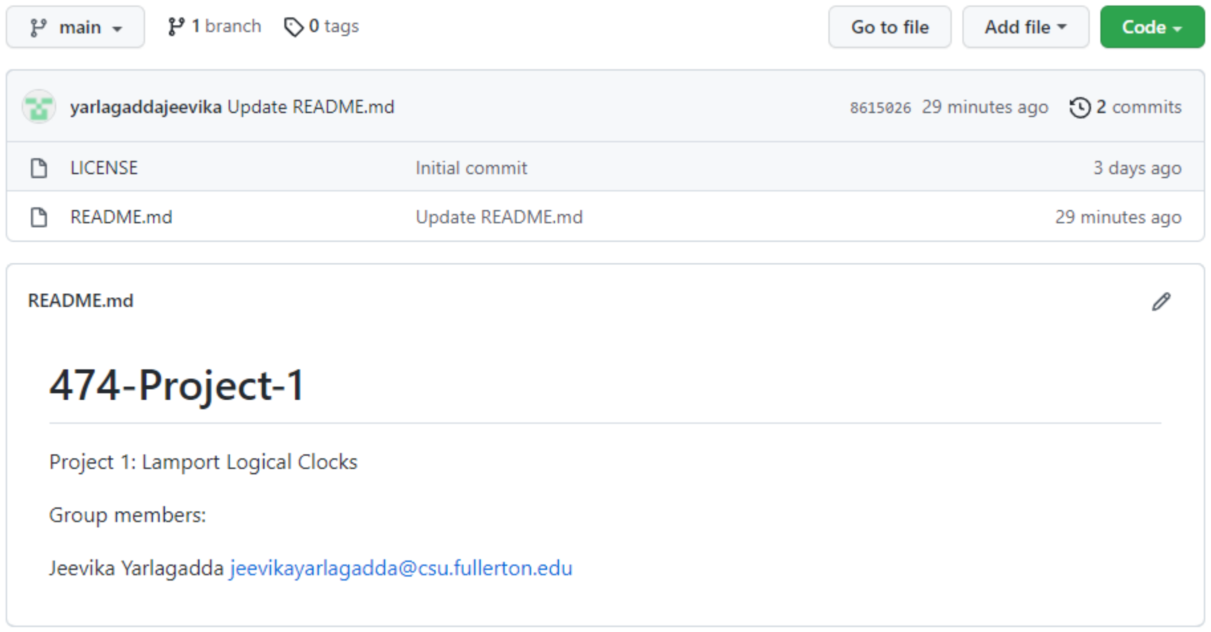
CPSC 474 – Project 1

GitHub url: https://github.com/CSUF-CPSC-Bein-FA21/project-1-servers-jeevika

Group member: Jeevika Yarlagadda, jeevikayarlagadda@csu.fullerton.edu

Submission for Project 1 – Lamport Logical Clocks



yarlagaddajeevika Update README.md 8615026 29 minutes ago 2 commits

LICENSE	Initial commit	3 days ago
README.md	Update README.md	29 minutes ago

README.md

474-Project-1

Project 1: Lamport Logical Clocks

Group members:

Jeevika Yarlagadda jeevikayarlagadda@csu.fullerton.edu

Summary:

Most of us use physical time to order events. For example, we say that an event at 8:15 AM occurs before an event at 8:16 AM. In distributed systems, physical clocks are not always precise, so we can't rely on physical time to order events.

Instead, we can use logical clocks to create a partial or total ordering of events. For such case, We need to identify the Lamport logical clock timestamp for each message that was sent and received. This will be achieved by calculating the lamport logical value of an event using the Lamport Calculate algorithm and this can be verified using Lamport Verify algorithm.

Pseudo Codes:

Algorithm Calculate:

Send events starts with ‘s’ and receiver events are ‘r’, ‘Null’ represents nothing, rest all alphabets are considered as internal events.

Step 1: Enter the input

Enter the number of processes(rows) and the number of events in a process(columns)

Store input in the form of a matrix array

Initialize values in output matrix with -2(some initial value)

Step 2: Run the loop for the input matrix

- If the event is an internal event or send event and in the first column, then the value of the event is 1. Update the output matrix with value.
- Increment the column position and move on to the next element.
- If the event is a sender event or internal event but not in the first column, value will be the of the previous event value plus 1. Update the output matrix with value.
- If the event is a receive event and in the first column, find the corresponding send event and add 1 to it. This will be the value of the receive event.
- If the event is a receive event not in the first column, hold the loop there and extract the number of the receiver event and run one more for loop to find the corresponding send event. The value of the send event depends on the previous value. This previous value can be found by running the above 2 conditions in a recursive flow.
- Once you find the send event value, then find the maximum of the send event and the previous event of receive plus 1, this will be the value of the receive event.
- Continue the loop after the receive event.
- All these steps will be run in a recursive flow when you try to find the receive event value.
- All the values will be updated in the output matrix.
- If the value is “NULL”, 0 will be updated in the output matrix.

Step 3: Print the output

Loop comes to an end when it reaches the number of processes and the events in the process, the execution is completed. Now, Print the output matrix which has the updated values.

Algorithm Verify:

All the positive integers are treated as events and there is an event with 0, it is considered as NULL.

Step 1: Enter the input in a file called verify.txt

Step 2: Read the input from file and store it in a matrix.

Step 3: Find the difference between the events in a row. If the difference is huge, mark it as receiver.

- Receiver value minus 1 will be the sender event value. Loop the matrix and find the send event.
- Ensure the receiver and sender event are not in the same row(process).
- The number of the first receiver event will be maximum number of rows(process) and this number will be reduced by 1 when it finds another receive event.
- Repeat the same process for the number of process and find all the receiver and sender events correspondingly.
- If the value encountered is 0 from input, put it as NULL.
- Rest all the other events are internal events. Define a int value corresponding to the alphabet and increment the count when an internal event is occurred.
- If there are receive events without send events, it means that the given input is incorrect.

Step 4: Print the output

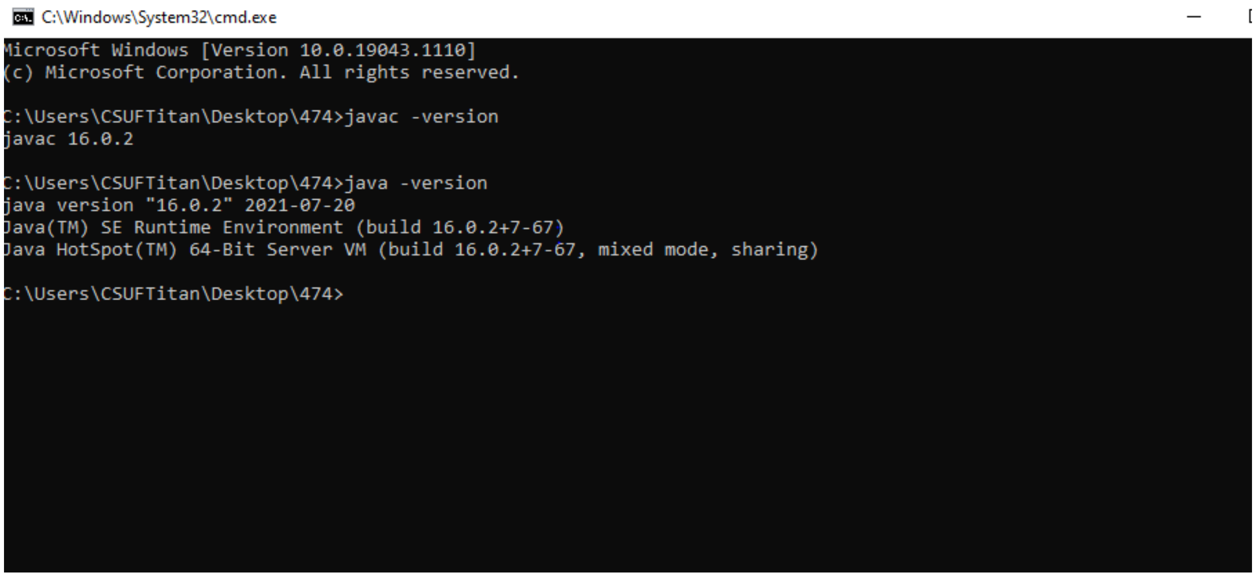
Loop comes to an end when it reaches the number of processes and the events in the process, the execution is completed. Now, Print the output matrix which has the updated values.

Compile the Code and Run:

Code is written in Java.

Compile used: javac 16.0.2

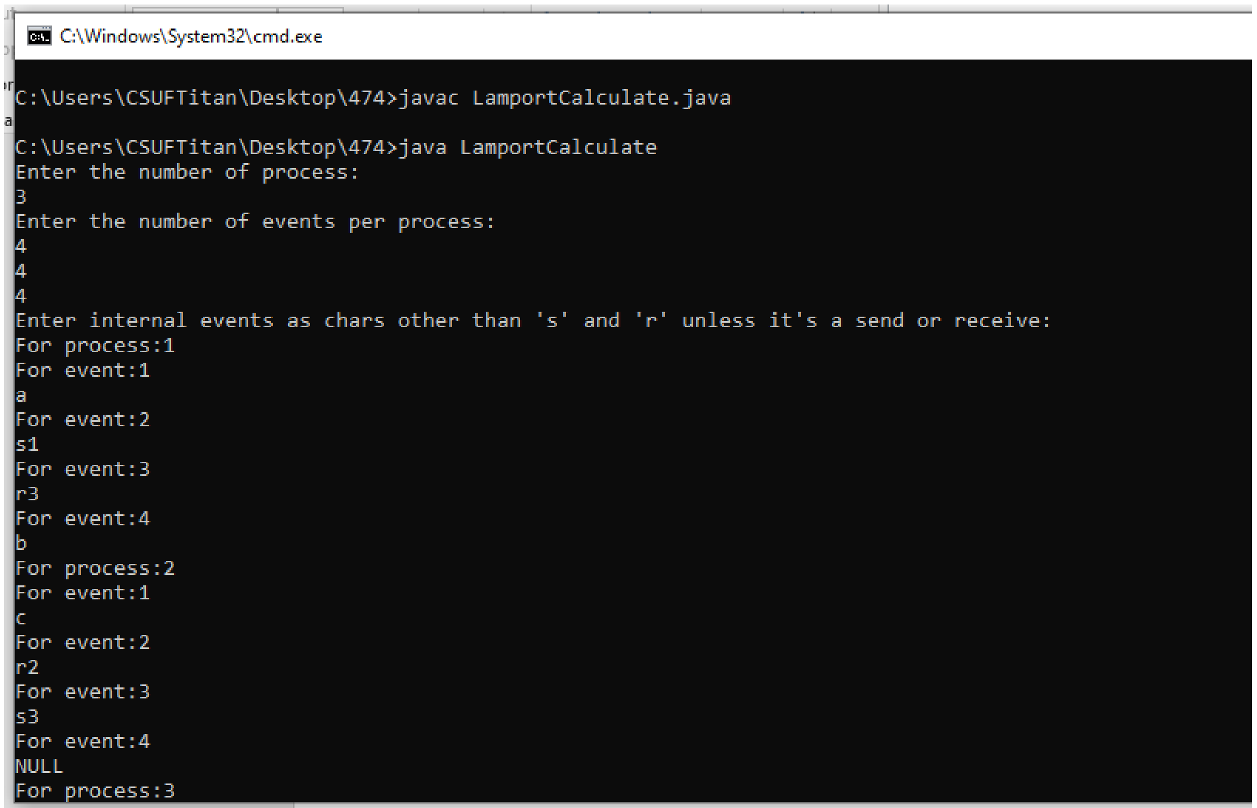
Java version: java version "16.0.2"



```
C:\Windows\System32\cmd.exe
icrosoft Windows [Version 10.0.19043.1110]
(c) Microsoft Corporation. All rights reserved.

:\Users\CSUFTitan\Desktop\474>javac -version
javac 16.0.2

:\Users\CSUFTitan\Desktop\474>java -version
java version "16.0.2" 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)

:\Users\CSUFTitan\Desktop\474>
```

