Devante Wilson Lab #4: Implementing Logical Clocks SOFE 4790U
100554361

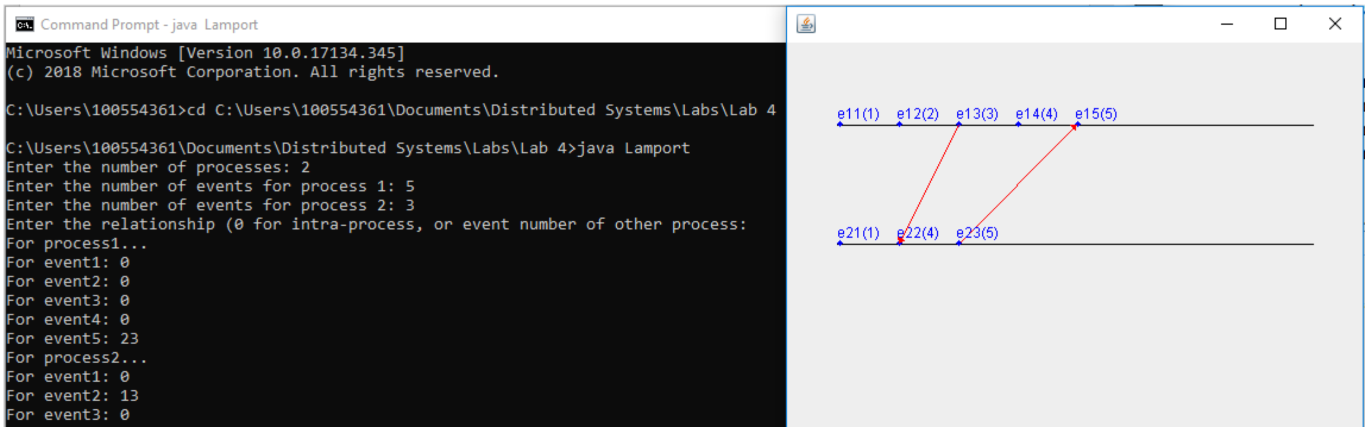
Task #1: Visualizing Lamport Logical Clocks

1. The results of both samples runs ended of showing the expected output. The visual representation is important to understand the logical clock relationship as it illustrates the order of the events, the amount of events for each process, the task timestamps, and the relationship for each communicating task (send, receive, intra-process step). If there were only the command line outputs, it would be confusing to the user to distinguish exactly what is happening.

2. The only challenge that was encountered was initially figuring out how to denote the relationships via the command line (ex. 23 means that process 2 task 3 sent a message to the receiving task).

3. This task helped me to solidify my understanding the events/relationships of logical clocks.

Sample Run #1



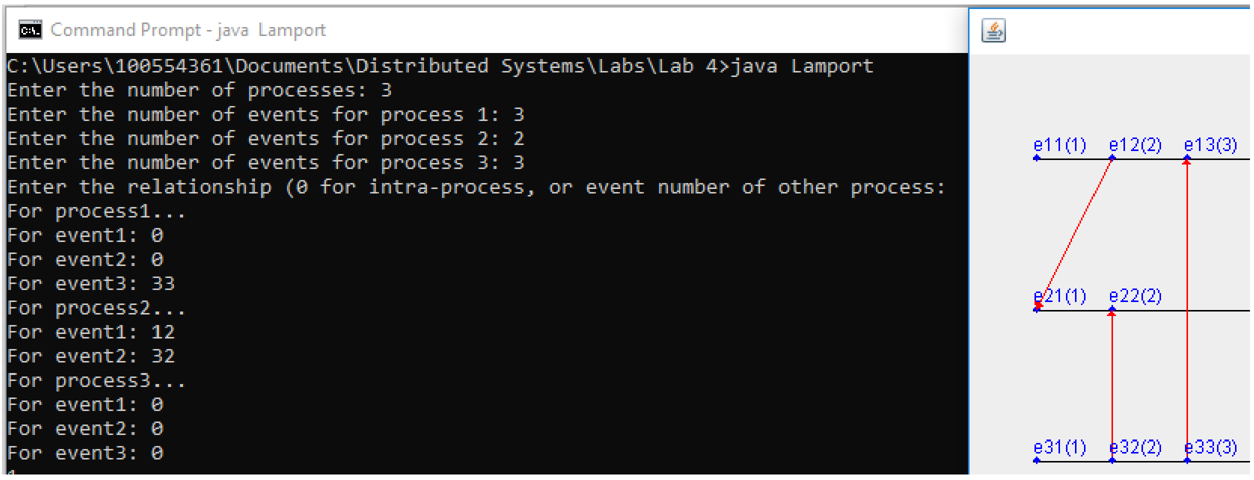
Command Prompt - java Lamport
e11(1) e12(2) e13(3) e14(4) e15(5)
e21(1) e22(4) e23(5)

```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\100554361>cd C:\Users\100554361\Documents\Distributed Systems\Labs\Lab 4

C:\Users\100554361\Documents\Distributed Systems\Labs\Lab 4>java Lamport
Enter the number of processes: 2
Enter the number of events for process 1: 5
Enter the number of events for process 2: 3
Enter the relationship (0 for intra-process, or event number of other process:
For process1...
For event1: 0
For event2: 0
For event3: 0
For event4: 0
For event5: 23
For process2...
For event1: 0
For event2: 13
For event3: 0
```

Sample Run #2



```
C:\Users\100554361\Documents\Distributed Systems\Labs\Lab 4>java Lamport
Enter the number of processes: 3
Enter the number of events for process 1: 3
Enter the number of events for process 2: 2
Enter the number of events for process 3: 3
Enter the relationship (0 for intra-process, or event number of other process:
For process1...
For event1: 0
For event2: 0
For event3: 33
For process2...
For event1: 12
For event2: 32
For process3...
For event1: 0
For event2: 0
For event3: 0
```

Task #2: Logical Timestamps

From this task, I learned how to handle a dynamic amount of processes and dynamic amount of events for each process, and how to denote the relationship between the processes. Additionally, this task helped to solidify my understanding of Lamport logical clocks; more specifically, understanding the *"happened before"* relationship for events. To accomplish this, Java threads were used to simulate processes. At the end of the program, the logical timestamp for every event was shown depending on the inputs.

Task #3: Vector Clocks

The most challenging part of this task was how to overcome the shortcomings of Lamport's logical clock. To accomplish this, each of the tasks are assigned an ID, then for each modification, this ID, along with the last vector clock timestamp that was seen is included. If two events store a change with vector clocks that do not descend (precede) from each other, the vector clock timestamp that has been succeeded will be removed.