W3 - Data Structure & Algorithms

Problem 1: Remove a Loop from a Singly Linked List

Overview: Detect and remove a loop in a singly linked list.

Algorithm Steps:

1. **Detect Loop**:
    - Use Floyd's Cycle detection algorithm (Tortoise and Hare approach) where two pointers move through the list at different speeds.
    - If there's a loop, they will eventually meet inside the loop.
2. **Find Loop Start**:
    - Once a loop is detected, initialize one pointer to the start of the list and keep the other at the meeting point.
    - Move both pointers one step at a time. The point where they meet again is the start of the loop.
3. **Remove Loop**:
    - Keep one pointer fixed at the start of the loop and move the other around the loop until it reaches the node just before the start of the loop.
    - Set the next pointer of this node to null to remove the loop.
4. **Code**

```
class Node {
    int data;
    Node next;

    Node(int d) {
        data = d;
        next = null;
    }
}
public class LinkedList {
    Node head; // head of the list

    // Function to detect and remove loop in a linked list
    public void removeLoop() {
        Node slow = head, fast = head;
        boolean loopExists = false;

        // Detect loop
        while (fast != null && fast.next != null) {
```

```
            slow = slow.next;
            fast = fast.next.next;
            if (slow == fast) {
                loopExists = true;
                break;
            }
        }

        // Remove loop if exists
        if (loopExists) {
            slow = head;
            while (slow.next != fast.next) {
                slow = slow.next;
                fast = fast.next;
            }
            fast.next = null; // Remove loop
        }
    }
}
```

Problem 2: Implement Circular Linked List for Josephus Problem

Overview: Implement a circular linked list and use it to solve the Josephus problem, where people are arranged in a circle and eliminated every kth person until one remains.

Algorithm Steps:

1. **Build Circular Linked List**:
    - Create a circular linked list with nodes representing each person in the problem.
2. **Josephus Solution**:
    - Start from the first person and iterate through the list, counting up to k.
    - Eliminate the kth person by removing the node from the list and close the gap by linking the previous node to the next node.
    - Continue the process until only one node remains in the list.
    - The remaining node represents the position that should be taken to avoid elimination.

```
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
```

```
    }
}

class CircularLinkedList {
    Node head = null;
    Node tail = null;

    // Add new node to the list
    public void add(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            tail.next = newNode;
        }
        tail = newNode;
        tail.next = head;
    }

    // Solve Josephus problem
    public int solveJosephus(int k) {
        Node curr = head;
        Node prev = tail;

        while (curr != prev) {
            int count = 1;
            while (count != k) {
                prev = curr;
                curr = curr.next;
                count++;
            }
            prev.next = curr.next; // Remove k-th node
            curr = prev.next;
        }
        return curr.data;
    }
}
```

Problem 3: Queue Simulation for ATM

↳ 1 cell hidden

Problem 4: Check for Balanced Parentheses, Brackets, and Curly Braces

↳ 1 cell hidden