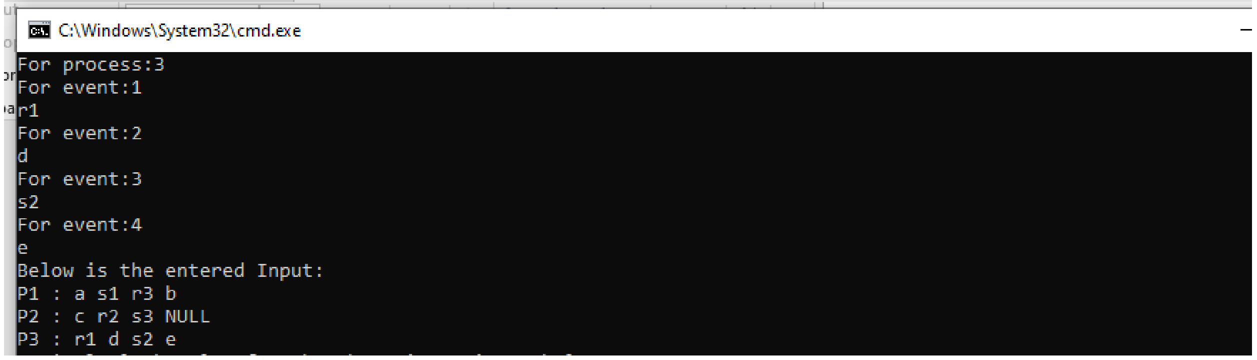
Algorithm Calculate:

- Once you download the file, please open the command prompt in the path of the file.
- Compile the file LamportCalculate.java on command prompt.
  command: javac LamportCalculate.java
- Run the code on the command prompt: java LamportCalculate
  It will ask you to enter the number of process that is the number of rows.
  Once you enter, it will prompt you to Enter the number of events per process(columns in each row)
  Enter all the values of the event for the processes. Once it is completed, it will give you the output.



C:\Windows\System32\cmd.exe

```
C:\Users\CSUFTitan\Desktop\474>javac LamportCalculate.java

C:\Users\CSUFTitan\Desktop\474>java LamportCalculate
Enter the number of process:
3
Enter the number of events per process:
4
4
4
Enter internal events as chars other than 's' and 'r' unless it's a send or receive:
For process:1
For event:1
a
For event:2
s1
For event:3
r3
For event:4
b
For process:2
For event:1
c
For event:2
r2
For event:3
s3
For event:4
NULL
For process:3
```

