Assignment No 5

Title: Implement Token Ring-Based Mutual Exclusion Algorithm

Objectives

- To understand the **token-based mutual exclusion** mechanism in distributed systems.
- To simulate the **Token Ring Algorithm** used for ensuring mutual exclusion in a distributed environment.
- To implement **safe critical section access** by multiple processes using token passing.
- To visualize the working of distributed process coordination and synchronization.

Problem Statement

In a distributed system, multiple processes may need exclusive access to a shared resource (critical section). Implement a simulation of the **Token Ring-Based Mutual Exclusion Algorithm**, where a logical ring is formed among `n` processes, and a **unique token** is passed between them. Only the process holding the token can enter the critical section. Demonstrate the process of:

- Token generation
- Token passing
- Controlled access to the critical section

Each process must enter and exit the critical section safely without causing race conditions.

Expected Outcomes

- Understanding of **distributed synchronization** and mutual exclusion.
- A working simulation of a **token ring** involving `n` processes.
- Clear demonstration of:
  - Token holding process entering the critical section
  - Token passing to the next process in the ring
- Verification that no two processes access the critical section simultaneously.

Software Requirements

Component	Specification
Operating System:	Windows / Linux / macOS
Programming Language:	C / C++ / Java / Python
IDE or Editor:	VS Code / Eclipse / IntelliJ / Terminal
Additional Tools:	(Optional) Threads or Process libraries (e.g., `threading` in Python, POSIX threads in C)

Hardware Requirements

Component	Specification
Processor	Multi-core CPU
RAM	Minimum 2 GB
Storage	50 MB or more

Theory: Token Ring-Based Mutual Exclusion Algorithm

What is Mutual Exclusion?

Mutual exclusion ensures that **only one process** can access a **shared resource or critical section (CS)** at any given time. In distributed systems, enforcing mutual exclusion is challenging because there's **no shared memory or clock**, and processes communicate via messages.

What is the Token Ring Algorithm?

The **Token Ring Algorithm** is a **distributed mutual exclusion algorithm** in which:

- `n` processes are arranged in a **logical ring** (each process knows its successor).
- A unique **token** (a control message) circulates around the ring.
- A process must **hold the token** to enter the critical section.
- After exiting the CS, the process **passes the token** to its neighbor.

This ensures:

- **No starvation** (every process eventually gets the token).
- **Fairness** (token moves in a fixed order).
- **Mutual exclusion** (only one token, hence one process in CS at a time).

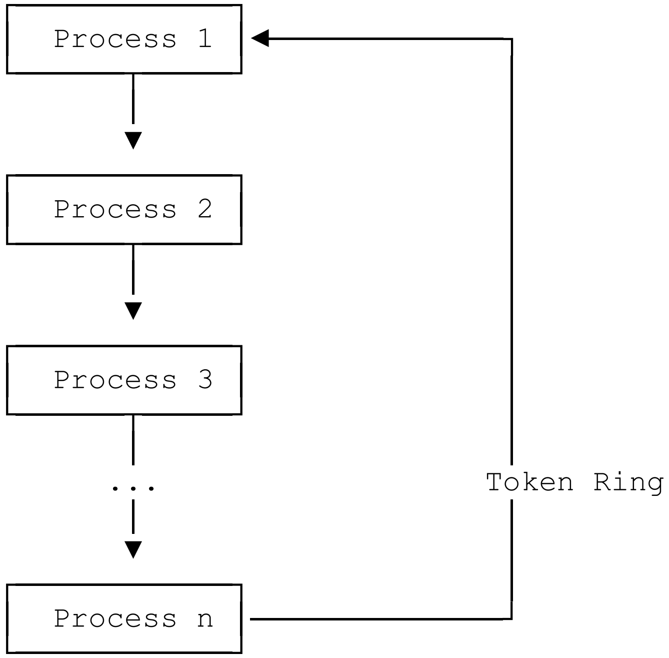
How It Works

1. **Initialization**:
   - One process starts with the token (token creation).
   - Other processes wait for the token.
2. **Request to Enter CS**:
   - A process that wants to enter the CS waits until it receives the token.
3. **Critical Section Execution**:
   - Once it has the token, the process enters the CS.
   - After completing the CS, it passes the token to the next process.
4. **Token Passing**:
   - If a process does not need the CS, it simply passes the token to its successor.
5. **Repeat**:
   - The cycle continues, ensuring every process gets a turn.

Benefits

- Simple to implement.
- Guarantees **mutual exclusion**, **no starvation**, and **bounded waiting**.
- Minimal message complexity: only **one message per critical section entry**.

Architecture Diagram: Token Ring Algorithm



- Each process holds the token in turn
- Only token holder can enter critical section
- Token passed in logical ring order

Conclusion: