St. Francis Institute of Technology
Department of Computer Engineering

Academic Year: 2021-2022 **Semester: VIII**
Subject: Distributed Computing **Class / Branch / Division: BE/CMPN/A**
Name: Rebecca Dias Roll Number:18

Experiment No: 06

Aim : Implementation of Mutual Exclusion Algorithm.

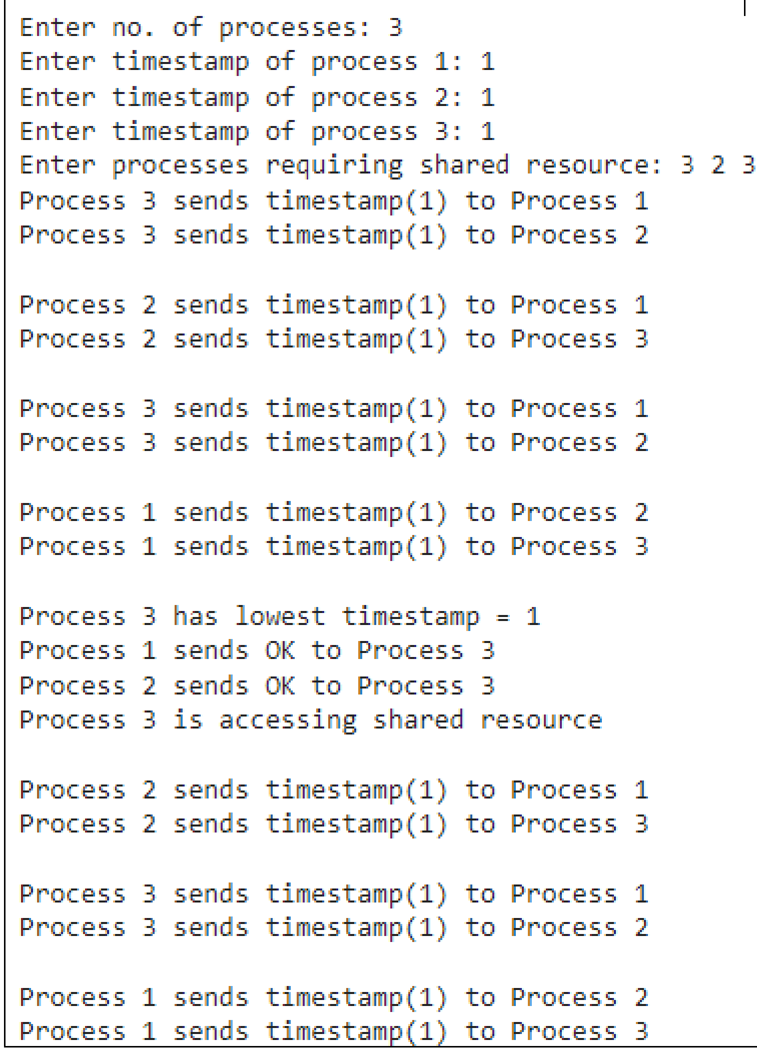
Theory:

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.
Requirements of Mutual exclusion Algorithm:
No Deadlock:
Two or more site 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.
Fault Tolerance:
In case of failure, it should be able to recognize it by itself in order to continue functioning without any disruption.

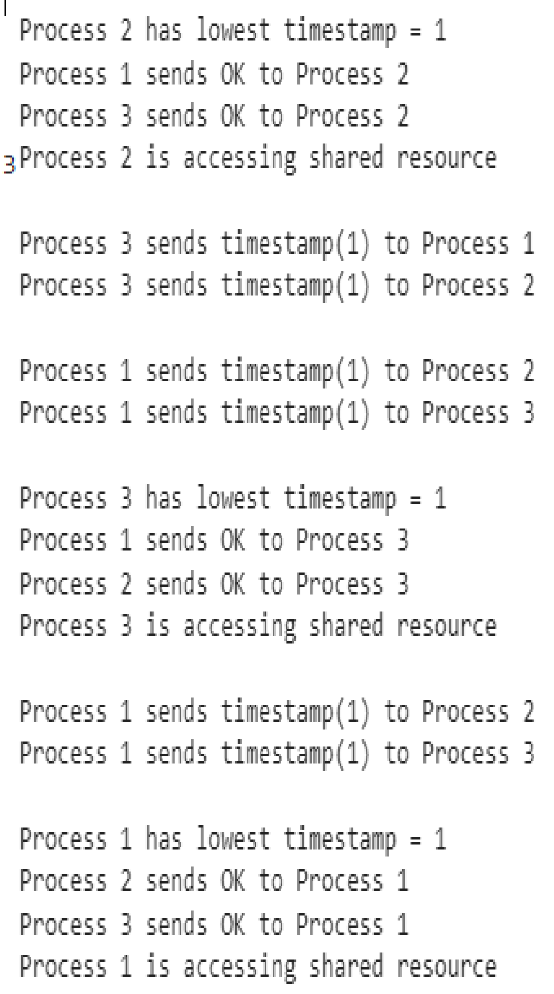
Implelementation:

```
import time
process = {}
n = int(input("Enter no. of processes: "))
for i in range(n):
    process[i + 1] = int(input(f"Enter timestamp of process {i + 1}: "))
p = list(map(int, input("Enter processes requiring shared resource: ").split(" ")))
while p != []:
    selected_time = 99
    selected_process = -1
    for pi in p:
        if process[pi] < selected_time:
            selected_time = process[pi]
            selected_process = pi
        for i in range(n):
            if i + 1 == pi:
                continue
            print(f"Process {pi} sends timestamp({process[pi]}) to Process {i + 1}")
        print()
    print(f"Process {selected_process} has lowest timestamp = {selected_time}")
    for i in range(n):
        if i + 1 == selected_process:
            continue
        print(f"Process {i + 1} sends OK to Process {selected_process}")
    print(f"Process {selected_process} is accessing shared resource\n")
    time.sleep(5)
    p.remove(selected_process)
```

