Page No.
Date / /

# Assignment No.-05

- **Aim :-** Implement Token Ring based Mutual Exclusion algo.

- **Tools and Environment :-** Java Runtime Environment, Java JDK.

- **Theory :-**

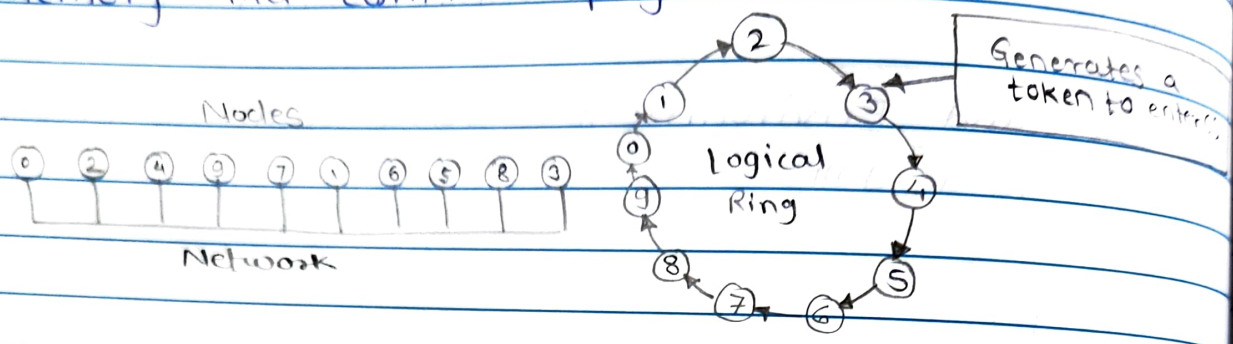
Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process can not enter its critical section while another concurrent process is currently present or executing in its critical section i.e only one process is allowed to execute the critical section at any given instance of time.

Mutual exclusion in single computer system Vs. distributed system.

In single computer system, memory & other resources are shared between different processes. The status of shared resources & the status of users is easily available in the shared memory so with the help of shared variable (for e.g :- semaphores) mutual exclusion problem can be easily solved.

In Distributed systems, we neither have shared memory nor a common physical clock & there for we can not solve mutual exclusion problem using shared variables. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used.

A site in distributed system do not have complete info. of state of the system due to lack of shared memory and common physical clock.




- Requirements of Mutual exclusion Algorithm:-
- No Deadlock:-

Two or more sites should not endlessly wait for any message that will never arrive.

- No Starvation:-

Every site who wants to execute critical section should get an opportunity to execute it in finite time. Any site should not wait indefinitely to execute critical section while other site are repeatedly executing critical section.

Fairness:-

Each site should get a fair chance to execute critical section. Any request to execute critical section must be executed in the order they are made i.e Critical section execution requests should be executed in the order of their arrival in the system.

Page No.
Date / /

- **Fault Tolerance :-**

In case of failure, it should be able to recognize it, by itself in order to continue functioning without any disruption.

- **Solution to distributed mutual exclusion :-**

As we know shared variables or a local kernel can not be used to implement mutual exclusion in distributed systems. Message passing is a way to implement mutual exclusion. Below are the three approaches based on message passing to implement mutual exclusion in distributed systems.

- **Token Based Algorithm :-**

A unique token is shared among all the sites. If a site processes the unique token, it is allowed to enter its critical section.

This approach uses sequence number to order requests for the critical section.

Each requests for critical section contains a sequence number. This sequence number is used to distinguish old and current requests.

This approach insures mutual exclusion as the token is unique.

- **Example :-**

Suzuki-Kasami's Broadcast Algorithm.

- **Conclusion :-**

Hence Token Ring algorithm achieves mutual exclusion in a distributed system by creating a bus network of processes

NAME: PALLAVI K. CHOPADE
ROLL NO.: 14
PRN NO.: 72036169K
BE (IT)
LP - V

```
#Token.py

import threading
import time

class TokenRing:
    def __init__(self, numProcesses):
        self.num_processes = numProcesses
        self.threads = []
        self.mutex = threading.Semaphore(1)
        self.tokens = [threading.Event() for _ in range(numProcesses)]
        self.current_token = 0

        for i in range(numProcesses):
            t = threading.Thread(target=self.process, args=(i,))
            self.threads.append(t)

    def start(self):
        for thread in self.threads:
            thread.start()

    def process(self, process_id):
        while True:
            self.tokens[process_id].wait()

            self.mutex.acquire()
            print("Process id:", process_id, "is in critical section.")
            time.sleep(2)
            self.mutex.release()

            print("Process id:", process_id, "is released")

            next_process_id = (process_id + 1) % self.num_processes
            self.tokens[next_process_id].set()

            self.tokens[process_id].clear()

    def initialize_token_ring(self):
        self.tokens[0].set()

if __name__ == "__main__":
    num_processes = 4
    tokenRing = TokenRing(num_processes)
    tokenRing.start()
    tokenRing.initialize_token_ring()
```

