Food Ordering System

```
#include <iostream>

#include <string>

using namespace std;

struct Order {

    int orderId;

    string customerName;

    string orderDetails;

    Order() {}

    Order(int id, const string &name, const string &details) : orderId(id), customerName(name),
orderDetails(details) {}

    void display() const {

        cout << "Order ID: " << orderId << endl;

        cout << "Customer Name: " << customerName << endl;

        cout << "Order Details: " << orderDetails << endl;

    }

};

class OrderProcessingSystem {

private:

    Order *orderQueue;

    int capacity;

    int front;

    int rear;

    int size;
```

```
    int nextOrderId;
    void resizeQueue() {
        int newCapacity = capacity * 2;
        Order *newQueue = new Order[newCapacity];
        for (int i = 0; i < size; i++) {
            newQueue[i] = orderQueue[(front + i) % capacity];
        }
        delete[] orderQueue;
        orderQueue = newQueue;
        capacity = newCapacity;
        front = 0;
        rear = size;
    }
public:
    OrderProcessingSystem(int initialCapacity = 10) : capacity(initialCapacity), front(0), rear(0), size(0),
nextOrderId (1) {
        orderQueue = new Order[capacity];
    }
    ~OrderProcessingSystem() {
        delete[] orderQueue;
    }
    void addOrder() {
        string customerName, orderDetails;
        cout << "Enter customer name: ";
        cin.ignore();
        getline(cin, customerName);
```

```
        cout << "Enter order details: ";
        getline(cin, orderDetails);
        if (size == capacity) {
            resizeQueue();
        }
        Order newOrder(nextOrderId, customerName, orderDetails);
        orderQueue[rear] = newOrder;
        rear = (rear + 1) % capacity;
        size++;
        nextOrderId++;
        cout << "Order added successfully!" << endl;
    }
    void processOrder() {
        if (size == 0) {
            cout << "No orders to process." << endl;
            return;
        }
        Order orderToProcess = orderQueue[front];
        cout << "\nProcessing the following order:" << endl;
        orderToProcess.display();
        front = (front + 1) % capacity;
        size--;
        cout << "Order processed successfully!" << endl;
    }
    void displayOrders() {
```

```
        if (size == 0) {
            cout << "No orders in the queue." << endl;
            return;
        }
        cout << "\nDisplaying all orders in the queue:" << endl;
        for (int i = 0; i < size; i++) {
            int index = (front + i) % capacity;
            orderQueue[index].display();
            cout << "--------------------" << endl;
        }
    }
    void searchOrder(int orderId) {
        bool found = false;
        for (int i = 0; i < size; i++) {
            int index = (front + i) % capacity;
            if (orderQueue[index].orderId == orderId) {
                cout << "\nOrder found:" << endl;
                orderQueue[index].display();
                found = true;
                break;
            }
        }
        if (!found) {
            cout << "\nOrder ID " << orderId << " not found!";
        }
```

```
    }
};
int main() {
    OrderProcessingSystem ops;
    int id;
    int choice;
    do {
        cout << "\nOrder Processing System" << endl;
        cout << "1. Add Order" << endl;
        cout << "2. Process Order" << endl;
        cout << "3. Display Orders" << endl;
        cout << "4. search Order" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
        case 1:
            ops.addOrder();
            break;
        case 2:
            ops.processOrder();
            break;
        case 3:
            ops.displayOrders();
            break;
```

```
        case 4:
            cout << "Enter Oreder ID: " << endl;
            cin >> id;
            ops.searchOrder(id);
            break;
        case 5:
            cout << "Exiting the program." << endl;
            break;
        default:
            cout << "Invalid choice. Please try again." << endl;
            break;
        }
    } while (choice != 5);
    return 0;
}
```

```
Order Processing System
1. Add Order
2. Process Order
3. Display Orders
4. search Order
5. Exit
Enter your choice: 1
Enter customer name: John
Enter order details: Pizza
Order added successfully!

Order Processing System
1. Add Order
2. Process Order
3. Display Orders
4. search Order
5. Exit
Enter your choice: 1
Enter customer name: Ken
Enter order details: Sanwich
Order added successfully!

Order Processing System
1. Add Order
2. Process Order
3. Display Orders
4. search Order
5. Exit
Enter your choice: 3

Displaying all orders in the queue:
Order ID: 1
Customer Name: John
Order Details: Pizza
--------------------
Order ID: 2
Customer Name: Ken
Order Details: Sanwich
--------------------
```

```
Order Processing System
1. Add Order
2. Process Order
3. Display Orders
4. search Order
5. Exit
Enter your choice: 4
Enter Oreder ID:
1

Order found:
Order ID: 1
Customer Name: John
Order Details: Pizza

Order Processing System
1. Add Order
2. Process Order
3. Display Orders
4. search Order
5. Exit
Enter your choice: 2

Processing the following order:
Order ID: 1
Customer Name: John
Order Details: Pizza
Order processed successfully!

Order Processing System
1. Add Order
2. Process Order
3. Display Orders
4. search Order
5. Exit
Enter your choice: 2

Processing the following order:
Order ID: 2
Customer Name: Ken
Order Details: Sanwich
Order processed successfully!

Order Processing System
1. Add Order
2. Process Order
3. Display Orders
4. search Order
5. Exit
Enter your choice: 5
Exiting the program.
PS C:\personal_documents\CSE\SEM 4\DS>
```