1

Page No.
Date

Aim- Write a program to demonstrate Lamport's Algorithm for distributed Mutual Exclusion

Theory-
Lamport's Distributed Mutual Exclusion Algorithm is a permission based algorithm proposed by Lamport's as an illustration of his synchronization scheme for distributed system

- In Lamports algorithm critical section requests are executed in the increasing order of timestam

- 3 types of message
  a) REQUEST - to get permission to enter CS
  b) REPLY - msg to requesting site to get their permission to enter CS.
  c) RELEASE - to all other site upon exiting the CS.

- Every site si keeps a queue to store CS requests ordered by their timestamps

- A timestamps is given to each CS request using Lamport's logical clock.

Teacher's Sign.:

Page No
Date

Algorithm -

a) To enter critical section:

when a site si wants to enter the CS, it sends a request message Request (ts, i) to all othe sites & place request on queue

when a site Sj receives the request message REQUEST (ts, i) from site si, it returns a time~~stop~~ stamped REPLY message to site si & place the request of site Si on queue.

b) To execute the critical section.

A site Si can enter the CS if it has received the message with timestamps larger than front all other sites & its own request is at the top of queue.

c) To release the critical section

when a site Si exists the CS, it removes its own request from the top of its request queue & sends a timestamped RELEASE message to all other sites

when a site Sj receives the timestamped RELEASE message from site Si, it remove of the request of Si from its request queue

Message complexity:

3(N-1) messages per critical section execution

(N-1) requests message

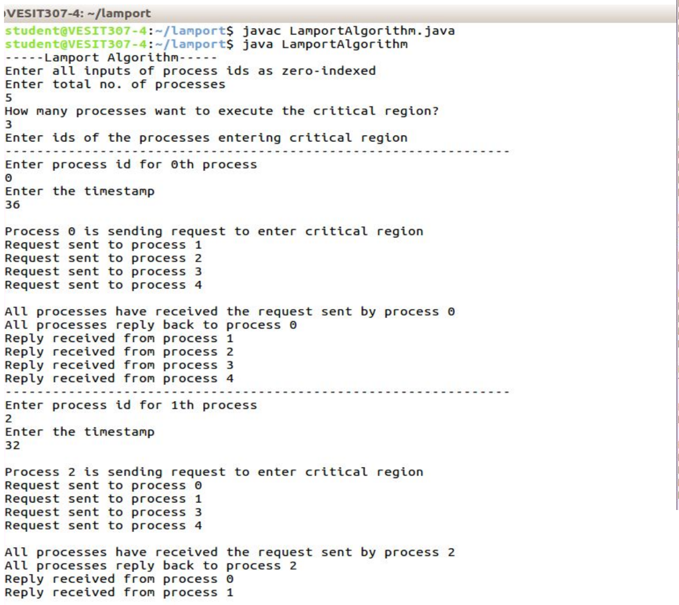
(N-1) reply message

(N-1) release message

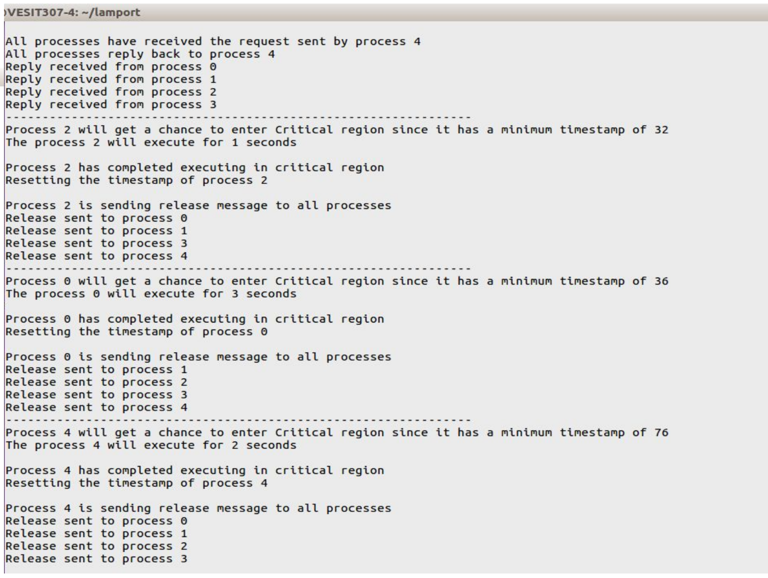
Teacher's Sign.:

