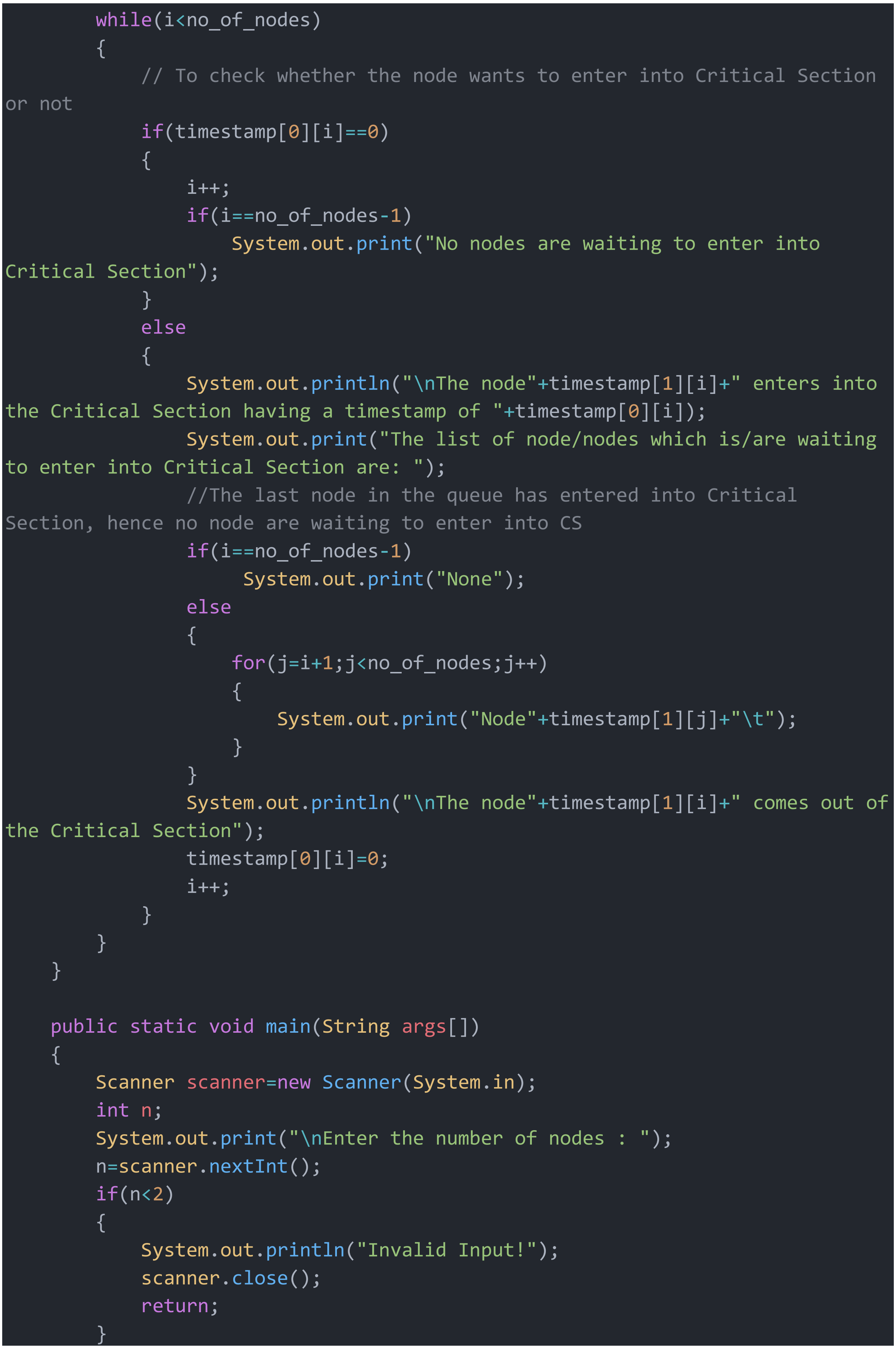
Assignment 2

Assumptions

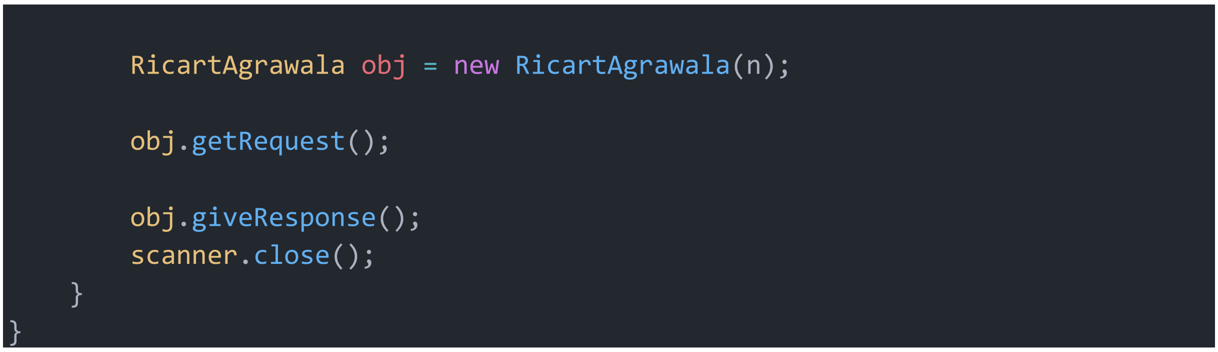
1. The range of the input nodes are from 1 to n, where n is given by the user.
2. At first it is assumed that one node is in the Critical Section. Hence, the total number of nodes is n+1.
3. The 0 value in the timestamp matrix indicates that, that corresponding node is not interested in entering into the Critical Section.
4. Input should either be Y/y if the respective node wants to enter into the Critical Section or N/n if the respective node does not want to enter into the Critical Section.
5. If two or more nodes with the same timestamp are waiting to enter into the Critical Section, then the node having least ID number is chosen.

Source Code

```java
// To implement Ricart Agrawala Symmetric Algorithm
import java.util.Scanner;

class RicartAgrawala
{
    int no_of_nodes, timestamp[][];
    // timestamp is a matrix of size 2xN where the ith column of the second row is the node having a timestamp stored in the ith column of the first row
    // 0 values in the first row of the timestamp indicates that the corresponding node is not interested in entering into the critical section

    public RicartAgrawala(int n)
    {
        no_of_nodes=n;
        timestamp=new int [2][no_of_nodes];
    }

    public void getRequest()
    {
        Scanner scanner=new Scanner(System.in);
        int i,val;
        char ans;
        for(i=0;i<no_of_nodes;i++)
        {
            // Input should either be Y if the respective node wants to enter into critical section or N if the repective node does not want to enter into critical scetion
