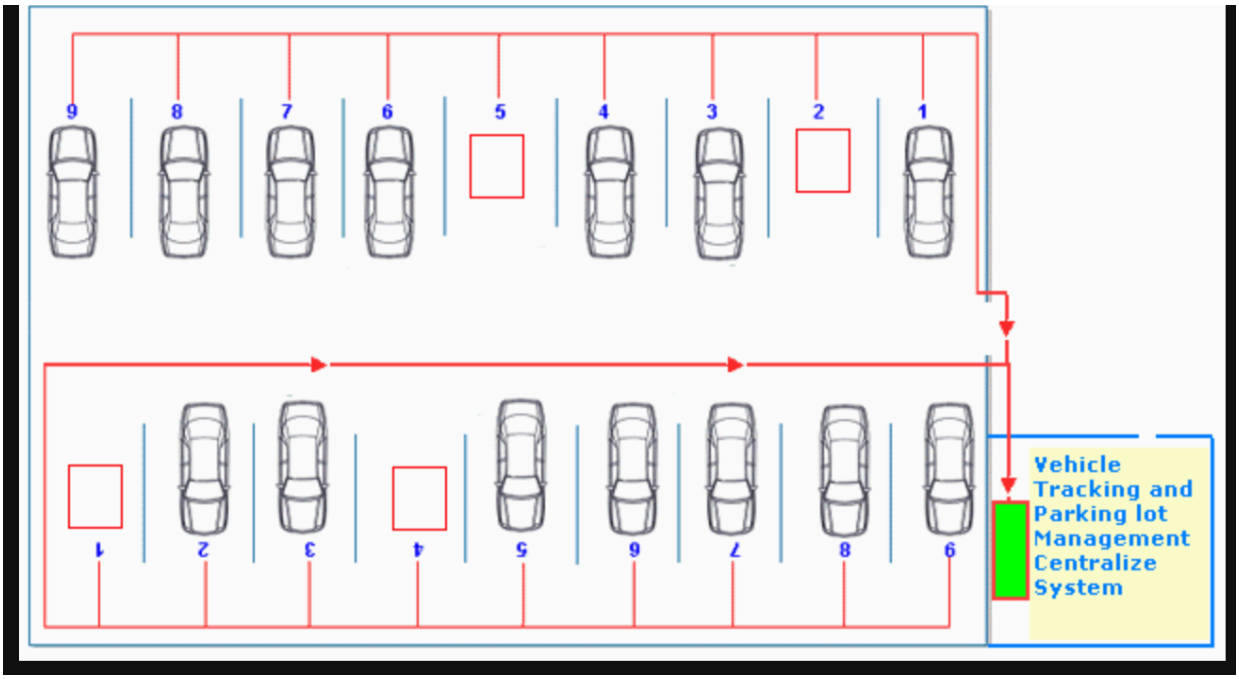

PARKING LOT
MANAGEMENT
SOFTWARE


9
8
7
6
5
4
3
2
1
Vehicle
Tracking and
Parking lot
Management
Centralize
System


P
P

Project Report

Parking Lot Management (with Charges & Slots)

Submitted by:

Name:- Divyanshu Shekhar
SAPID:- 590026263
Batch:- B59
Course:- Btech(CSE)

Submitted to:

Instructor Name:- Prashant Trivedi

Academic Year: 2024 – 2025

Department of Computer Science
UPES,Dehradun,Uttarakhand

Abstract:-

This mini project is made for managing a vehicle parking lot. This project is implemented in c programming language. The system efficiently manages parking lot (parked vehicle, available spot etc.) for different types of vehicles such as bike, car, trucks.

On every vehicle entry it notes details such as vehicle number, vehicle type parking duration, and calculates its charges accordingly. The system uses arrays and structures to store data.

Overall, the system provides a basic yet effective simulation of a real-world parking lot, showcasing how C programming can be used to solve practical management problems.

