5th Assignment

Data Structure and Algorithm (Java)

Deadline: 27th June 2025 **Marks: 40**

Scenario 1:

You are developing a student ID management system using an AVL Tree. Each student has a unique ID (an integer), and you want to maintain the IDs in a balanced binary search tree to allow efficient insertion, deletion, and search operations.

a) Insert the following student IDs into an initially empty AVL tree:
40, 20, 10, 25, 30, 22, 50, 60
   i. Draw the AVL tree **after each insertion**.
   ii. Clearly indicate:
      a. At which step(s) the tree becomes unbalanced.
      b. What **rotation(s)** are applied (LL, RR, LR, RL).
      c. The final balanced tree.

b) Now delete the following student IDs from your tree:

30, 40

After each deletion:

a. Show the structure of the tree.
b. Identify if any node becomes unbalanced.
c. Apply necessary rotations to restore AVL balance.

c) Perform search operations on the AVL tree you obtained **after Part B**:

   i. Search for the following IDs: 25, 40, 60
   ii. For each search:
   Indicate whether the ID is **found or not**.
   Show the **path** taken from the root to the node (e.g., "Start at 25 → go right to 50 → go left to 30").

Scenario 2:

You are managing a dynamic AVL Tree to store unique Book IDs in a digital library system. These IDs must be kept balanced for efficient retrieval.

a) Insert the following Book IDs into an empty AVL tree:

45, 20, 60, 10, 30, 25, 35, 50, 70, 65, 80, 55

i. After each insertion:
   - Draw the tree (or provide a trace of the structure).
   - Identify the balance factor of each node.
   - Indicate what type of rotation occurs (LL, RR, LR, RL).

ii. Final task: Show the final balanced AVL tree.

b) Now, delete the following Book IDs from the AVL tree:

10, 30, 45

After each deletion:

i. Identify where the imbalance occurs.
ii. Apply appropriate rotation(s) to fix it.
iii. Show the tree structure after each rebalancing.
iv. Track how deletions can propagate imbalance upward.

c) Search for the following Book IDs in your final tree after deletions:

25, 45, 55, 100

For each:

i. State whether the ID is found or not found.
ii. Provide the search path (e.g., "Start at 50 → left to 25").

Scenario 3:

You are building a smart parking system for a shopping mall. Each car is assigned a unique license plate number (as an integer key). These license numbers must be stored in a hash table to track parked vehicles quickly and efficiently.

The mall has 10 parking slots.
You will use a hash table of size 10, with the following hash function:

h(license_plate) = license_plate % 10

a) **Linear Probing**

You decide to first implement linear probing to resolve collisions.

Insert the following license plate numbers in the given order:
1001, 1011, 1021, 1002, 1003

Tasks:

1. Show the hash table after each insertion.
2. Indicate when and where collisions occur.
3. What slot does 1003 end up in?
4. Search for license plate 1021. Show the probe path.

b) **Quadratic Probing**

Switch to quadratic probing.
Insert the same license plate numbers:
1001, 1011, 1021, 1002, 1003

Use quadratic probing formula:
$(h + i^2) \% 10$

Tasks:

1. Show the table after insertions.
2. Where does 1003 end up now?
3. Compare the number of probes needed for 1021 in linear vs quadratic probing.

c) **Separate Chaining**

Now implement separate chaining.

Insert the following license plate numbers:
1001, 1011, 1111, 1211, 1311

Tasks:

1. Show the hash table as an array of linked lists.
2. Indicate which slot contains multiple cars.
3. Remove 1211. Show the updated structure.
4. Search for 1311 and show the path.

Scenario 4:

You are designing a campus navigation system for a university. The campus consists of various buildings connected by walking paths. Each building is represented as a node, and each path as an edge in an undirected graph.

You are given the following connections:

Buildings: A, B, C, D, E, F, G
Paths (edges):
A - B, A - C, B - D, B - E, C - F, E - G
The goal is to help a student find routes between buildings using BFS and DFS.

a) Represent the Graph
   1. Represent the graph using:
      - An adjacency list
      - An adjacency matrix

b) BFS Traversal
   1. Perform a Breadth-First Search (BFS) starting from building A.
   2. List the order in which buildings are visited.

c) DFS Traversal
   1. Perform a Depth-First Search (DFS) starting from building A.
   2. List the order of visited buildings.

---End of Assignment! ---