            System.out.print("\n Does the node "+(i+1)+" want to enter into Critical Section (Y/N) ? ");
            ans=scanner.next().charAt(0);
            if(ans=='y' || ans=='Y')
            {
                System.out.print("\nEnter the timestamp of the node"+(i+1)+"
:");
                val=scanner.nextInt();
                if(val<1)
                {
                    System.out.println("Invalid Input!");
                    scanner.close();
                    System.exit(0);
                }
                timestamp[0][i]=val;
                timestamp[1][i]=i+1;
            }
            else if(ans=='n' || ans=='N')
                timestamp[1][i]=i+1;
            else
            {
                System.out.println("Invalid Input!");
                scanner.close();
                System.exit(0);
            }
        }
        scanner.close();
    }

    public void giveResponse()
    {
        int i,j,temp1,temp2;

        for (i = 0; i < no_of_nodes-1; i++)
        {
            for (j = 0; j < no_of_nodes-i-1; j++)
            {
                if (timestamp[0][j] > timestamp[0][j+1])
                {
                    // The matrix timestamp is sorted with respect to the
first row.
                    temp1 = timestamp[0][j];
                    temp2 = timestamp[1][j];
                    timestamp[0][j] = timestamp[0][j+1];
                    timestamp[1][j] = timestamp[1][j+1];
                    timestamp[0][j+1] = temp1;
                    timestamp[1][j+1] = temp2;
                }
            }
        }

        System.out.println("The node M comes out of the Critical Section");
        i=0;
```