Problem Definition:-

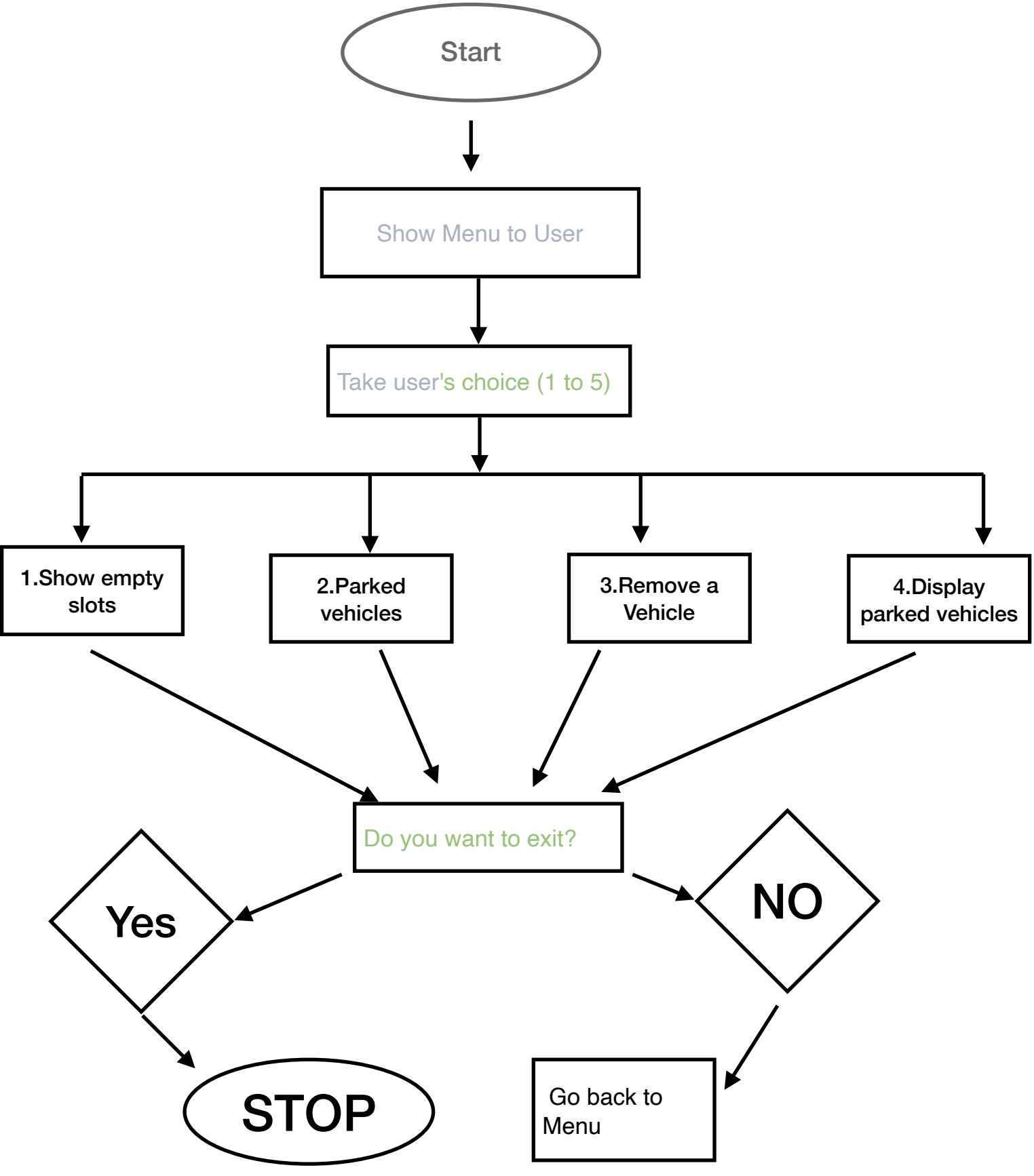
In many places, managing a parking area is still done manually—writing vehicle numbers in a notebook, remembering which slot is free, and calculating charges by hand. This often leads to confusion, slow processing and mistakes. When the parking area gets crowded, it becomes even harder to keep track of everything.

