INHA UNIVERSITY IN TASHKENT
School of Computer and Information Engineering
FALL SEMESTER 2025

Project- (Students Learning)

DATA STRUCTURES

M.M: 20 Marks Submission Date: **3 DECEMBER 2025**

Parking Lot Management System (Stacks + Queues)

Problem Description

In many real parking lots, cars are parked one behind another in a narrow lane. If the car at the front wants to leave, all cars behind it need to be moved out temporarily. In this project, you will simulate **such a parking area** using data structures. Your program must allow cars to be parked, removed, or viewed, and must show how many movements were made when a blocked car exits.

Example: How the Parking Lot System Works

Imagine there is **one narrow parking lane**. Cars enter from one side and park **one behind another**, like a **stack**. Suppose the parking lane currently has these cars (front to back):

Top of stack → Car 40, Car 30, Car 20, Car 10 → Bottom

So the parking order is:

1. **Car 10** (came first, at the very back)
2. **Car 20**
3. **Car 30**
4. **Car 40** (last car, near the exit)

Core Data Structures

A. Stack → Parking Lane

- Each lane functions like a stack:
  Last Car In → First Car Out
- Cars are pushed when they enter
- Cars are popped when leaving
- If a car in the middle wants to leave, all cars above it must be popped temporarily.

B. Queue → Temporary Holding Lane

- When rearranging cars, the removed cars are stored in a **queue** so that:

  The order in which you take them out is the order in which you put them back.

- This simulates real-life movement.

C. Array → List of Parking Slots or Lanes

- Each element of the array represents a lane (stack).
- Useful if the extension of multiple lanes is implemented.

Features

Park a Car

- The user enters car number or car ID.
- If the lane is not full, the car is **pushed** into the stack.
- If full, show *"Parking Full"*.

2. Remove a Specific Car

This is the main logic of the project.

Steps:

1. Identify the lane in which the car is parked.
2. Keep popping cars from the stack until you find the target car.
3. Each removed car is added to a **queue** (temporary lane).
4. Remove the target car.
5. Move cars from the queue back into the stack (in the same order).

Your program must display:

- How many moves were made
- Which cars were moved

  Example:

  Top → 40, 30, 20, 10 → Bottom

Car **20** wants to exit.

Steps:

1. Move 40 → queue
2. Move 30 → queue
3. Remove car 20
4. Move 30 back
5. Move 40 back

Total movements = 5

This simulates the real-world scenario of removing a blocked car.

Display All Parked Cars

- Show all car IDs in each lane.
- Display top and bottom of each lane.

4. Show Number of Moves for a Car Exit

- Each pop or movement counts as a “move”.
- When a car exits, show:

    Cars moved temporarily

    Total moves required

Extensions (Optional But Recommended)

A. Multiple Lanes with User-Defined Capacity

- Use an array of stacks.
- Example:

    Lane 1 → capacity 5

    Lane 2 → capacity 4

- Park cars in the first lane with available space.

B. Add Pricing Based on Hours

- When parking, record:

    Car number

    Entry time

- When leaving:

    Ask for exit time

    Calculate hours parked and cost
    Example:

```
Rate = 20 units per hour
```