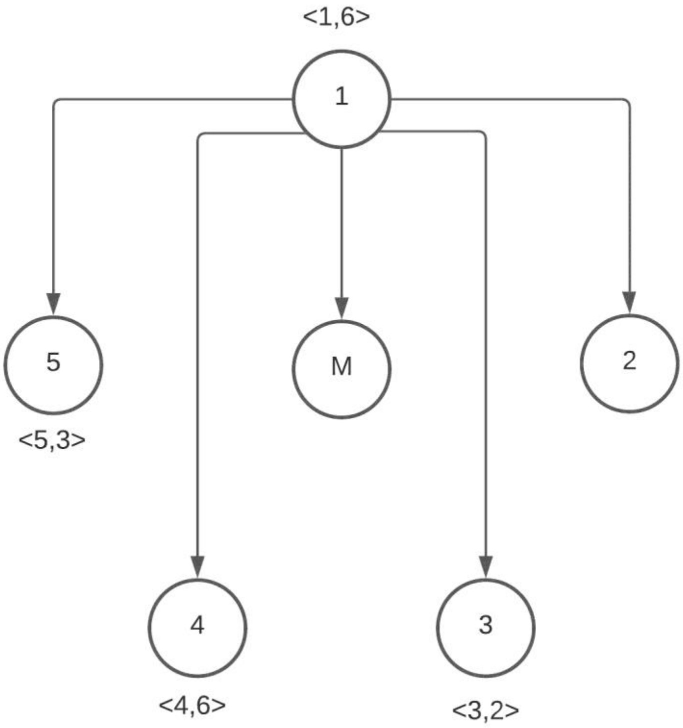


```
        while(i<no_of_nodes)
        {
            // To check whether the node wants to enter into Critical Section
or not
            if(timestamp[0][i]==0)
            {
                i++;
                if(i==no_of_nodes-1)
                    System.out.print("No nodes are waiting to enter into
Critical Section");
            }
            else
            {
                System.out.println("\nThe node"+timestamp[1][i]+" enters into
the Critical Section having a timestamp of "+timestamp[0][i]);
                System.out.print("The list of node/nodes which is/are waiting
to enter into Critical Section are: ");
                //The last node in the queue has entered into Critical
Section, hence no node are waiting to enter into CS
                if(i==no_of_nodes-1)
                     System.out.print("None");
                else
                {
                    for(j=i+1;j<no_of_nodes;j++)
                    {
                        System.out.print("Node"+timestamp[1][j]+"\t");
                    }
                }
                System.out.println("\nThe node"+timestamp[1][i]+" comes out of
the Critical Section");
                timestamp[0][i]=0;
                i++;
            }
        }
    }

    public static void main(String args[])
    {
        Scanner scanner=new Scanner(System.in);
        int n;
        System.out.print("\nEnter the number of nodes : ");
        n=scanner.nextInt();
        if(n<2)
        {
            System.out.println("Invalid Input!");
            scanner.close();
            return;
        }
```

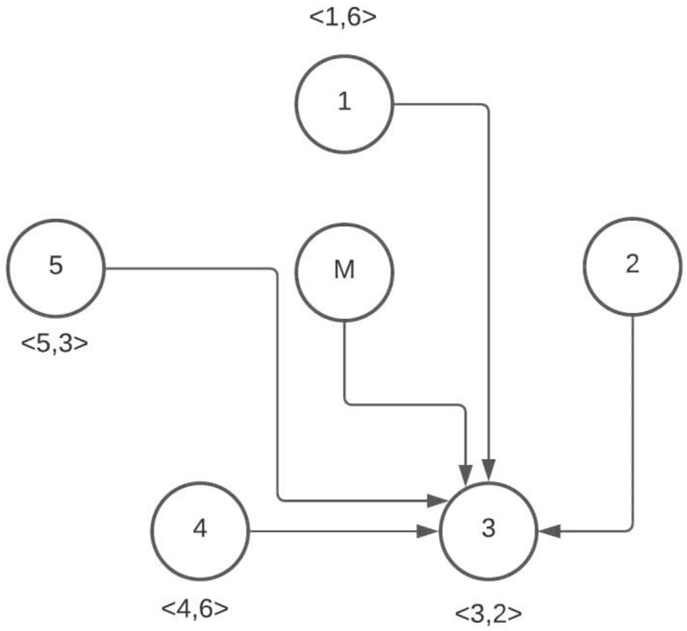


```
        RicartAgrawala obj = new RicartAgrawala(n);

        obj.getRequest();

        obj.giveResponse();
        scanner.close();
    }
}
```