To solve this, we need a simple computer program that can help manage parkng in a more organized way. The program should be able to show which slots are empty, allow a vehicle to be parked by entering basic details, calculate the parking charge automatically, and remove the vehicle when it leaves. It should also display all the vehicles currently parked so the user can quickly check the status of the parking lot.

The aim of the project is to create a small but useful Vehicle Parking Managment System in C that reduces manual work, avoids errors and makes the parking process faster and smoother.

System Design (Flowcharts, Algorithms):-

1. Flowchart:-



2. Algorithms:-

Algorithm: Show Available Slots

1. Start
2. Repeat for every slot from 1 to MAX_SLOTS
3. If the slot is empty, print that it is empty
4. End

Algorithm: Park a Vehicle

1. Start
2. Search for the first empty slot
3. If no empty slot is found, print “Parking Full” and stop
4. Ask the user to enter vehicle number
5. Ask the user to enter vehicle type
6. Ask the user to enter number of hours
7. Calculate the parking charge
8. Store all details in that slot
9. Mark the slot as filled
10. Print that the vehicle is parked
11. End

Algorithm: Remove a Vehicle

1. Start
2. Ask the user to enter the vehicle number
3. Search the parking slots for the matching number
4. If the vehicle is found, print the slot number and charge
5. Mark the slot as empty
6. If the vehicle is not found, print “Vehicle not found”
7. End

Algorithm: Display Parked Vehicles

1. Start
2. Repeat for every slot
3. If the slot is filled, print the vehicle details
4. End

Implementation Details (with snippets):-

This System is written in the C programming language. It uses structures, arrays, and functions to manage the parking lot.

1. Structure for Vehicle Details:-

```
struct Vehicle {
    char number[15];
    char type[10];
    int hours;
    float charge;
    int slot;
    int isParked;
};
```

2. Array to Store Parking Slots:-

➜ Max limit for slots is 20.
➜ Each slot is empty at start.

```
#define MAX_SLOTS 20
struct Vehicle parking[MAX_SLOTS];

for (int i = 0; i < MAX_SLOTS; i++) {
    parking[i].isParked = 0;
```

3. Function to Calculate Parking Charge:-

```
float calculateCharge(char type[], int hours) {
    float rate;
    if (strcmp(type, "car") == 0) rate = 20;
    else if (strcmp(type, "bike") == 0) rate = 10;
    else if (strcmp(type, "truck") == 0) rate = 30;
    else rate = 15;

    return rate * hours;
}
```

4. Showing Available Slots:-

```
void showAvailableSlots() {
    for (int i = 0; i < MAX_SLOTS; i++) {
        if (parking[i].isParked == 0) {
            printf("Slot %d is empty\n", i + 1);
        }
    }
}
```

5. Parking a Vehicle:-

```
void parkVehicle() {
    int slotFound = -1;

    for (int i = 0; i < MAX_SLOTS; i++) {
        if (parking[i].isParked == 0) {
            slotFound = i;
            break;    }    }
```

```
   struct Vehicle v;
    printf("Enter Vehicle Number: ");
    scanf("%s", v.number);

    printf("Enter Vehicle Type: ");
    scanf("%s", v.type);

    printf("Enter No. of Hours: ");
    scanf("%d", &v.hours);

v.charge = calculateCharge(v.type, v.hours);
    v.slot = slotFound + 1;
    v.isParked = 1;
    parking[slotFound] = v;
```

