Quiz Data Structure
Week 7 – PPTI 20

Tuesday, 25 February 2025, 09.45 – 11.45 WIB
Odd Student Number

The quiz is a closed book. You only allow access to Ms. Word, Ms. Excel, Calculator and IDE.

- Application of Stack and Expression Tree
  1. Given this notation:

     A × ((B + C) × (D - E)) / (F + G)

     a. [10 points] Convert that Infix to Prefix notation using stack algorithm
     b. [5 points] Create the expression tree of that notation and print the the Inorder, Preorder and Postorder of that tree

  2. Given this expression:

     + * 5 4 / * 6 - 10 2 + 3 1

     a. [10 points] Evaluate that Prefix Notation using stack algorithm so the calculation result can be achieved
     b. [5 points] Create the expression tree of that notation and print the the Inorder, Preorder and Postorder of that tree

- Application of Linked List and Queue
  3. [20 points] You are a flight engineer stationed at Orion Alpha, the largest space station orbiting Mars. Orion Alpha serves as a crucial refueling and repair hub for interstellar missions. However, due to a sudden solar storm alert, all inbound spacecraft are rushing toward the station, requesting emergency docking. However with limited docking bays, you must prioritize which spacecraft can land first using a sophisticated priority queue system. But time is running out—if you don't act fast, some spacecraft may run out of fuel, while others in critical condition might not make it at all!

     Docking Priority Rules
     - Emergency landings must be given the highest priority. If a ship sends a distress signal, it should dock immediately.
     - If there's no emergency, prioritize ships based on remaining fuel levels—ships with less fuel should dock first.
     - If two ships have the same fuel level, dock them in the order they arrived.

     Input:

     a. The first input will be:

        X

        Where:
        - **X** is the number of spacecraft requesting to dock.

     b. The second input will be:

        enqueue("SpacecraftName", FuelLevel, EmergencyStatus)

Where:

- SpacecraftName: The unique identifier of the ship.
- FuelLevel: The amount of remaining fuel (lower fuel = higher priority).
- EmergencyStatus: 1 if the ship is in emergency mode, 0 otherwise.

c. The third input will be:

dequeue

This command processes **one docking** and removes the **highest priority** spacecraft from the queue.

Output:

- Print the order in which ships **should dock** before any dequeuing.
- Print the order **after** one ship docks.

Your mission is to **implement a priority queue in C** using a **single linked list** to manage the spacecraft docking sequence. Ensure all **emergency landings are handled first** and optimize docking efficiency based on **fuel levels and arrival order**.

Example:

Sample Input	Expected Output
5 enqueue("Apollo", 75, 0) enqueue("Voyager", 50, 0) enqueue("Odyssey", 90, 1) enqueue("Endeavour", 15, 0) enqueue("Pioneer", 80, 1) dequeue	Docking Order: Pioneer -> Odyssey -> Endeavour -> Voyager -> Apollo -> NULL Docking Order After Dequeue: Odyssey -> Endeavour -> Voyager -> Apollo -> NULL

Explanation:

- Pioneer and Odyssey have emergency status, so they dock first.
- Among them, Pioneer arrived earlier, so it docks first.
- Next, Endeavour has the lowest fuel (15), so it docks next.
- Voyager (50 fuel) docks before Apollo (75 fuel).

Note: Assume all the inputs are correct, so you did not require to make any error handling

- Application of Hash Tables

4. [50 points] In Metropolitan City X, parking has become a major challenge due to increasing vehicle numbers and limited parking spaces. The City Council has observed several issues with existing parking lots, which lead to inefficiencies and user frustration. Some of the most pressing problems including Drivers struggle to find available parking spaces, Parking lots operate manually, and the same vehicle might be registered multiple times.

   To address these issues, the City Council is recruiting you to develop an automated **Smart Parking Lot Management System**. The goal is to digitally track vehicles and efficiently handle data storage using **hash tables**. The Smart Parking System will:

- Digitally track all vehicles entering and leaving.
- Store data efficiently using **hash tables (size 100)**.
- Assign a **unique parking code** to each vehicle.
- Use hashing techniques (**Mid-Square & Division**) to ensure fast lookups.
- Handle parking collisions (when two cars get the same hash index) use **chaining techniques**.

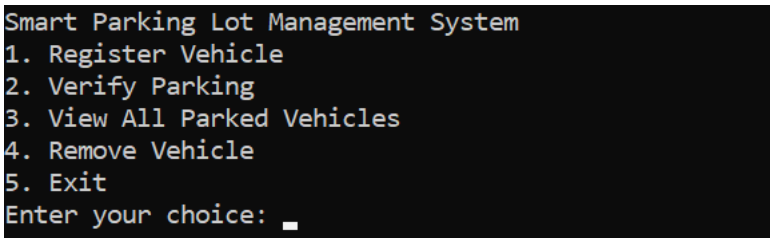

Figure 1 Main Menu

Menu Detail:

1. Register Vehicle
    - The driver enters their license plate number and name.
    - The system calculates a hash index using Mid-Square & Division hashing.
    - A unique parking code is generated for the driver. The generated code are 5 random alphanumeric characters (a-z, A-Z, 0-9)
    - Example:
      Enter license plate number: B1234XYZ
      ASCII Sum: 535
      Mid-Square: 535^2 = 286225 → Extracted: 62
      Index: 62
      Generated Parking Code: p1KaLm