Output:



```
Enter no. of processes: 3
Enter timestamp of process 1: 1
Enter timestamp of process 2: 1
Enter timestamp of process 3: 1
Enter processes requiring shared resource: 3 2 3
Process 3 sends timestamp(1) to Process 1
Process 3 sends timestamp(1) to Process 2

Process 2 sends timestamp(1) to Process 1
Process 2 sends timestamp(1) to Process 3

Process 3 sends timestamp(1) to Process 1
Process 3 sends timestamp(1) to Process 2

Process 1 sends timestamp(1) to Process 2
Process 1 sends timestamp(1) to Process 3

Process 3 has lowest timestamp = 1
Process 1 sends OK to Process 3
Process 2 sends OK to Process 3
Process 3 is accessing shared resource

Process 2 sends timestamp(1) to Process 1
Process 2 sends timestamp(1) to Process 3

Process 3 sends timestamp(1) to Process 1
Process 3 sends timestamp(1) to Process 2

Process 1 sends timestamp(1) to Process 2
Process 1 sends timestamp(1) to Process 3
```



```
Process 2 has lowest timestamp = 1
Process 1 sends OK to Process 2
Process 3 sends OK to Process 2
Process 2 is accessing shared resource

Process 3 sends timestamp(1) to Process 1
Process 3 sends timestamp(1) to Process 2

Process 1 sends timestamp(1) to Process 2
Process 1 sends timestamp(1) to Process 3

Process 3 has lowest timestamp = 1
Process 1 sends OK to Process 3
Process 2 sends OK to Process 3
Process 3 is accessing shared resource

Process 1 sends timestamp(1) to Process 2
Process 1 sends timestamp(1) to Process 3

Process 1 has lowest timestamp = 1
Process 2 sends OK to Process 1
Process 3 sends OK to Process 1
Process 1 is accessing shared resource
```

Conclusion:
In this experiment successfully implemented Ricarts Agrawala Algorithm for Mutual Exclusion.

POST LAB Questions:

1. Compare all the algorithm for mutual exclusion.

Parameters	**Centralized algorithm**	**Distributed algorithm**	**Token Ring algorithm**
Election	One process is,elected as coordinator	Total ordering of all events in the system	Uses token for entering critical section
Messages per entry/exit	Requires three messages to enter and exit a critical region	Requires 2(n-1) messages	Variable number of messages required.
Delay in message times	Delay for messages is 2 messages	Delay for messages is 2(n-1)	The time varies from 0 to n-1 tokens
Mutual exclusion	Guarantees mutual exclusion	Mutual exclusion guaranteed without dead lock	Mutual exclusion is guaranteed
Starvation	No starvation	No starvation	No starvation
Complexity	Easy to implement	Complicated process	Implementation is easy

1. **What are the requirements of Mutual Exclusion Algorithm.**

A. **No Deadlock:**
    a. Two or more site should not endlessly wait for any message that will never arrive.

B. **No Starvation:**
    a. 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

C. **Fairness:**
    a. 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.

2. **Different Type of Mutual Exclusion Algorithm**

1. Token Based Algorithm:
    a. A unique token is shared among all the sites.
    b. If a site possesses the unique token, it is allowed to enter its critical section
    c. This approach uses sequence number to order requests for the critical section.
    d. Each requests for critical section contains a sequence number. This sequence number is used to distinguish old and current requests.
    e. This approach insures Mutual exclusion as the token is unique
    f. Example:
    g. Suzuki-Kasami’s Broadcast Algorithm
2. Non-token based approach:
    a. A site communicates with other sites in order to determine which sites should execute critical section next. This requires exchange of two or more successive round of messages among sites.
    b. This approach use timestamps instead of sequence number to order requests for the critical section.
    c. When ever a site make request for critical section, it gets a timestamp. Timestamp is also used to resolve any conflict between critical section requests.
    d. All algorithm which follows non-token based approach maintains a logical clock. Logical clocks get updated according to Lamport’s scheme
    e. Example:
    f. Lamport's algorithm, Ricart–Agrawala algorithm
3. Quorum based approach:
    a. Instead of requesting permission to execute the critical section from all other sites, Each site requests only a subset of sites which is called a quorum.
    b. Any two subsets of sites or Quorum contains a common site.
    c. This common site is responsible to ensure mutual exclusion
    d. Example:
    e. Maekawa’s Algorithm