#include <iostream>

#include <unordered\_map>

#include <string>

#include <limits>

using namespace std;

// Structure to store car information

struct Car {

string licensePlate;

string ownerName;

};

// Class to manage parking

class ParkingManagementSystem {

private:

unordered\_map<int, Car> parkingSpots; // Map of parking spots with their car details

int totalSpots; // Total number of parking spots in the university

public:

ParkingManagementSystem(int spots) : totalSpots(spots) {}

// Function to park a car

void parkCar(int spotId, const string& licensePlate, const string& ownerName) {

if (spotId < 1 || spotId > totalSpots) {

cout << "Invalid parking spot ID!" << endl;

return;

}

if (parkingSpots.find(spotId) != parkingSpots.end()) {

cout << "Parking spot " << spotId << " is already occupied!" << endl;

} else {

parkingSpots[spotId] = {licensePlate, ownerName};

cout << "Car parked in spot " << spotId << " successfully!" << endl;

}

}

// Function to check if a spot is occupied

void checkSpot(int spotId) {

if (spotId < 1 || spotId > totalSpots) {

cout << "Invalid parking spot ID!" << endl;

return;

}

if (parkingSpots.find(spotId) != parkingSpots.end()) {

cout << "Spot " << spotId << " is occupied by "

<< parkingSpots[spotId].ownerName << " (License: "

<< parkingSpots[spotId].licensePlate << ")." << endl;

} else {

cout << "Spot " << spotId << " is available." << endl;

}

}

// Function to remove a car from a spot

void removeCar(int spotId) {

if (spotId < 1 || spotId > totalSpots) {

cout << "Invalid parking spot ID!" << endl;

return;

}

if (parkingSpots.find(spotId) != parkingSpots.end()) {

parkingSpots.erase(spotId);

cout << "Car removed from spot " << spotId << " successfully!" << endl;

} else {

cout << "No car is parked in spot " << spotId << "." << endl;

}

}

// Function to show all parked cars

void showAllParkedCars() {

if (parkingSpots.empty()) {

cout << "No cars are parked currently." << endl;

} else {

cout << "Cars currently parked:" << endl;

for (const auto& entry : parkingSpots) {

cout << "Spot " << entry.first << ": "

<< entry.second.ownerName << " (License: "

<< entry.second.licensePlate << ")" << endl;

}

}

}

};

int main() {

int totalSpots;

cout << "Enter total number of parking spots in the university: ";

cin >> totalSpots;

ParkingManagementSystem parkingSystem(totalSpots);

while (true) {

int choice;

cout << "\nUniversity Parking Management System\n";

cout << "1. Park a car\n";

cout << "2. Check parking spot\n";

cout << "3. Remove car from a spot\n";

cout << "4. Show all parked cars\n";

cout << "5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: {

int spotId;

string licensePlate, ownerName;

cout << "Enter parking spot ID (1 to " << totalSpots << "): ";

cin >> spotId;

cout << "Enter car license plate: ";

cin >> licensePlate;

cout << "Enter owner's name: ";

cin.ignore(); // To clear the newline character from input buffer

getline(cin, ownerName);

parkingSystem.parkCar(spotId, licensePlate, ownerName);

break;

}

case 2: {

int spotId;

cout << "Enter parking spot ID to check: ";

cin >> spotId;

parkingSystem.checkSpot(spotId);

break;

}

case 3: {

int spotId;

cout << "Enter parking spot ID to remove car from: ";

cin >> spotId;

parkingSystem.removeCar(spotId);

break;

}

case 4:

parkingSystem.showAllParkedCars();

break;

case 5:

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

return 0;

default:

cout << "Invalid choice. Please try again!" << endl;

}

}

return 0;

}