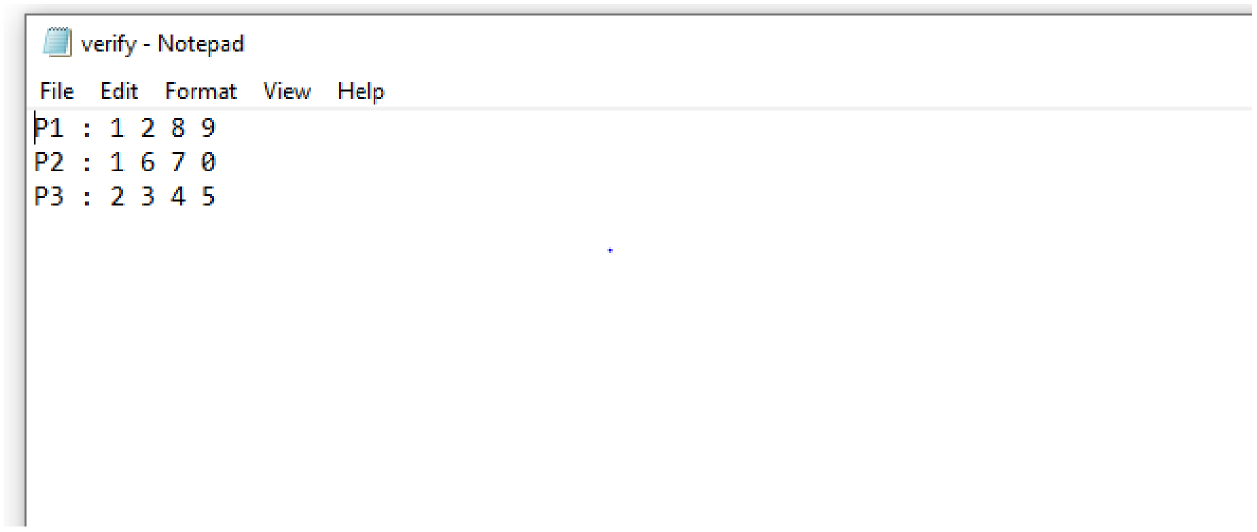


C:\Windows\System32\cmd.exe

```
For process:3
For event:1
r1
For event:2
d
For event:3
s2
For event:4
e
Below is the entered Input:
P1 : a s1 r3 b
P2 : c r2 s3 NULL
P3 : r1 d s2 e
```

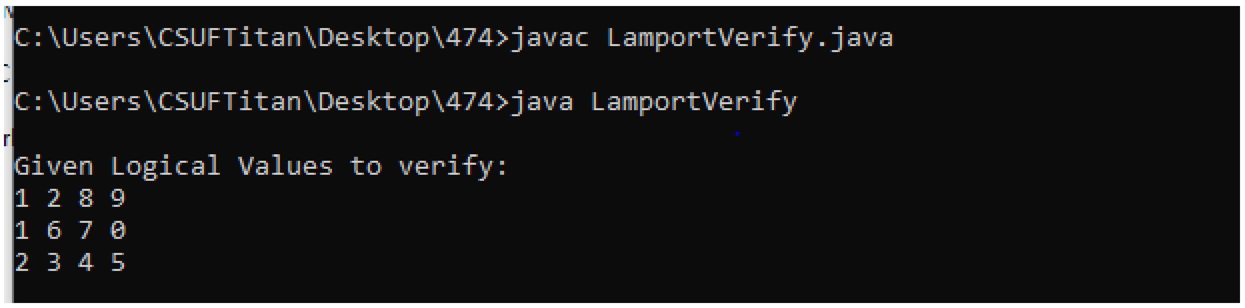
Algorithm Verify:

- Once you download the file, please open the command prompt in the path of the file.
- Compile the file LamportVerify.java using command prompt.
  command: javac LamportVerify.java
- Please download the text file "verify.txt", this is used as an input file to the verify algorithm. Enter the input in the file as shown:



```
P1 : 1 2 8 9
P2 : 1 6 7 0
P3 : 2 3 4 5
```

- Please save the text file.
- Go to the command prompt and Run the code using the command: java LamportVerify

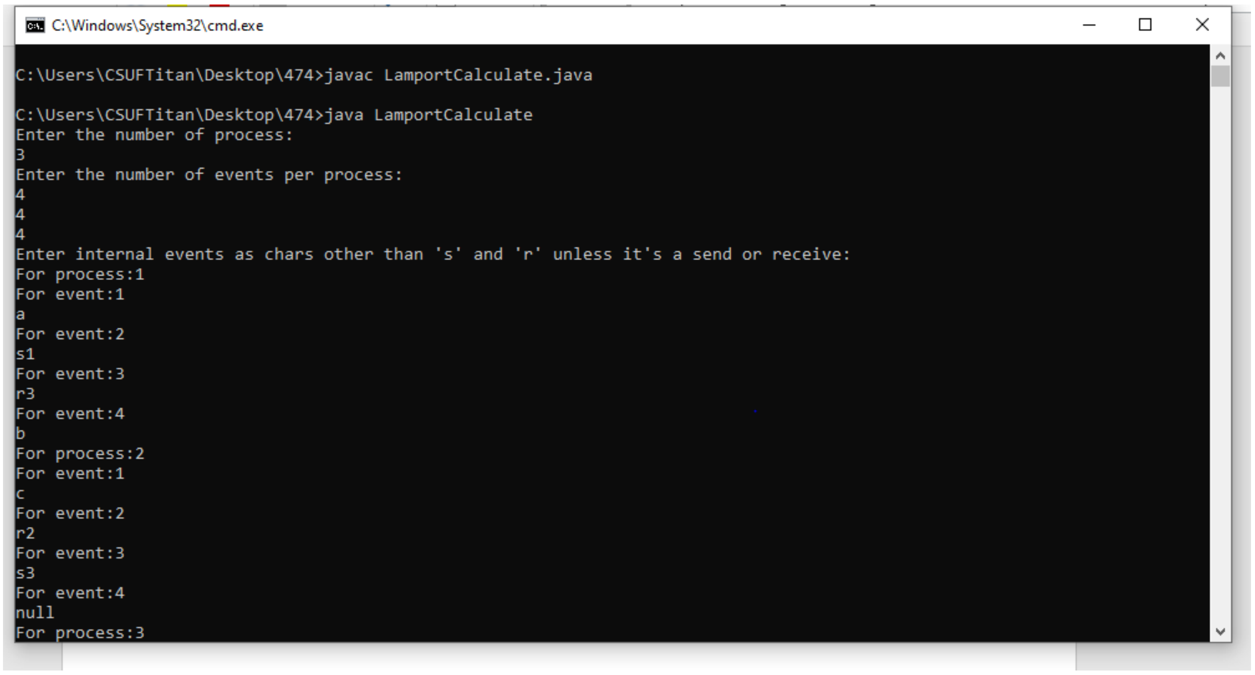


```
C:\Users\CSUFTitan\Desktop\474>javac LamportVerify.java

C:\Users\CSUFTitan\Desktop\474>java LamportVerify

Given Logical Values to verify:
1 2 8 9
1 6 7 0
2 3 4 5
```

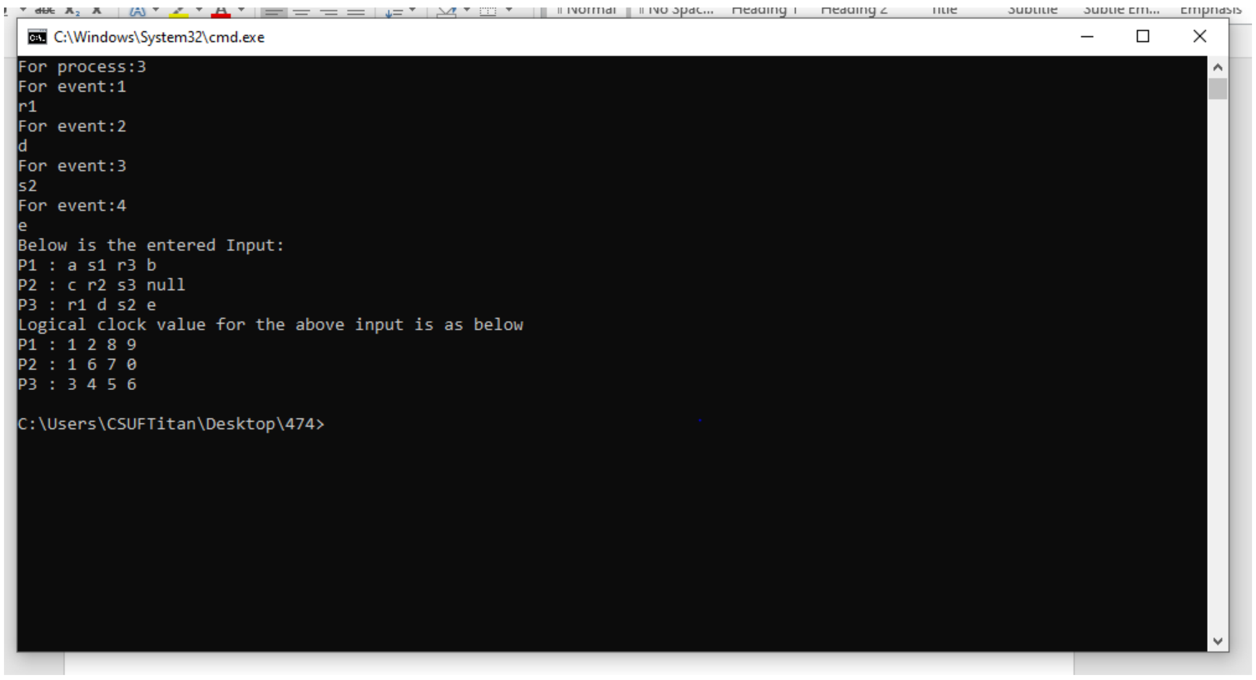
Snapshots of the code with the Test data:

Algorithm Calculate:

Test data 1:

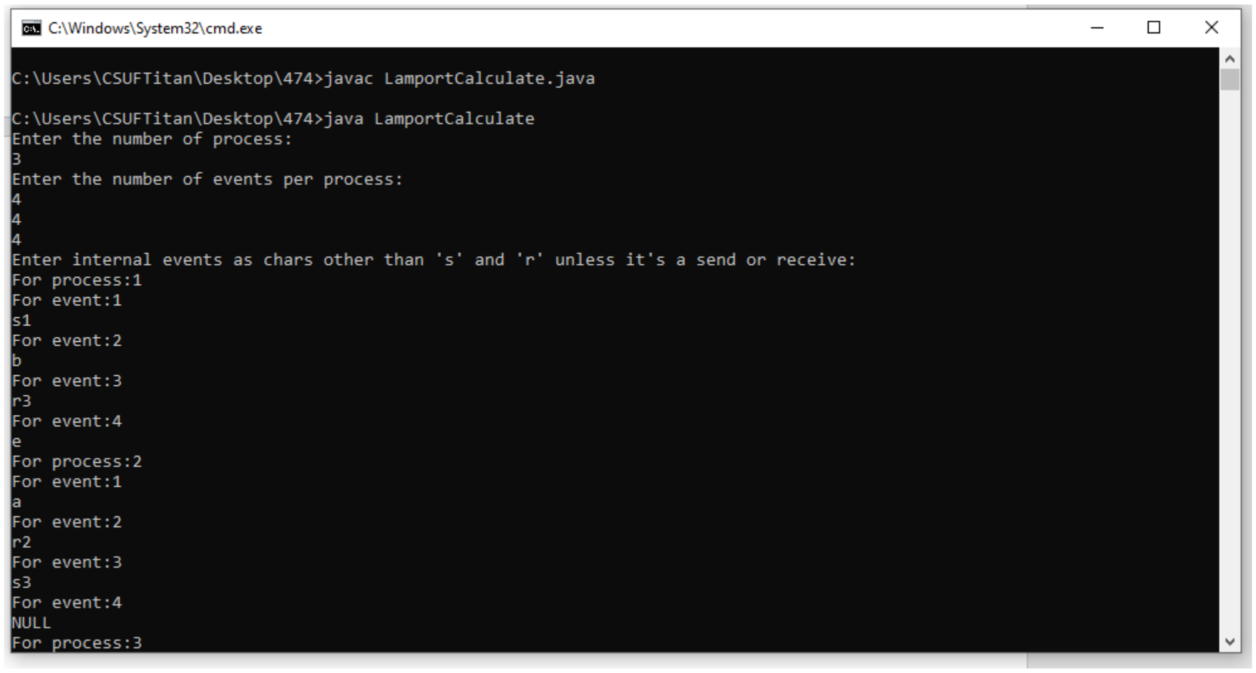


```
C:\Users\CSUFTitan\Desktop\474>javac LamportCalculate.java

C:\Users\CSUFTitan\Desktop\474>java LamportCalculate
Enter the number of process:
3
Enter the number of events per process:
4
4
4
Enter internal events as chars other than 's' and 'r' unless it's a send or receive:
For process:1
For event:1
a
For event:2
s1
For event:3
r3
For event:4
b
For process:2
For event:1
c
For event:2
r2
For event:3
s3
For event:4
null
For process:3
```

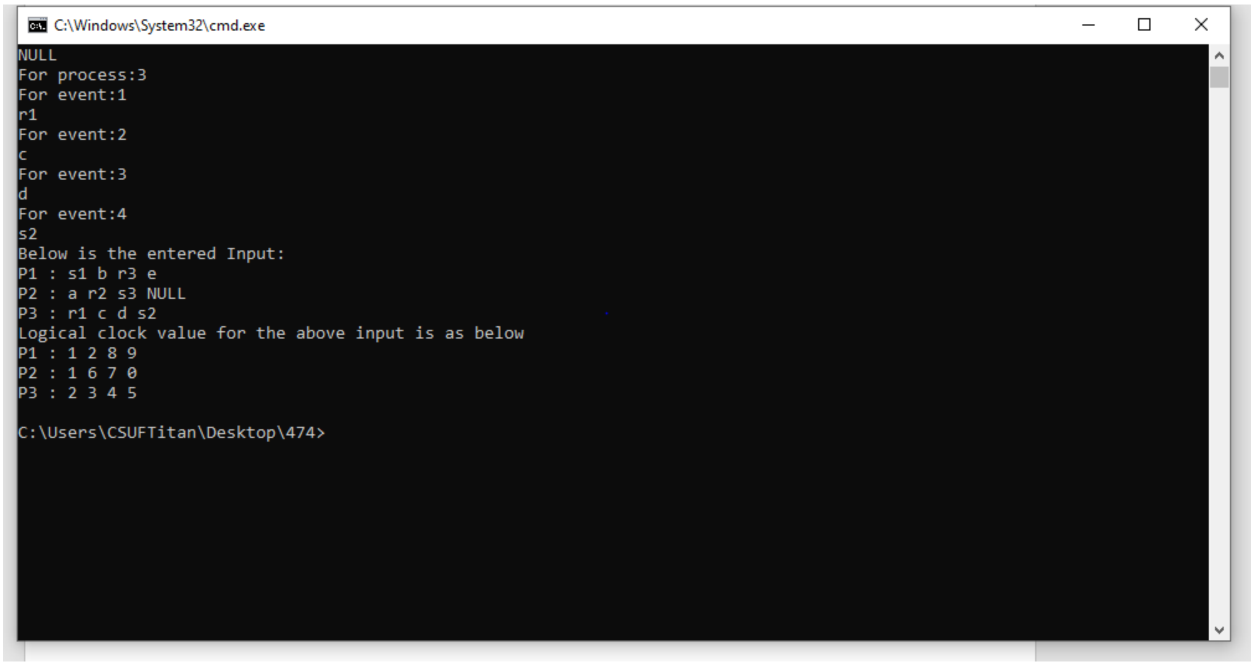


```
For process:3
For event:1
r1
For event:2
d
For event:3
s2
For event:4
e
Below is the entered Input:
P1 : a s1 r3 b
P2 : c r2 s3 null
P3 : r1 d s2 e
Logical clock value for the above input is as below
P1 : 1 2 8 9
P2 : 1 6 7 0
P3 : 3 4 5 6

C:\Users\CSUFTitan\Desktop\474>
```

Test data 2:



```
C:\Users\CSUFTitan\Desktop\474>javac LamportCalculate.java

C:\Users\CSUFTitan\Desktop\474>java LamportCalculate
Enter the number of process:
3
Enter the number of events per process:
4
4
4
Enter internal events as chars other than 's' and 'r' unless it's a send or receive:
For process:1
For event:1
s1
For event:2
b
For event:3
r3
For event:4
e
For process:2
For event:1
a
For event:2
r2
For event:3
s3
For event:4
NULL
For process:3
```

