INSTITUTE OF ENGINEERING

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

Kupondole, Lalitpur

(AFFILIATED TO TRIBHUVAN UNIVERSITY)

Lab No:3

Subject: Distributed System

Submitted By:

Name: Sameep Dhakal

Roll no: ACE074BCT063

Date: 04/07/2021

Submitted To:

Department of Computer and Electronics Engineering

Lab 3

Title:Lamport's Logical Clock IN C

Objective:

The objective of this lab is to implement Lamport's logical clock and analyse its output.

Software used:

Online C compiler was used for the compilation and running of code.

Introduction:

Lamport's Logical Clock was created by Leslie Lamport. It is a procedure to determine the order of events occurring. It provides a basis for the more advanced Vector Clock Algorithm. Due to the absence of a Global Clock in a Distributed Operating System Lamport Logical Clock is needed.

Algorithm:

- Happened before relation(->): a -> b, means 'a' happened before 'b'.
- Logical Clock: The criteria for the logical clocks are:
    - [C1]: Ci (a) < Ci(b), [Ci -> Logical Clock, If 'a' happened before 'b', then time of 'a' will be less than 'b' in a particular process.]
    - [C2]: Ci(a) < Cj(b), [Clock value of Ci(a) is less than Cj(b)]

Reference:

- Process: Pi
- Event: Eij, where i is the process in number and j: jth event in the ith process.
- tm: vector time span for message m.
- Ci vector clock associated with process Pi, the jth element is Ci[j] and contains Pi's latest value for the current time in process Pj.
- d: drift time, generally d is 1.

Implementation Rules[IR]:

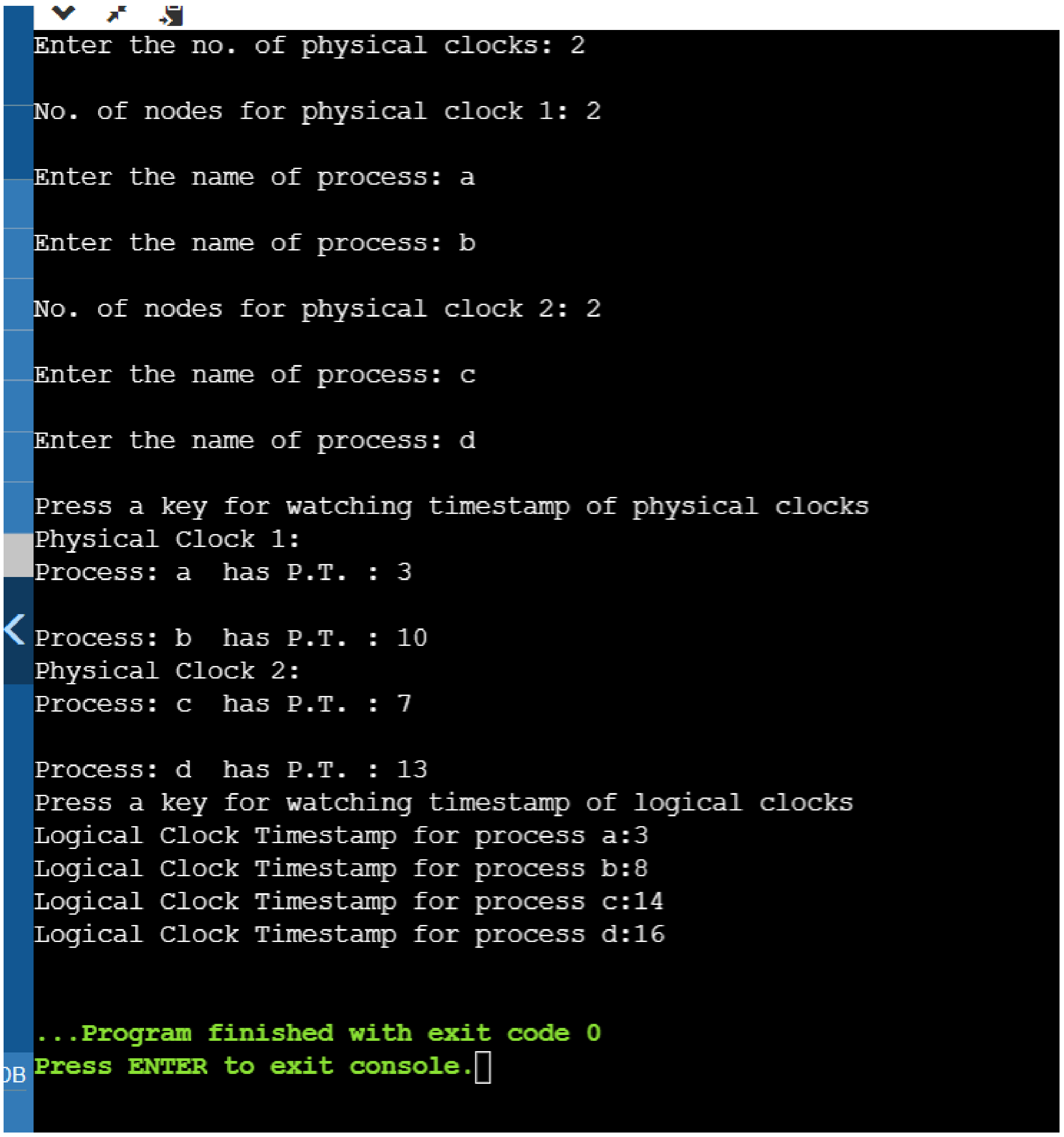
- [IR1]: If a -> b ['a' happened before 'b' within the same process] then, Ci(b) =Ci(a) + d
- [IR2]: Cj = max(Cj, tm + d) [If there's more number of processes, then tm = value of Ci(a), Cj = max value between Cj and tm + d]

