Terna Engineering College
Computer Engineering Department
Program: Sem VIII

Course: Distributed Computing Lab (CSL802)

Faculty: Rohini Patil

Experiment No. 4

A.1 Aim: To Implement Lamport Logical clock Algorithm.

PART B
(PART B: TO BE COMPLETED BY STUDENTS)

Roll No. 50	**Name:** AMEY MAHENDRA THAKUR
Class: BE COMPS B 50	**Batch:** B3
Date of Experiment: 03-02-2022	**Date of Submission:** 03-02-2022
Grade:	

B.1 Software Code written by student:

- **LCS.py**

```
# Python program to illustrate the Lamport's Logical Clock
# Function to find the maximum timestamp
# between 2 events
def max1(a, b) :

	# Return the greatest of th two
	if a > b :
		return a
	else :
		return b

# Function to display the logical timestamp
def display(e1, e2, p1, p2) :
	print()
	print("The time stamps of events in P1:")
	for i in range(0, e1) :
```

```
            print(p1[i], end = " ")

    print()
    print("The time stamps of events in P2:")

    # Print the array p2[]
    for i in range(0, e2) :
            print(p2[i], end = " ")

# Function to find the timestamp of events
def lamportLogicalClock(e1, e2, m) :
    p1 = [0]*e1
    p2 = [0]*e2

    # Initialize p1[] and p2[]
    for i in range (0, e1) :
            p1[i] = i + 1

    for i in range(0, e2) :
            p2[i] = i + 1

    for i in range(0, e2) :
            print(end = '\t')
            print("e2", end = "")
            print(i + 1, end = "")

    for i in range(0, e1) :
            print()
            print("e1", end = "")
            print(i + 1, end = "\t")

            for j in range(0, e2) :
                    print(m[i][j], end = "\t")

    for i in range(0, e1) :

            for j in range(0, e2) :

                    # Change the timestamp if the
                    # message is sent
                    if(m[i][j] == 1) :
                            p2[j] = max1(p2[j], p1[i] + 1)
                            for i in range(j + 1, e2) :
```

```
                p2[k] = p2[k - 1] + 1

        # Change the timestamp if the
        # message is received
        if(m[i][j] == -1) :
            p1[i] = max1(p1[i], p2[j] + 1)
            for k in range(i + 1, e1) :
                p1[k] = p1[k - 1] + 1

    # Function Call
    display(e1, e2, p1, p2)

# Driver Code

if __name__ == "__main__" :
    e1 = 5
    e2 = 3
    m = [[0]*3 for i in range(0,5)]

    # dep[i][j] = 1, if message is sent
    # from ei to ej
    # dep[i][j] = -1, if message is received
    # by ei from ej
    # dep[i][j] = 0, otherwise

    m[0][0] = 0
    m[0][1] = 0
    m[0][2] = 0
    m[1][0] = 0
    m[1][1] = 0
    m[1][2] = 1
    m[2][0] = 0
    m[2][1] = 0
    m[2][2] = 0
    m[3][0] = 0
    m[3][1] = 0
    m[3][2] = 0
    m[4][0] = 0
    m[4][1] = -1
    m[4][2] = 0

    # Function Call
    lamportLogicalClock(e1, e2, m)
```

B.2 Input and Output:

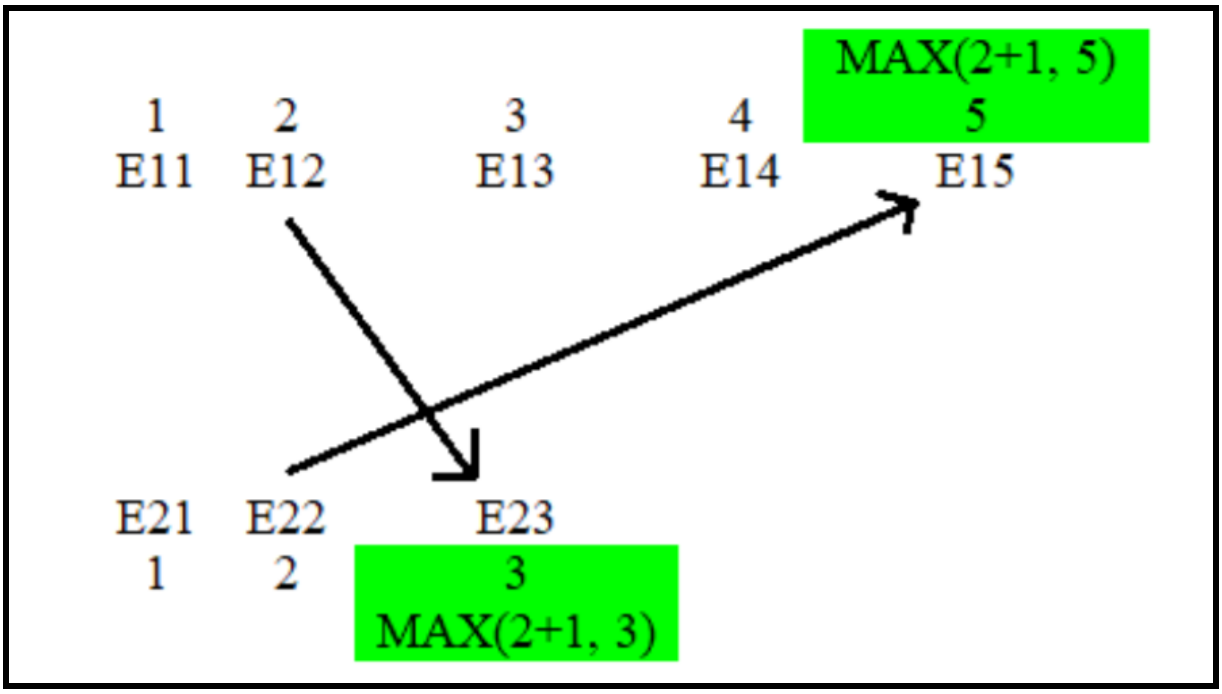
```
Command Prompt
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ameyt>cd Desktop

C:\Users\ameyt\Desktop>py LCS.py
        e21     e22     e23
e11     0       0       0
e12     0       0       1
e13     0       0       0
e14     0       0       0
e15     0       -1      0
The time stamps of events in P1:
1 2 3 4 5
The time stamps of events in P2:
1 2 3
C:\Users\ameyt\Desktop>
```

