Assignment2 Report

Reproducing Lamport Clock:

- Take note of the input size for processes. In this example we have 2 processes with number of events being 7 for process1 and 6 for process2. So, we would need a 2d array of size 7*6 to store any arrows. And two 1d arrays of size 7 and 6 to represent timestamps for the two processes.

```
public class LamportClock{
    public static void main(String[]args) {
        int process1[] = new int[7];
        int process2[] = new int[6];
        int arrows[][] = new int[7][6];
```

- Now initialize 2d array based on example input and process arrays in incrementing order staring from 1.

```
    public static void takeInputAndInitialize(int[][] arrows, int[] process1, int[] process2) {
        for(int i=0; i<7; i++) {
            for(int j=0; j<6; j++) {
                arrows[i][j]=0;
            }
        }
        arrows[1][2] = 1;
        arrows[4][4] = 1;
        arrows[6][3] = -1;

        for(int i=0; i<process1.length; i++) {
            process1[i] = i+1;
        }

        for(int i=0; i<process2.length; i++) {
            process2[i] = i+1;
        }
    }
```

- Now calculate the timestamps for each events in both processes based on the input 2d array and below condition:
  By default, timestamps will be in incrementing order unless there is an incoming edge. In that case we take max of default and edge_origin_timestamp+1.
  Also update the time stamps of the events following the incoming edge event.

```
public static void calculateClock(int[] process1, int[] process2, int[][] arrows) {

    for(int i=0; i<process1.length; i++) {
        for(int j=0; j<process2.length; j++) {

            if(arrows[i][j]==1) {
                process2[j] = Math.max(process2[j], process1[i]+1);
                for(int repair=j+1; repair<process2.length; repair++) {
                    process2[repair] = process2[repair-1] + 1;
                }
            }

            if(arrows[i][j]==-1) {
                process1[i] = Math.max(process1[i], process2[j]+1);
                for(int repair=i+1; repair<process1.length; repair++) {
                    process1[repair] = process1[repair-1] + 1;
```

- Display the input matrix and resultant timestamps for all the events.

```
        System.out.println("Input:");
        for(int i=0; i<7; i++) {
            System.out.println("");
            for(int j=0; j<6; j++) {
                if(arrows[i][j]==-1)
                    System.out.print(arrows[i][j]+"  ");
                else
                System.out.print(arrows[i][j]+"   ");
            }
        }

        System.out.println("");
        System.out.println("");
        System.out.println("");
        System.out.println("Output:");
        System.out.println("");
        System.out.println("EventNumber    TimeStamp");
        for(int i=0; i<process1.length; i++) {
            System.out.println("      e1" + (i+1) + ":           " + process1[i]);
        }

        for(int i=0; i<process2.length; i++) {
            System.out.println("      e2" + (i+1) + ":           " + process2[i]);
        }
```

```
Input:

0   0   0   0   0   0
0   0   1   0   0   0
0   0   0   0   0   0
0   0   0   0   0   0
0   0   0   0   1   0
0   0   0   0   0   0
0   0   0   -1  0   0


Output:

EventNumber    TimeStamp
      e11:           1
      e12:           2
      e13:           3
      e14:           4
      e15:           5
      e16:           6
      e17:           7
      e21:           1
      e22:           2
      e23:           3
      e24:           4
      e25:           6
      e26:           7
```