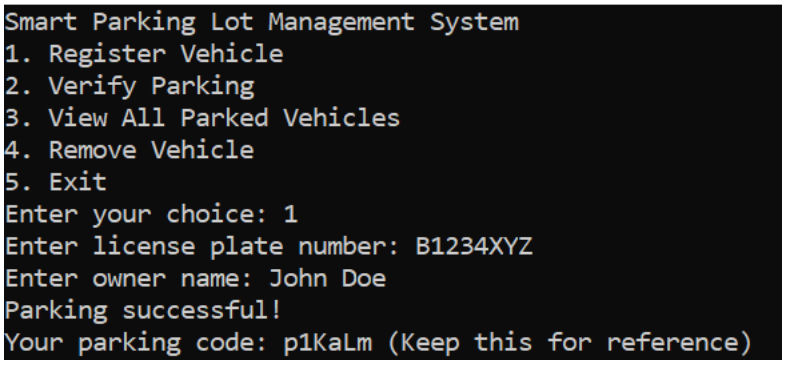

Figure 2 User Input for Register Vehicle and Display Generated Parking Code

2. Verify Parking
    - The driver **enters their unique parking code** to retrieve their details.

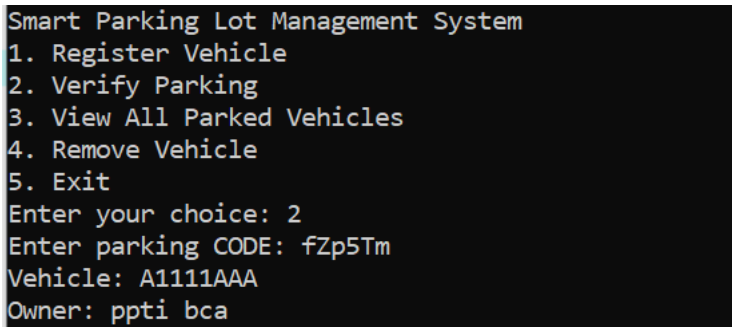

Figure 3 Successful Verification - Showing the Vehicle Details

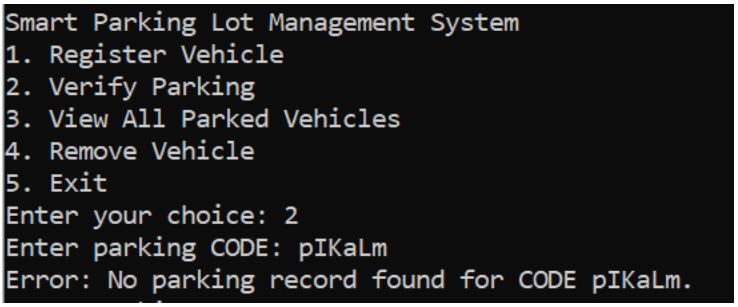

Figure 4 Unsuccessful Verification - The Parking Code does not exist

3. View All Parked Vehicles
    - The system **displays all vehicles** sorted by **hash index**.

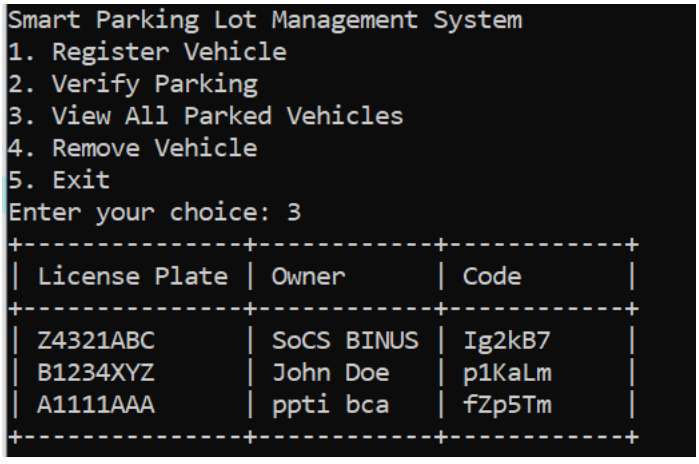

Figure 5 Show All Vehicle Data

4. Remove Vehicle
    - The driver enters their **license plate number** to **exit the parking lot**.

```
Smart Parking Lot Management System
1. Register Vehicle
2. Verify Parking
3. View All Parked Vehicles
4. Remove Vehicle
5. Exit
Enter your choice: 4
Enter license plate number to remove: B1234XYZ
Vehicle B1234XYZ has exited.
```

Figure 6 Successful Deleting the Vehicle Data

```
Smart Parking Lot Management System
1. Register Vehicle
2. Verify Parking
3. View All Parked Vehicles
4. Remove Vehicle
5. Exit
Enter your choice: 4
Enter license plate number to remove: p1KaLm
Error: No parking record found for vehicle p1KaLm.
```

Figure 7 Unsuccessful Deleting Data that doesn't Exist

5. Exit
    - Exit the program

```
Smart Parking Lot Management System
1. Register Vehicle
2. Verify Parking
3. View All Parked Vehicles
4. Remove Vehicle
5. Exit
Enter your choice: 5
Exiting program...
```

Figure 8 Exit