8) The ticket booking system of Cinemax theater has to be implemented using C++ program. There are 10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to each row. On demand a) The list of available seats is to be displayed b) The seats are to be booked c) The booking can be cancelled.

***CODE :-***

#include <iostream>

#include <iomanip>

using namespace std;

struct Seat

{

int seatNumber;

bool isBooked;

Seat\* next;

Seat\* prev;

Seat(int num) : seatNumber(num), isBooked(false), next(nullptr), prev(nullptr) {}

};

class Row

{

public:

Seat\* head;

Row()

{

head = nullptr;

for (int i = 1; i <= 7; i++)

{

addSeat(i);

}

}

void addSeat(int seatNumber)

{

Seat\* newSeat = new Seat(seatNumber);

if (!head)

{

head = newSeat;

head->next = head;

head->prev = head;

}

else

{

Seat\* last = head->prev;

last->next = newSeat;

newSeat->prev = last;

newSeat->next = head;

head->prev = newSeat;

}

}

void displayAvailableSeats()

{

Seat\* temp = head;

do

{

if (!temp->isBooked)

{

cout << "Seat " << temp->seatNumber << " is available.\n";

}

temp = temp->next;

} while (temp != head);

}

bool bookSeat(int seatNumber)

{

Seat\* temp = head;

do

{

if (temp->seatNumber == seatNumber && !temp->isBooked)

{

temp->isBooked = true;

cout << "Seat " << seatNumber << " has been successfully booked.\n";

return true;

}

temp = temp->next;

} while (temp != head);

cout << "Seat " << seatNumber << " is either not available or already booked.\n";

return false;

}

bool cancelBooking(int seatNumber)

{

Seat\* temp = head;

do

{

if (temp->seatNumber == seatNumber && temp->isBooked)

{

temp->isBooked = false;

cout << "Booking for seat " << seatNumber << " has been successfully cancelled.\n";

return true;

}

temp = temp->next;

} while (temp != head);

cout << "Seat " << seatNumber << " is not booked.\n";

return false;

}

};

class Cinemax

{

private:

Row\* rows[10]; // Array of pointers to rows (10 rows)

public:

Cinemax()

{

for (int i = 0; i < 10; i++)

{

rows[i] = new Row();

}

}

void displayAvailableSeats()

{

for (int i = 0; i < 10; i++)

{

cout << "Row " << i + 1 << ":\n";

rows[i]->displayAvailableSeats();

cout << "\n";

}

}

void bookSeat(int row, int seatNumber)

{

if (row < 1 || row > 10)

{

cout << "Invalid row number.\n";

return;

}

rows[row - 1]->bookSeat(seatNumber);

}

void cancelBooking(int row, int seatNumber)

{

if (row < 1 || row > 10)

{

cout << "Invalid row number.\n";

return;

}

rows[row - 1]->cancelBooking(seatNumber);

}

};

int main()

{

Cinemax cinema;

int choice, row, seatNumber;

while (true)

{

cout << "Cinemax Ticket Booking System\n";

cout << "1. Display Available Seats\n";

cout << "2. Book Seat\n";

cout << "3. Cancel Booking\n";

cout << "4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice)

{

case 1:

cinema.displayAvailableSeats();

break;

case 2:

cout << "Enter row number (1-10): ";

cin >> row;

cout << "Enter seat number (1-7): ";

cin >> seatNumber;

cinema.bookSeat(row, seatNumber);

break;

case 3:

cout << "Enter row number (1-10): ";

cin >> row;

cout << "Enter seat number (1-7): ";

cin >> seatNumber;

cinema.cancelBooking(row, seatNumber);

break;

case 4:

cout << "Exiting the system. Goodbye!\n";

return 0;

default:

cout << "Invalid choice. Please try again.\n";

}

}

return 0;

}

***OUTPUT :-***