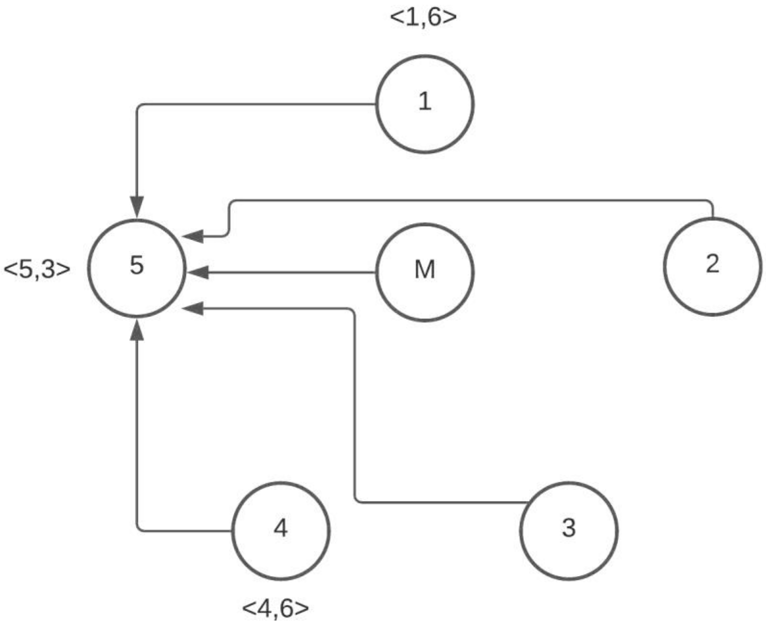
Dataset Used



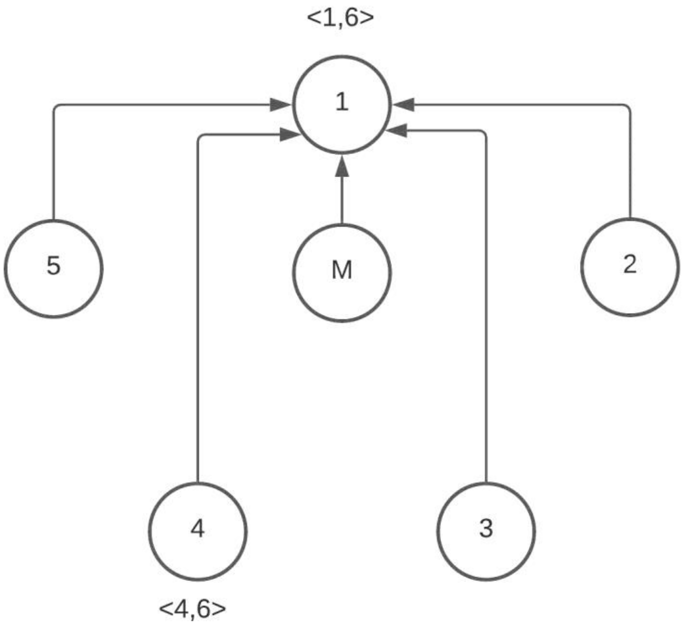
Here, the node M is in Critical Section. The node 1 with a timestamp of 6 units, node 3 with a timestamp of 2 units, node 4 with a timestamp of 6 units and node 5 with a timestamp of 3 units are waiting to enter into the Critical Section. So, each node sends a request to the node M and the remaining n-1 nodes. As shown in the above figure, node 1 sends a request to all the remaining nodes. The nodes 3, 4 and 5 sends request in the similar fashion.