6. Removing a Vehicle:-

```
void removeVehicle() {
    char num[15];
    printf("Enter Vehicle Number to remove: ");
    scanf("%s", num);

    for (int i = 0; i < MAX_SLOTS; i++) {
        if (parking[i].isParked == 1 &&
strcmp(parking[i].number, num) == 0) {
            printf("Vehicle %s removed from slot %d\n",
parking[i].number, parking[i].slot);
            printf("Total Charge: %.2f\n",
parking[i].charge);
            parking[i].isParked = 0;
```

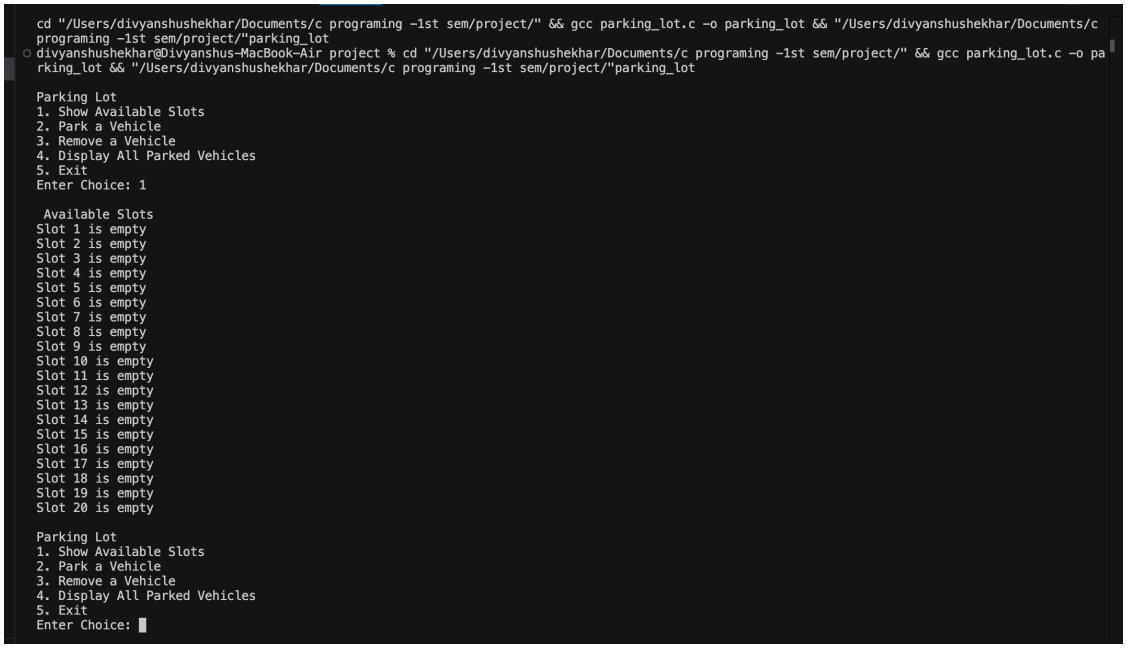
```
  return;
        }
    }

    printf("Vehicle not found.\n");
}
```

7. Displaying All Parked Vehicles:-

```
void displayParked() {
    for (int i = 0; i < MAX_SLOTS; i++) {
        if (parking[i].isParked == 1) {
            printf("Slot %d | Number: %s | Type: %s |
Hours: %d | Charge: %.2f\n",
                parking[i].slot, parking[i].number,
parking[i].type,
                parking[i].hours, parking[i].charge);
        }
    }
}
```

