#include <iostream>

using namespace std;

// Structure for a seat

struct Seat {

int seatNumber;

bool isBooked; // true if the seat is booked, false otherwise

Seat\* next;

Seat\* prev;

};

// Function to create a new seat node

Seat\* createSeat(int seatNumber) {

Seat\* newSeat = new Seat();

newSeat->seatNumber = seatNumber;

newSeat->isBooked = false;

newSeat->next = nullptr;

newSeat->prev = nullptr;

return newSeat;

}

// Function to create a row with 7 seats (doubly circular linked list)

Seat\* createRow() {

Seat\* head = nullptr;

Seat\* tail = nullptr;

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

Seat\* newSeat = createSeat(i);

if (head == nullptr) {

head = newSeat;

tail = newSeat;

head->next = head;

head->prev = head;

} else {

tail->next = newSeat;

newSeat->prev = tail;

newSeat->next = head;

head->prev = newSeat;

tail = newSeat;

}

}

return head;

}

// Function to display available seats in a row

void displayAvailableSeats(Seat\* head) {

Seat\* temp = head;

cout << "Available seats: ";

do {

if (!temp->isBooked) {

cout << temp->seatNumber << " ";

}

temp = temp->next;

} while (temp != head);

cout << endl;

}

// Function to book a seat

void bookSeat(Seat\* head, int seatNumber) {

Seat\* temp = head;

do {

if (temp->seatNumber == seatNumber) {

if (!temp->isBooked) {

temp->isBooked = true;

cout << "Seat " << seatNumber << " booked successfully." << endl;

} else {

cout << "Seat " << seatNumber << " is already booked." << endl;

}

return;

}

temp = temp->next;

} while (temp != head);

cout << "Seat " << seatNumber << " not found." << endl;

}

// Function to cancel a booking

void cancelBooking(Seat\* head, int seatNumber) {

Seat\* temp = head;

do {

if (temp->seatNumber == seatNumber) {

if (temp->isBooked) {

temp->isBooked = false;

cout << "Seat " << seatNumber << " booking cancelled successfully." << endl;

} else {

cout << "Seat " << seatNumber << " is not booked." << endl;

}

return;

}

temp = temp->next;

} while (temp != head);

cout << "Seat " << seatNumber << " not found." << endl;

}

int main() {

const int rows = 10;

Seat\* rowHeads[rows];

// Create rows

for (int i = 0; i < rows; i++) {

rowHeads[i] = createRow();

}

// Initial random bookings (for demonstration)

bookSeat(rowHeads[0], 1);

bookSeat(rowHeads[0], 3);

bookSeat(rowHeads[1], 2);

// Menu-driven system

int choice, row, seat;

do {

cout << "\n1. Display available seats\n2. Book a seat\n3. Cancel a booking\n4. Exit\nEnter your choice: ";

cin >> choice;

switch (choice) {

case 1:

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

cin >> row;

if (row < 1 || row > 10) {

cout << "Invalid row number!" << endl;

} else {

displayAvailableSeats(rowHeads[row - 1]);

}

break;

case 2:

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

cin >> row;

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

cin >> seat;

if (row < 1 || row > 10 || seat < 1 || seat > 7) {

cout << "Invalid row or seat number!" << endl;

} else {

bookSeat(rowHeads[row - 1], seat);

}

break;

case 3:

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

cin >> row;

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

cin >> seat;

if (row < 1 || row > 10 || seat < 1 || seat > 7) {

cout << "Invalid row or seat number!" << endl;

} else {

cancelBooking(rowHeads[row - 1], seat);

}

break;

case 4:

cout << "Exiting..." << endl;

break;

default:

cout << "Invalid choice!" << endl;

}

} while (choice != 4);

return 0;

}