LAMPORT'S ALGORITHM

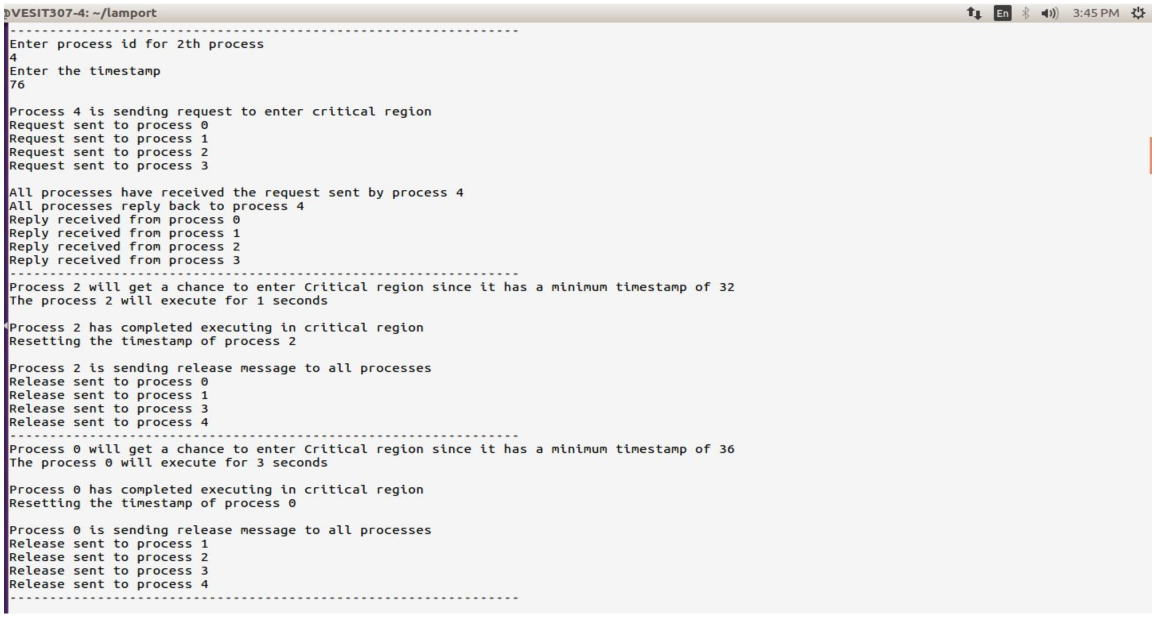
```
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

class LamportAlgorithm{
public static void main(String args[]){

Scanner sc = new Scanner(System.in);
int max = 1000;
System.out.println("-----Lamport Algorithm-----");
System.out.println("Enter all inputs of process
ids as zero-indexed");
System.out.println("Enter total no. of
processes");
int noOfProcesses = sc.nextInt();
int requestArray[][] = new int[noOfProcesses][2];
for (int i=0; i<noOfProcesses; i++) {
requestArray[i][0] = 0;
requestArray[i][1] = max;           }
System.out.println("How many processes want
to execute the critical region?");
int noOfProcessesCriticalRegion = sc.nextInt();

System.out.println("Enter ids of the processes
entering critical region");

for (int i=0; i<noOfProcessesCriticalRegion; i++)
{

System.out.println("--------------------------------");

System.out.println("Enter process id for " + i +
"th process");
int processId = sc.nextInt();

System.out.println("Enter the timestamp");
int timestamp = sc.nextInt();

requestArray[processId][0] = processId;

requestArray[processId][1] = timestamp;

System.out.println();

System.out.println("Process " + processId + " is
sending request to enter critical region");
for (int j=0; j<noOfProcesses; j++) {
if(j != processId){

System.out.println("Request sent to process " +
j);
}}
System.out.println();
System.out.println("All processes have received
the request sent by process " + processId);

System.out.println("All processes reply back to
process " + processId);
for (int j=0; j<noOfProcesses; j++) {if(j !=
processId){
System.out.println("Reply received from process
" + j);}}}
for (int i=0; i<noOfProcessesCriticalRegion; i++)
{
int minTimestamp = max;
int minTimestampIndex = 0;
for (int j=0; j<noOfProcesses; j++) {
if(requestArray[j][1] < minTimestamp){
minTimestamp = requestArray[j][1];
minTimestampIndex = j; }}
System.out.println("----------------------------------");
System.out.println("Process " +
minTimestampIndex + " will get a chance to
enter Critical region since it has a minimum
timestamp of " + minTimestamp);
int executionTime = 1 + (int)(Math.random() * ((5
- 1) + 1));
System.out.println("The process " +
minTimestampIndex + " will execute for " +
executionTime + " seconds");
try{
TimeUnit.SECONDS.sleep(executionTime);}
catch(InterruptedException e){
        System.out.println("Unexpected
interrupt");}
System.out.println();
System.out.println("Process " +
minTimestampIndex + " has completed
executing in critical region");
System.out.println("Resetting the timestamp of
process " + minTimestampIndex);
requestArray[minTimestampIndex][1] = max;
System.out.println();
System.out.println("Process " +
minTimestampIndex + " is sending release
message to all processes");
for (int j=0; j<noOfProcesses; j++) {
if(j != minTimestampIndex){
System.out.println("Release sent to process " +
j);       }}}

                System.out.println("All
requesting processes have completed executing
in the critical region");


        }
}
```



```
VESIT307-4: ~/lamport
student@VESIT307-4:~/lamport$ javac LamportAlgorithm.java
student@VESIT307-4:~/lamport$ java LamportAlgorithm
-----Lamport Algorithm-----
Enter all inputs of process ids as zero-indexed
Enter total no. of processes
5
How many processes want to execute the critical region?
3
Enter ids of the processes entering critical region
----------------------------------------------------------------
Enter process id for 0th process
0
Enter the timestamp
36

Process 0 is sending request to enter critical region
Request sent to process 1
Request sent to process 2
Request sent to process 3
Request sent to process 4

All processes have received the request sent by process 0
All processes reply back to process 0
Reply received from process 1
Reply received from process 2
Reply received from process 3
Reply received from process 4
----------------------------------------------------------------
Enter process id for 1th process
2
Enter the timestamp
32

Process 2 is sending request to enter critical region
Request sent to process 0
Request sent to process 1
Request sent to process 3
Request sent to process 4

All processes have received the request sent by process 2
All processes reply back to process 2
Reply received from process 0
Reply received from process 1
```



```
VESIT307-4: ~/lamport

All processes have received the request sent by process 4
All processes reply back to process 4
Reply received from process 0
Reply received from process 1
Reply received from process 2
Reply received from process 3
----------------------------------------------------------------
Process 2 will get a chance to enter Critical region since it has a minimum timestamp of 32
The process 2 will execute for 1 seconds

Process 2 has completed executing in critical region
Resetting the timestamp of process 2

Process 2 is sending release message to all processes
Release sent to process 0
Release sent to process 1
Release sent to process 3
Release sent to process 4
----------------------------------------------------------------
Process 0 will get a chance to enter Critical region since it has a minimum timestamp of 36
The process 0 will execute for 3 seconds

Process 0 has completed executing in critical region
Resetting the timestamp of process 0

Process 0 is sending release message to all processes
Release sent to process 1
Release sent to process 2
Release sent to process 3
Release sent to process 4
----------------------------------------------------------------
Process 4 will get a chance to enter Critical region since it has a minimum timestamp of 76
The process 4 will execute for 2 seconds

Process 4 has completed executing in critical region
Resetting the timestamp of process 4

Process 4 is sending release message to all processes
Release sent to process 0
Release sent to process 1
Release sent to process 2
Release sent to process 3
```



```
VESIT307-4: ~/lamport                                         En   3:45 PM
----------------------------------------------------------------
Enter process id for 2th process
4
Enter the timestamp
76

Process 4 is sending request to enter critical region
Request sent to process 0
Request sent to process 1
Request sent to process 2
Request sent to process 3

All processes have received the request sent by process 4
All processes reply back to process 4
Reply received from process 0
Reply received from process 1
Reply received from process 2
Reply received from process 3
----------------------------------------------------------------
Process 2 will get a chance to enter Critical region since it has a minimum timestamp of 32
The process 2 will execute for 1 seconds

Process 2 has completed executing in critical region
Resetting the timestamp of process 2

Process 2 is sending release message to all processes
Release sent to process 0
Release sent to process 1
Release sent to process 3
Release sent to process 4
----------------------------------------------------------------
Process 0 will get a chance to enter Critical region since it has a minimum timestamp of 36
The process 0 will execute for 3 seconds

Process 0 has completed executing in critical region
Resetting the timestamp of process 0

Process 0 is sending release message to all processes
Release sent to process 1
Release sent to process 2
Release sent to process 3
Release sent to process 4
----------------------------------------------------------------
```