B.3 Observations and learning:

Lamport clocks algorithm:



Lamport clocks algorithm:
on initialisation do
    t := 0 #each node has its own local variable t
end on

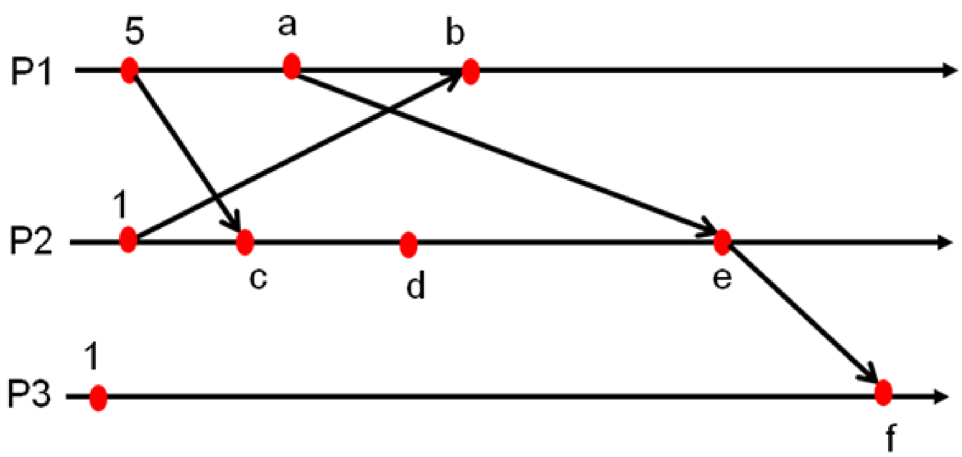
on any event occurring at the local node do
    t := t + 1
end on

on request to send message m do
    t := t + 1; send (t, m) via the underlying network link
end on

on receiving (t', m) via the underlying network link do
    t := max(t, t') + 1
    deliver m to the application
end on

B.4 Conclusion:
Successfully implemented Lamport Logical clock Algorithm.

B.5 Question of Curiosity.
Q1: Assign Lamport timestamps to the events (a, b, c, d, e, f) as shown in the figure:



1) a: 6, b: 2, c: 6, d: 7, e: 7, f: 8
2) a: 1, b: 2, c: 2, d: 3, e: 4, f: 2
3) a: 6, b: 7, c: 6, d: 7, e: 8, f: 9
4) a: 6, b: 7, c: 6, d: 7, e: 7, f: 8

ANS:
3) a: 6, b: 7, c: 6, d: 7, e: 8, f: 9
- Solved by the property of Scalar Time

Q2. Consider the following statements:

Event a has a Lamport timestamp of 3.
Event b has a Lamport timestamp of 6.

What can we tell about events a and b ?

1) Events a and b are causally related.
2) Events a and b are concurrent.
3) Event a happened before event b.
4) If events a and b are causally related, then event a happened before event b.

ANS:

4) If events a and b are causally related, then event a happened before event b.

- Scalar clocks satisfy the monotonicity and hence the consistency property: for two events ei and ej , ei → ej ⇒ C(ei) < C(ej).

Q3: Consider the following statements:

1. The system of vector clocks is not strongly consistent; that is, for two events ei and ej , C(ei) < C(ej) ==> ei → ej
2. The system of vector clocks is not strongly consistent; thus. By examining the scalar timestamp of two events, we can determine if the events are causally related

1) Both are true
2) Both are false
3) Only statement 1 is true
4) Only statement 2 is true

ANS:

2) Both are false

- Correct statements are:
  By the property of scalar clocks and vector clocks:
  1. The system of scalar clocks is not strongly consistent; that is, for two events ei and ej , C(ei) < C(ej) ⇒ ei → ej
  2. The system of vector clocks is strongly consistent; thus, by examining the vector timestamp of two events, we can determine if the events are causally related.

Q4. What is the problem with Lamport clocks that vector clocks solve?
ANS:

- The problem with Lamport Timestamps is that they can't tell if events are concurrent or not. This problem is solved by Vector Clocks.