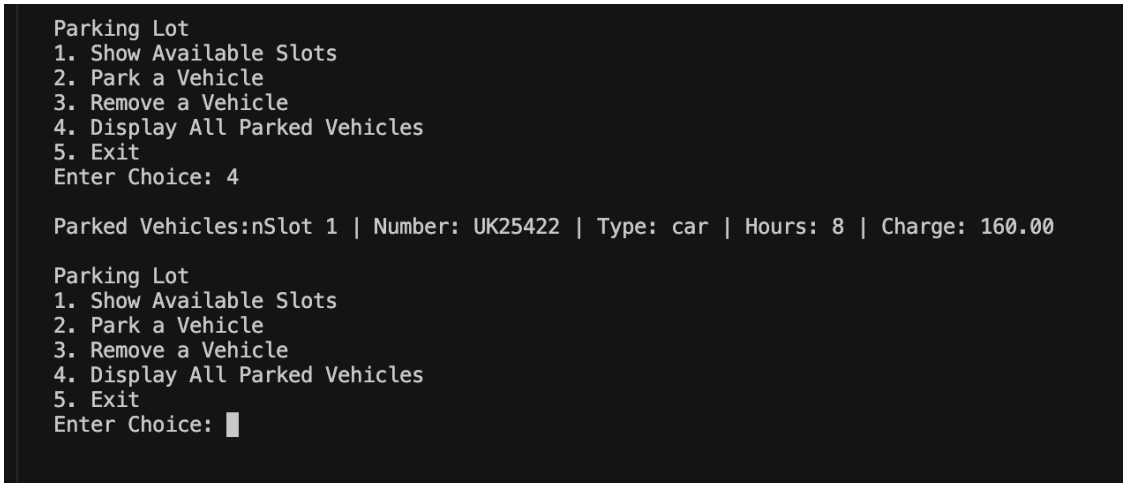
Testing & Results:-

Test case 1: Show available slots:

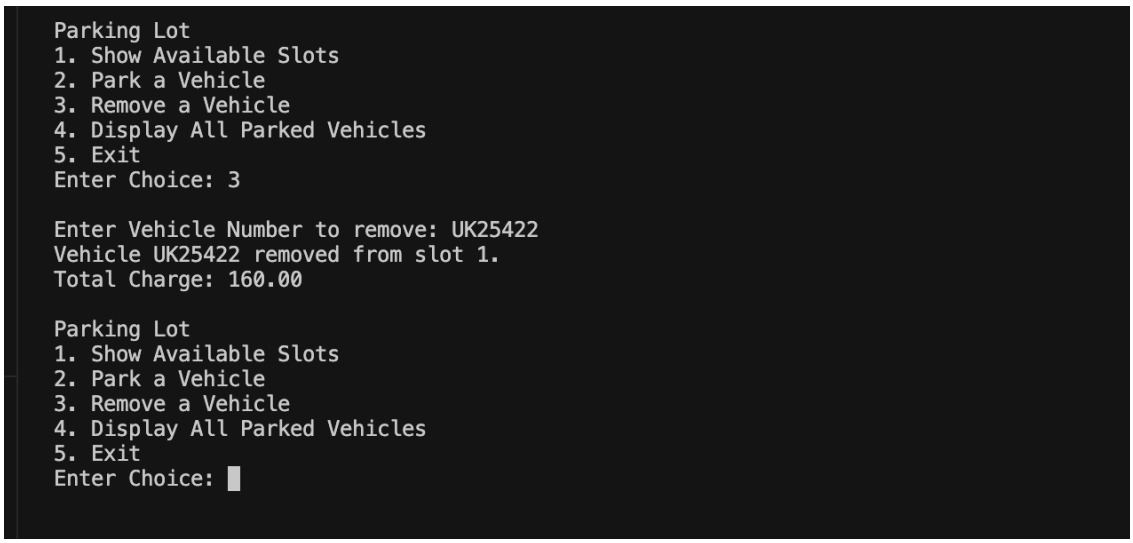


Test Case 2: Park Vehicle:

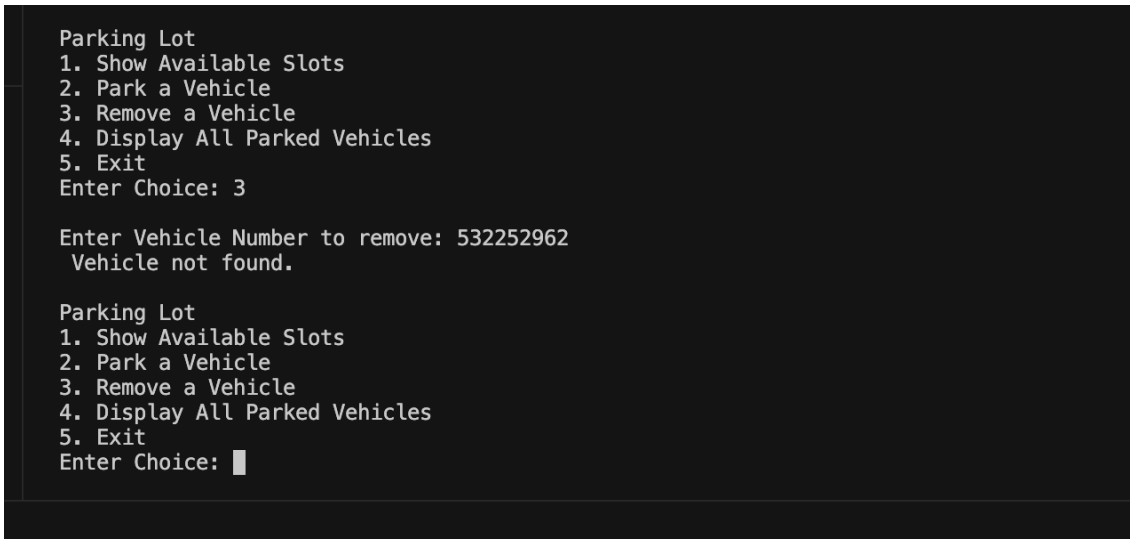
Test Case 3: Display all parked vehicle:



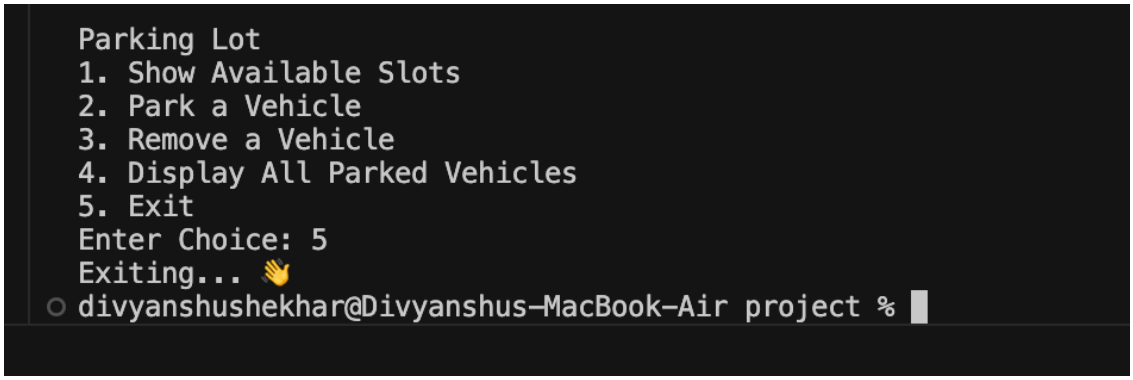
Test Case 4: Remove Vehicle:



Test Case 5: Remove a Vehicle That Does Not Exist:



Test Case 6: Parking Full Condition:



Conclusion & Future Work:-

Conclusion:

In this project I created a simple Vehicle Parking Management System using the C programming language. The system helps manage parking slots by allowing users to park vehicles, remove them, check empty slots, and see all parked vehicles easily. While working on this project, I learned how to use structures, functions arrays, and conditional statements in a real-life type problem.

Overall, the system works well for basic parking management. It reduces manual work and makes the process more organized. This project also helped me understand how a small program can solve practical problems in everyday life.

Future Works:

- Adding password/login so only authorized people can access the system.
- Storing data in a file so that vehicle records are not lost when the program closes.
- Adding a search option to quickly find any parked vehicle.
- Showing total daily earnings from all vehicles.

References:-

- Classnotes
- YouTube