Code implementation:

```
#include <stdio.h>

#include <conio.h>
```

```
#include <stdlib.h>

void main()

{

int i,j,k;

int time=0;

char process[10][10];

int num,clock[10],b[10][10];

//clrscr();

printf("Enter the no. of physical clocks: ");

scanf("%d",&num);

for(i=0;i<num;i++)

{

printf("\nNo. of nodes for physical clock %d: ",i+1);

scanf("%d",&clock[i]);

time=0;

for(j=0;j<clock[i];j++)

{

printf("\nEnter the name of process: ");

scanf(" %c",&process[i][j]);

b[i][j]=time + rand() % 10;

time=b[i][j]+1;

}

}

printf("\nPress a key for watching timestamp of physical clocks\n");

getch();

//clrscr();

for(i=0;i<num;i++)

{

printf("Physical Clock %d: ",i+1);

for(j=0;j<clock[i];j++)

{

printf("\nProcess: %c ",process[i][j]);
printf(" has P.T. : %d ",b[i][j]);

printf("\n");

}

}

printf("Press a key for watching timestamp of logical clocks\n");

getch();

//clrscr();

time=0;


for(j=0;j<num;j++)

{


for(k=0;k<clock[j];k++)

{


time = rand() % 10 + time;

printf("Logical Clock Timestamp for process %c",process[j][k]);

printf(":%d ",time);

printf("\n");

}}

getch();

return;

}
```

Result:



```
Enter the no. of physical clocks: 2

No. of nodes for physical clock 1: 2

Enter the name of process: a

Enter the name of process: b

No. of nodes for physical clock 2: 2

Enter the name of process: c

Enter the name of process: d

Press a key for watching timestamp of physical clocks
Physical Clock 1:
Process: a  has P.T. : 3

Process: b  has P.T. : 10
Physical Clock 2:
Process: c  has P.T. : 7

Process: d  has P.T. : 13
Press a key for watching timestamp of logical clocks
Logical Clock Timestamp for process a:3
Logical Clock Timestamp for process b:8
Logical Clock Timestamp for process c:14
Logical Clock Timestamp for process d:16


...Program finished with exit code 0
Press ENTER to exit console.
```

Discussion And Conclusion:

In this lab we entered number of physical clock required and nodes in those clocks. After that we watched timestamp for different clocks and their nodes. We find out that every process had different time stamps. We also revised c programming.

Thus in this lab we implemented the Lamport's logical clock and analyzed its output through c Programming.