1. **Manage Menu Items**:

Add and display menu items for each restaurant.

1. **Place Orders**:

Customers can place orders from a specific restaurant.

1. **Process and Deliver Orders**:

The system processes the next order in the queue.

Solution:

// Food ordering and delivery system

/\*

**System Design**

1. Array of Pointers: Used for storing restaurants. Each restaurant pointer points to a dynamically allocated restaurant object.
2. Linked List: Used to store the menu items of each restaurant.
3. Circular Linked List: Used to manage orders, where the last order links back to the first order, allowing efficient cyclic processing of orders.

\*/

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class MenuItems

{

public:

string itemName;

double price;

MenuItems(const string& name, double price) : itemName(name), price(price) {}

};

// Restaurant Class (Linked List of Menu Items)

class Restaurant {

public:

int restId;

string restName;

vector<MenuItems> menu;

Restaurant(int id, const string& name) : restId(id), restName(name) {}

};

// Order Class for Circular Linked List

class Order

{

public:

int orderId;

int restaurantId;

string customerName;

vector<MenuItems> items;

double totalPrice;

Order\* next;

Order(int orderId, int restaurantId, const string& customerName, const vector<MenuItems>& items)

: orderId(orderId), restaurantId(restaurantId), customerName(customerName), items(items), next(nullptr) {

totalPrice = 0;

for (const auto& item : items) {

totalPrice += item.price;

}

}

};

class FoodOrderingSystem

{

private:

Restaurant\*\* restaurants; // Array of pointers to restaurants

int maxRestaurants; // Maximum number of restaurants

int restaurantCount; // Current number of restaurants

Order\* orderHead; // Circular linked list for orders

int orderCount; // Order ID counter

public:

FoodOrderingSystem(int maxRestaurants)

: maxRestaurants(maxRestaurants), restaurantCount(0), orderHead(nullptr), orderCount(0)

{

restaurants = new Restaurant\*[maxRestaurants]; // Allocate array of pointers

for (int i = 0; i < maxRestaurants; ++i)

{

restaurants[i] = nullptr;

}

}

~FoodOrderingSystem()

{

for (int i = 0; i < restaurantCount; ++i)

{

delete restaurants[i];

}

delete[] restaurants;

// Delete orders in circular linked list

while (orderHead)

{

processOrder();

}

}

// Add a new restaurant

void addRestaurant(const string& name)

{

if (restaurantCount >= maxRestaurants)

{

cout << "Maximum number of restaurants reached"<<endl;

return;

}

restaurants[restaurantCount++] = new Restaurant(restaurantCount, name);

cout << "Restaurant " << name << " added with ID: " << restaurantCount << endl;

}

// Add a menu item to a restaurant

void addMenuItem(int restaurantId, const string& itemName, double price)

{

if (restaurantId < 1 || restaurantId > restaurantCount)

{

cout << "Invalid restaurant ID"<<endl;

return;

}

restaurants[restaurantId - 1]->menu.push\_back(MenuItems(itemName, price));

cout << "Item " << itemName << " added to restaurant " << restaurants[restaurantId - 1]->restName <<endl;

}

// Display menu for a restaurant

void displayMenu(int restaurantId) {

if (restaurantId < 1 || restaurantId > restaurantCount) {

cout << "Invalid restaurant ID"<<endl;

return;

}

cout <<endl<< "Menu for " <<

restaurants[restaurantId - 1]->restName << ":"<<endl;

for (size\_t i = 0; i < restaurants[restaurantId - 1]->menu.size(); ++i)

{

cout << i + 1 << ". " << restaurants[restaurantId - 1]->menu[i].itemName

<< " - Rs." << restaurants[restaurantId - 1]->menu[i].price << endl;

}

}

// Place an order

void placeOrder(int restaurantId, const string& customerName, const vector<int>& itemIndex)

{

if (restaurantId < 1 || restaurantId > restaurantCount)

{

cout << "Invalid restaurant ID"<<endl;

return;

}

vector<MenuItems> items;

for (int index : itemIndex)

{

if (index > 0 && index <= (int)restaurants[restaurantId - 1]->menu.size())

{

items.push\_back(restaurants[restaurantId - 1]->menu[index - 1]);

}

else

{

cout << "Invalid menu item index: " << index << endl;

return;

}

}

Order\* newOrder = new Order(++orderCount, restaurantId, customerName, items);

if (!orderHead)

{

orderHead = newOrder;

orderHead->next = orderHead;

}

else

{

Order\* temp = orderHead;

while (temp->next != orderHead)

{

temp = temp->next;

}

temp->next = newOrder;

newOrder->next = orderHead;

}

cout << endl<<"Order ID: " << orderCount <<" placed successfully";

}

// Process the next order in the queue

void processOrder()

{

if (!orderHead)

{

cout << "No orders to process.\n";

return;

}

Order\* temp = orderHead;

if (temp->next == orderHead)

{

orderHead = nullptr;

}

else

{

Order\* prev = temp;

while (prev->next != orderHead)

{

prev = prev->next;

}

orderHead = orderHead->next;

prev->next = orderHead;

}

cout << endl<<"Processing Order ID: " << temp->orderId << endl;

cout << "Customer: " << temp->customerName << "\n";

cout << "Items:"<<endl;

for (const auto& item : temp->items)

{

cout << "- " << item.itemName << ": Rs." << item.price << endl;

}

cout << "Total Price: Rs." << temp->totalPrice << endl;

delete temp;

}

};

int main()

{

FoodOrderingSystem fos(3);

// Adding restaurants

fos.addRestaurant("Pizza Hut");

fos.addRestaurant("Burger King");

fos.addRestaurant("Starbucks");

// Adding menu items

fos.addMenuItem(1, "Margherita Pizza", 169.00);

fos.addMenuItem(1, "Classic Corn", 109.00);

fos.addMenuItem(1, "Murg Malai Chicken", 379.00);

fos.addMenuItem(2, "Crispy Chicken Burger", 99.00);

fos.addMenuItem(2, "King Fries", 139.00);

fos.addMenuItem(2, "Crispy Veg Double Patty", 198.00);

fos.addMenuItem(3, "Cappuccino", 275.00);

fos.addMenuItem(3, "Cafe Mocha", 280.00);

// Display menu for a restaurant

cout << endl<<"--- Menu for Restaurant ---"<<endl;

fos.displayMenu(1);

fos.displayMenu(2);

fos.displayMenu(3);

// Place orders

fos.placeOrder(1, "A", {1, 2});

fos.placeOrder(2, "B", {1, 3});

fos.placeOrder(3, "C", {1});

// Process orders

cout << endl<<endl;

cout<<"--- Processing Orders ---"<<endl;

fos.processOrder();

fos.processOrder();

fos.processOrder();

fos.processOrder(); // No orders left

return 0;

}