Simulating Election Algorithms – Bully and Ring Algorithm

Objective

- To understand and simulate leader election algorithms in distributed systems.
- To implement the **Bully Algorithm** and **Ring Election Algorithm** using Java or Python.
- To observe how coordination and fault tolerance are achieved via elections.

Theory Summary

1. Bully Algorithm

- Used when any process can initiate the election.
- The process with the highest ID wins and becomes the coordinator.
- If a process detects that the coordinator has failed, it initiates an election.
- Higher ID processes respond with their own election message.

2. Ring Election Algorithm

- All processes are arranged in a logical ring.
- An election message is passed around containing the IDs of the candidates.
- The process with the highest ID becomes the new coordinator.

Lab Questions

Part A: Bully Algorithm

1. Simulate a system with N processes, each with a unique process ID.

2. Let any process detect the failure of the coordinator and initiate an election.
3. Implement message passing between processes to identify the new leader.
4. Simulate multiple processes starting the election at the same time. What happens?
5. Simulate recovery of a failed coordinator. Should it bully the others?

Code Hint (Java-style pseudocode)

```
class Process {
    int id;
    boolean isAlive;

    void startElection(List<Process> processes) {
        for (Process p : processes) {
            if (p.id > this.id && p.isAlive) {
                System.out.println("Message sent to Process " + p.id);
                // Higher process responds and starts its own election
            }
        }
        // If no higher process responds, declare self as coordinator
    }
}
```

Part B: Ring Election Algorithm

1. Arrange processes in a ring (circular array).
2. Each process can initiate an election and pass a token with the highest ID seen so far.
3. The election continues until the token comes back to the initiator.
4. The process with the highest ID becomes the new coordinator.

Code Hint (Java-style pseudocode)

```
class RingProcess {
    int id;
    boolean isActive;
    RingProcess next;

    void initiateElection() {
        List<Integer> electionMessage = new ArrayList<>();
        electionMessage.add(this.id);
        passMessage(electionMessage);
    }

    void passMessage(List<Integer> message) {
        message.add(this.id);
        if (next != null) next.passMessage(message);
        else {
            int leader = Collections.max(message);
            announceLeader(leader);
        }
    }

    void announceLeader(int leaderId) {
        System.out.println("New coordinator is: " + leaderId);
    }
}
```

Expected Outcomes

- Students should be able to simulate distributed leader election.
- They should understand differences in how **message passing** and **failure recovery** work in Bully vs. Ring.
- Should demonstrate understanding of process coordination and distributed control.