# Assignment No.- 05

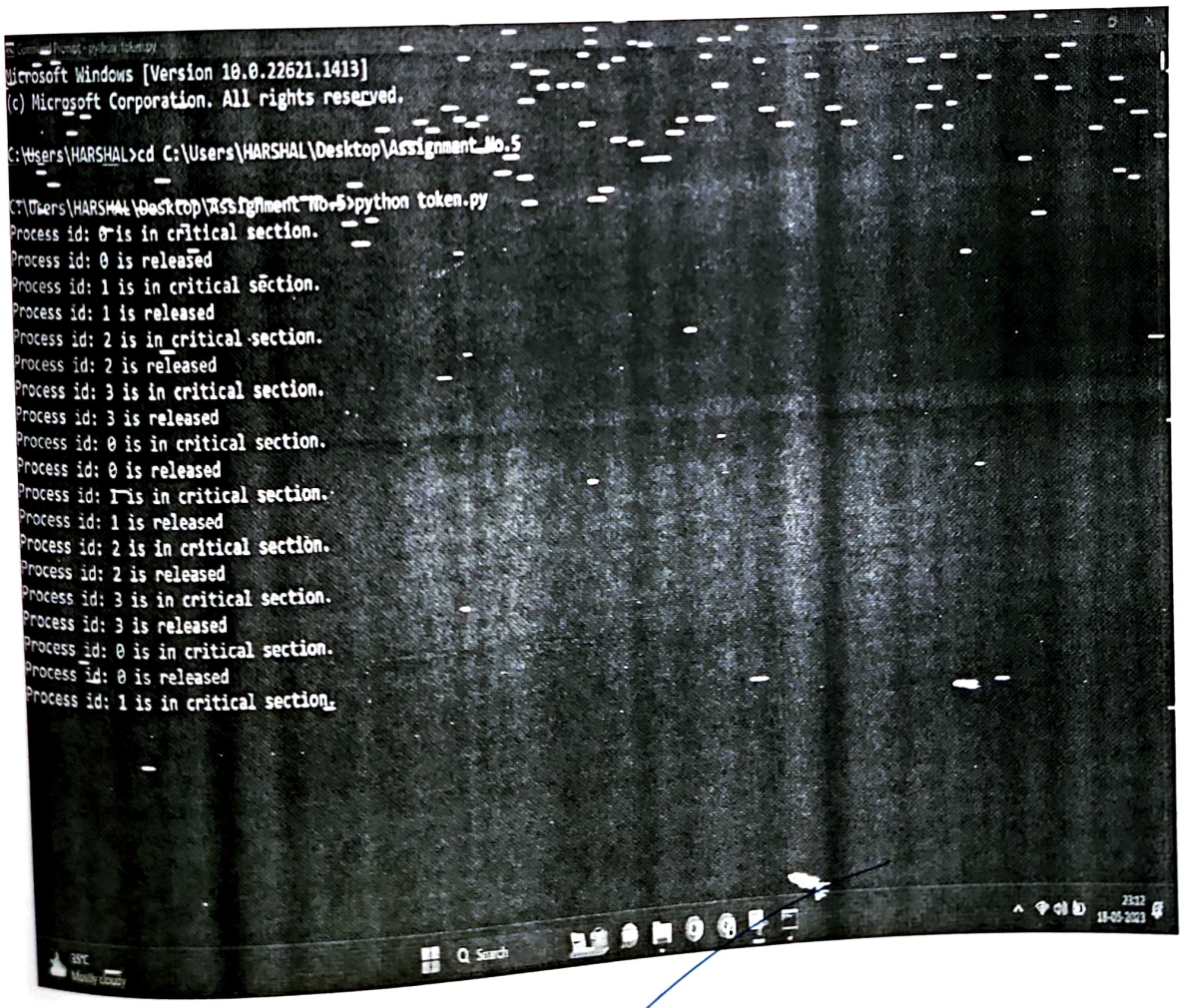
Name: - Pallavi Kamlakar Chopade.

Roll No.: - 14

PRN No.: - 72036169K

Subject: - Distributed Systems

Class: - BE(IT)



```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HARSHAL>cd C:\Users\HARSHAL\Desktop\Assignment_No.5

C:\Users\HARSHAL\Desktop\Assignment_No.5>python token.py
Process id: 0 is in critical section.
Process id: 0 is released
Process id: 1 is in critical section.
Process id: 1 is released
Process id: 2 is in critical section.
Process id: 2 is released
Process id: 3 is in critical section.
Process id: 3 is released
Process id: 0 is in critical section.
Process id: 0 is released
Process id: 1 is in critical section.
Process id: 1 is released
Process id: 2 is in critical section.
Process id: 2 is released
Process id: 3 is in critical section.
Process id: 3 is released
Process id: 0 is in critical section.
Process id: 0 is released
Process id: 1 is in critical section.
```