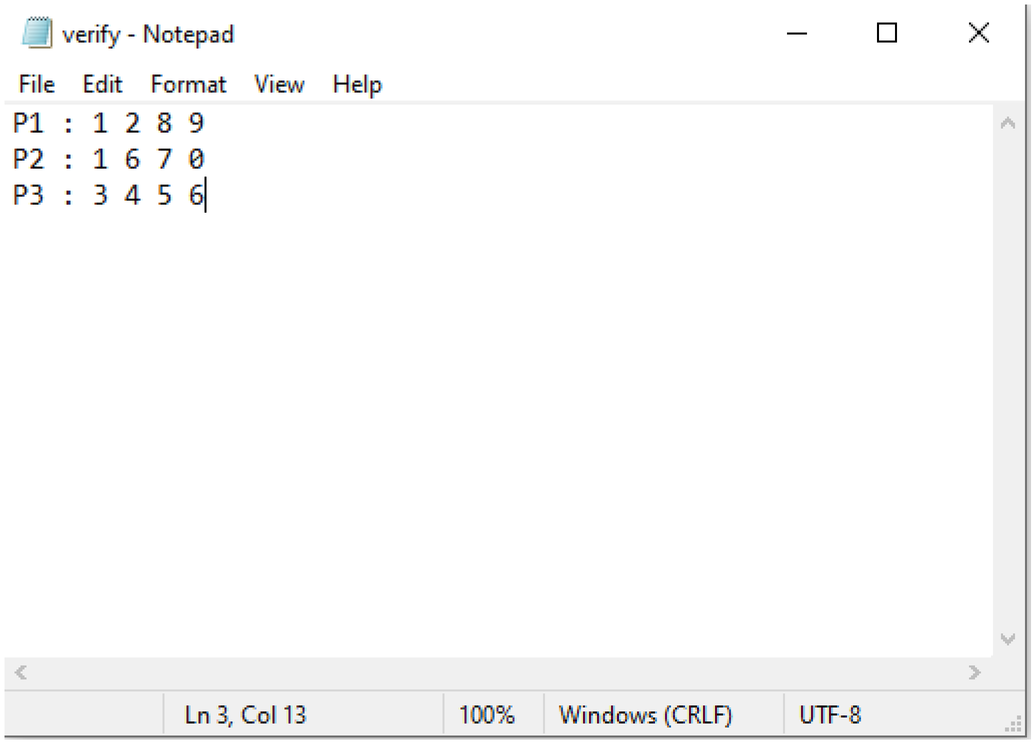


```
NULL
For process:3
For event:1
r1
For event:2
c
For event:3
d
For event:4
s2
Below is the entered Input:
P1 : s1 b r3 e
P2 : a r2 s3 NULL
P3 : r1 c d s2
Logical clock value for the above input is as below
P1 : 1 2 8 9
P2 : 1 6 7 0
P3 : 2 3 4 5

C:\Users\CSUFTitan\Desktop\474>
```

Algorithm Verify:

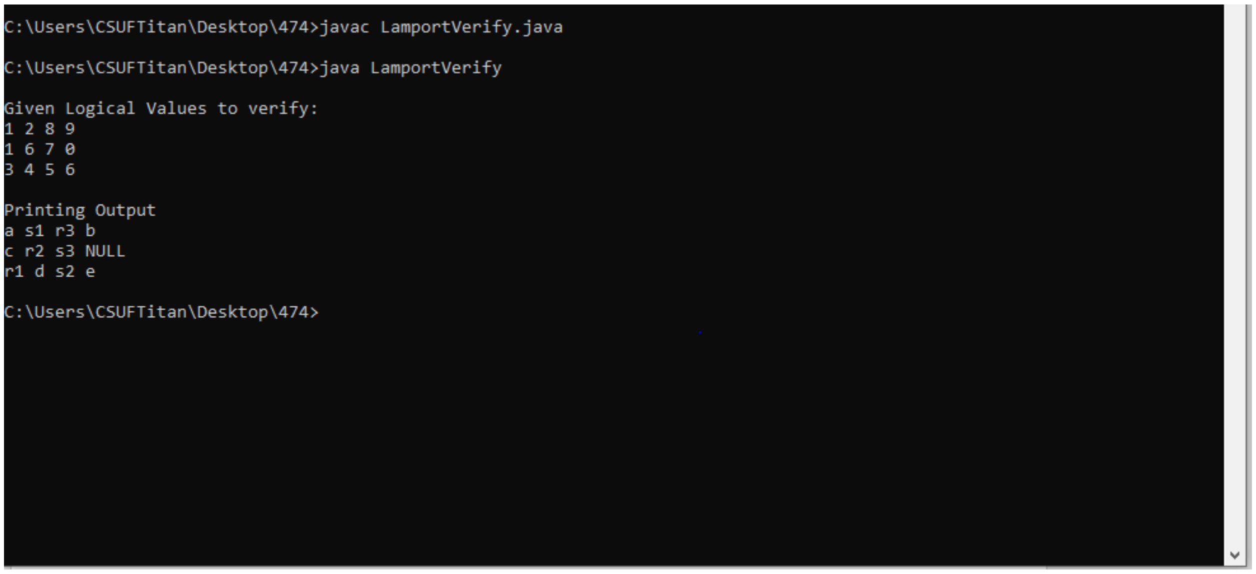
Test data 1:

Input:



```
P1 : 1 2 8 9
P2 : 1 6 7 0
P3 : 3 4 5 6
```

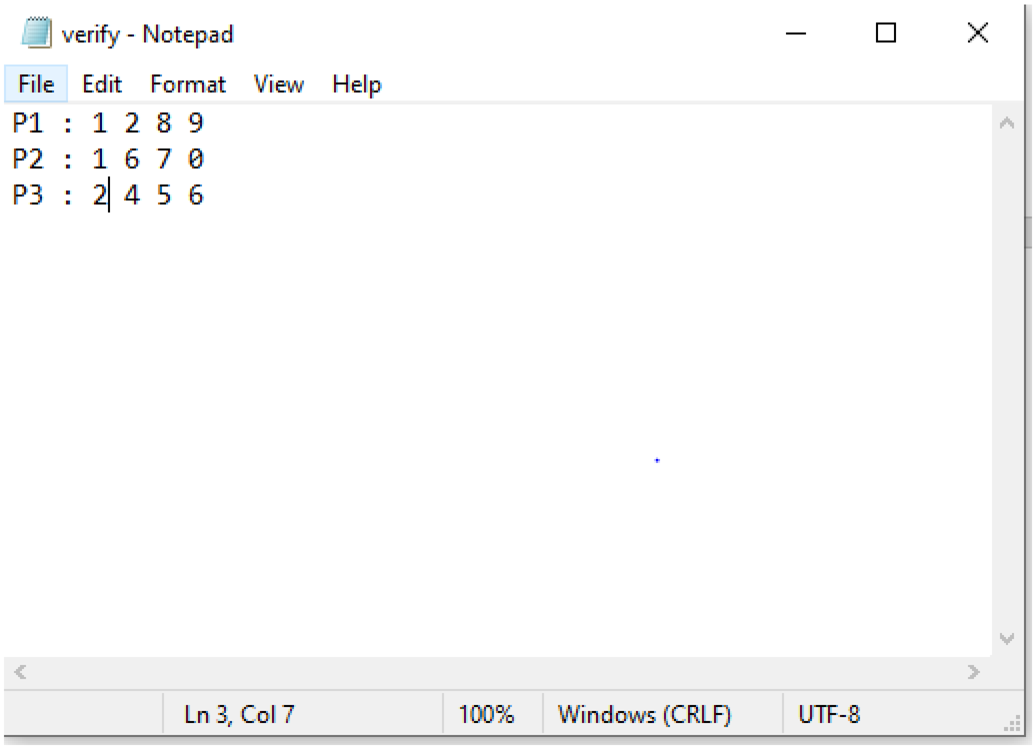
Output:



```
C:\Users\CSUFTitan\Desktop\474>javac LamportVerify.java

C:\Users\CSUFTitan\Desktop\474>java LamportVerify

Given Logical Values to verify:
1 2 8 9
1 6 7 0
3 4 5 6

Printing Output
a s1 r3 b
c r2 s3 NULL
r1 d s2 e

C:\Users\CSUFTitan\Desktop\474>
```

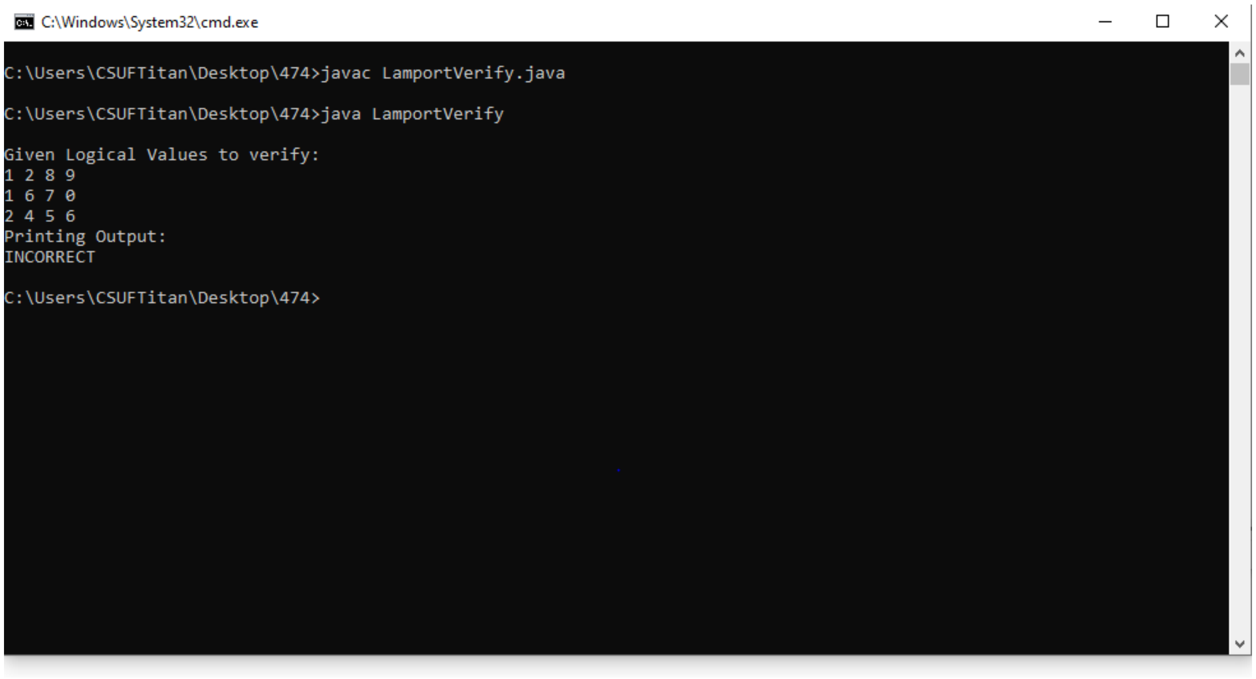
Test data 2:

Input:



```
P1 : 1 2 8 9
P2 : 1 6 7 0
P3 : 2 4 5 6
```

Output:



```
C:\Users\CSUFTitan\Desktop\474>javac LamportVerify.java

C:\Users\CSUFTitan\Desktop\474>java LamportVerify

Given Logical Values to verify:
1 2 8 9
1 6 7 0
2 4 5 6
Printing Output:
INCORRECT

C:\Users\CSUFTitan\Desktop\474>
```