ROSHNI RAZA

DSA LAB 9

SAP: 61278

**TASK 1:**

#include <iostream>

#define MAX 5

using namespace std;

class OrderQueue {

private:

int orders[MAX];

int front, rear;

public:

OrderQueue() {

front = -1;

rear = -1;

}

bool isFull() {

return (rear == MAX - 1);

}

bool isEmpty() {

return (front == -1 || front > rear);

}

void addOrder(int orderNo) {

if (isFull()) {

cout << "Sorry, teaCafe can't take more orders right now. Order #" << orderNo << " rejected." << endl;

} else {

if (front == -1) front = 0;

rear++;

orders[rear] = orderNo;

cout << "teaCafe.add order 00" << orderNo << " ??" << endl;

}

}

void processOrder() {

if (isEmpty()) {

cout << "No pending orders. All served!" << endl;

} else {

cout << "teaCafe is now serving order 00" << orders[front] << " ?" << endl;

front++;

}

}

void showOrders() {

if (isEmpty()) {

cout << "Queue is clear. No active orders." << endl;

} else {

cout << "Current queue at teaCafe: ";

for (int i = front; i <= rear; i++) {

cout << "[00" << orders[i] << "] ";

}

cout << endl;

}

}

};

int main() {

OrderQueue teaCafe;

teaCafe.processOrder(); // no orders yet

teaCafe.addOrder(1);

teaCafe.addOrder(2);

teaCafe.addOrder(3);

teaCafe.addOrder(4);

teaCafe.addOrder(5);

teaCafe.addOrder(6); // should show full

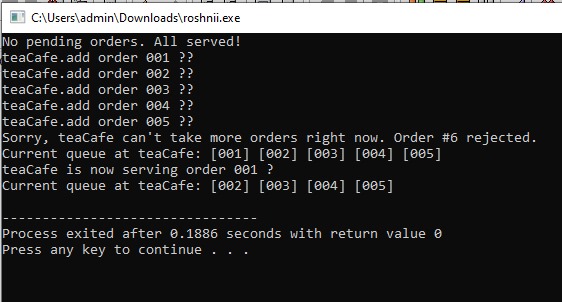
teaCafe.showOrders();

teaCafe.processOrder(); // serve one

teaCafe.showOrders(); // show after serving

return 0;

}



**TASK 2:**

#include <iostream>

using namespace std;

class Ticket {

public:

int id;

Ticket\* next;

};

class TicketLine {

private:

Ticket\* start;

Ticket\* end;

public:

TicketLine() {

start = end = NULL;

}

void newBooking(int id);

void serveBooking();

void listBookings();

~TicketLine();

};

void TicketLine::newBooking(int id) {

Ticket\* temp = new Ticket();

temp->id = id;

temp->next = NULL;

if (start == NULL) {

start = end = temp;

} else {

end->next = temp;

end = temp;

}

cout << "Ticket #" << id << " is now in line ???" << endl;

}

void TicketLine::serveBooking() {

if (start == NULL) {

cout << "?? No tickets to serve. Line is empty." << endl;

return;

}

cout << "Now serving Ticket #" << start->id << " ??" << endl;

Ticket\* temp = start;

if (start == end) {

start = end = NULL;

} else {

start = start->next;

}

delete temp;

}

void TicketLine::listBookings() {

if (start == NULL) {

cout << "?? Ticket line is clear. No current bookings." << endl;

return;

}

Ticket\* walker = start;

cout << "?? Tickets waiting: ";

while (walker != NULL) {

cout << "#" << walker->id << " ";

walker = walker->next;

}

cout << endl;

}

TicketLine::~TicketLine() {

while (start != NULL) {

Ticket\* temp = start;

start = start->next;

delete temp;

}

end = NULL;

}

int main() {

TicketLine movieLine;

movieLine.listBookings();

movieLine.newBooking(501);

movieLine.newBooking(502);

movieLine.newBooking(503);

movieLine.newBooking(504);

movieLine.listBookings();

movieLine.serveBooking();

movieLine.serveBooking();

movieLine.listBookings();

movieLine.serveBooking();

movieLine.serveBooking();

movieLine.serveBooking();

return 0;

}