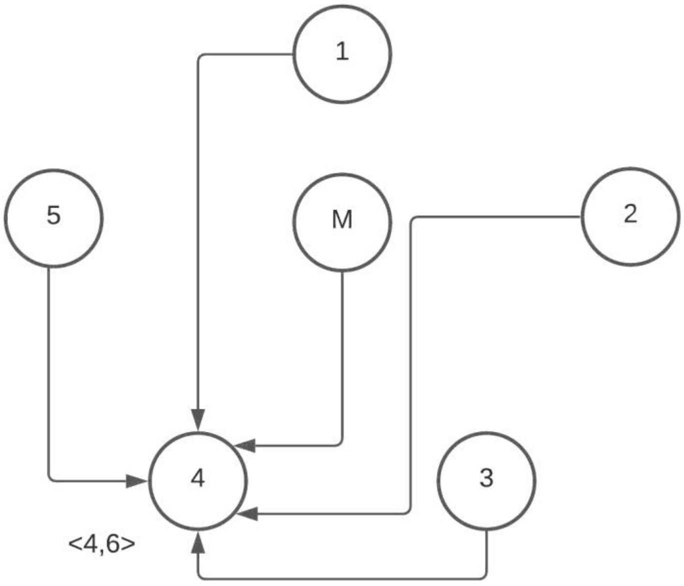
After the node M comes out of the Critical Section, the list of nodes which are waiting to enter into the Critical Section are 1, 3, 4 and 5. It is seen that the node 3 has the least timestamp. Hence, the node 3 receives a reply from all the nodes in the distributed system. Thus, the node 3 is the next node to be in the Critical Section.



After the node 3 comes out of the Critical Section, the list of nodes which are waiting to enter into the Critical Section are 1, 4 and 5. It is seen that the node 5 has the least timestamp. Hence, the node 5 receives a reply from all the nodes in the distributed system. Thus, the node 5 is the next node to be in the Critical Section.

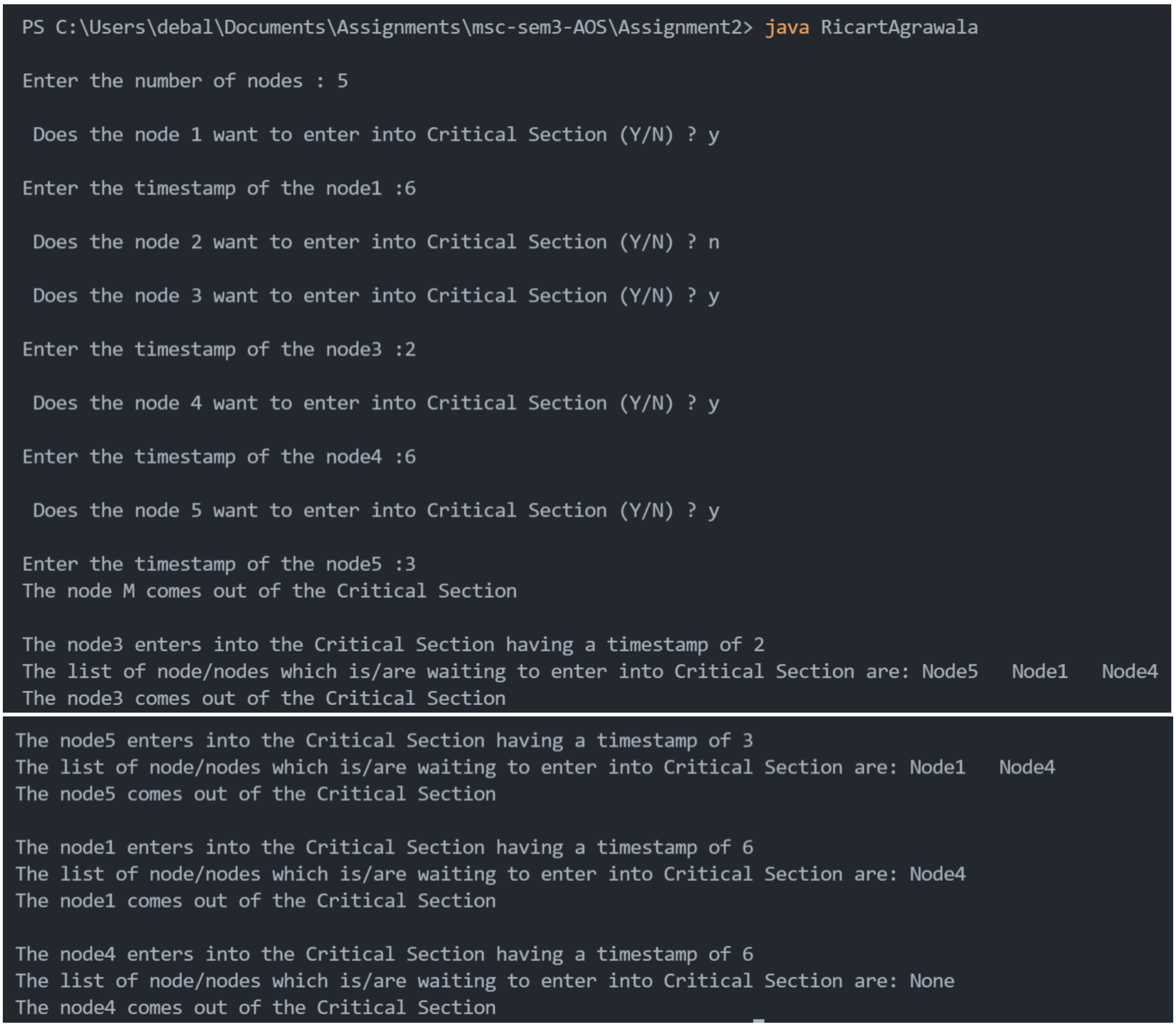


After the node 5 comes out of the Critical Section, the list of nodes which are waiting to enter into the Critical Section are 1 and 4. It is seen that both nodes 1 and 4 have the same timestamp. Hence, the node 1 whose ID is least (1<4) receives a reply from all the nodes in the distributed system. Thus, the node 1 is the next node to be in the Critical Section.



After the node 1 comes out of the Critical Section, only the node 4 is waiting to enter into the Critical Section which receives a reply from all the remaining nodes as shown in the above figure. Thus, the node 4 is the last node to be in the Critical Section.

Output Obtained:



```
PS C:\Users\debal\Documents\Assignments\msc-sem3-AOS\Assignment2> java RicartAgrawala

Enter the number of nodes : 5

 Does the node 1 want to enter into Critical Section (Y/N) ? y

Enter the timestamp of the node1 :6

 Does the node 2 want to enter into Critical Section (Y/N) ? n

 Does the node 3 want to enter into Critical Section (Y/N) ? y

Enter the timestamp of the node3 :2

 Does the node 4 want to enter into Critical Section (Y/N) ? y

Enter the timestamp of the node4 :6

 Does the node 5 want to enter into Critical Section (Y/N) ? y

Enter the timestamp of the node5 :3
The node M comes out of the Critical Section

The node3 enters into the Critical Section having a timestamp of 2
The list of node/nodes which is/are waiting to enter into Critical Section are: Node5   Node1   Node4
The node3 comes out of the Critical Section
The node5 enters into the Critical Section having a timestamp of 3
The list of node/nodes which is/are waiting to enter into Critical Section are: Node1   Node4
The node5 comes out of the Critical Section

The node1 enters into the Critical Section having a timestamp of 6
The list of node/nodes which is/are waiting to enter into Critical Section are: Node4
The node1 comes out of the Critical Section

The node4 enters into the Critical Section having a timestamp of 6
The list of node/nodes which is/are waiting to enter into Critical Section are: None
The node4 comes out of the Critical Section
```