# Assignment 3

## Assumptions

1. It is assumed that the network given by the user is minimally connected.
2. It is assumed that within the given network there is exactly one node which is already in the Critical Section.
3. The node numbers range from 1 to n, where n is the total number of nodes in the given network which is given by the user.
4. The number of nodes wanting to enter into the Critical Section is given by the user. The nodes are however chosen randomly.

## Source Code

```
/* It is assumed that the network which is given by the user is connected. */

import java.util.Scanner;
import java.util.Queue;
import java.util.LinkedList;
import java.util.Random;
import java.util.Set;
import java.util.HashSet;

class Raymond
{
    // The total number of nodes in the network is stored in no_of_nodes
    // Each node is pointing to another node (parent) which is stored in
holder
    // Each node is assigned a node number. (Eg : Node P1 has a node_number of
1, Node P2 has a node_number of 2, etc)
    int no_of_nodes, holder, node_number;
    // Each node is assigned a queue.
    Queue<Integer> q;

    public Raymond(int n)
    {
        this.no_of_nodes = n;
        this.q = new LinkedList<Integer>();
        this.holder = 0;
        this.node_number = 0;
    }

    public void inputHolder(int node)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("\nEnter the holder of the node P"+node+" : ");
        this.holder=scanner.nextInt();
    }
```

```
    public void inputNodeNumber(int node_number)
    {
        this.node_number = node_number;
    }

    public void displayHolder()
    {
        System.out.print(this.holder);
    }

    public void getRequest(Raymond obj[], int node_in)
    {
        //The request of entering into the CS has been informed to the node
which is currently in CS
        if(this.node_number == this.holder)
        {
            this.q.add(node_in);
            return;
        }

        //This node is not currently in CS. However a message has been sent to
the priviledged node before as it is seen that the respective queue is non
empty.
        else if(this.q.size() != 0)
        {
            this.q.add(node_in);
            return;
        }

        // This node is not currently in CS. A message will have to be
transfered to the holder node as it is seen that the respective queue is
empty.
        else
        {
            this.q.add(node_in);
            obj[this.holder-1].getRequest(obj, this.node_number);
        }
    }

    public void givePriviledge(Raymond obj[])
    {
        int node;
        node = this.q.remove();

        //This node will be given the priviledge of entering into the CS.
        if(node == this.node_number)
        {
```

```
            System.out.println("\nThe node P"+node+" enters into the Critical
Section");
            System.out.println("The contents of the queue of each node are :
");
            for(int i=0; i<no_of_nodes; i++)
            {
                System.out.println("Node P"+(i+1)+" : "+obj[i].q);
            }
            System.out.println("The node P"+node+" comes out of the Critical
Section");

            //All the nodes wanting to enter into CS have been given the
priviledge.
            if(obj[node-1].q.size() == 0)
                return;

            //Some nodes are waiting to enter into CS.
            else
                obj[node-1].givePriviledge(obj);
        }

        //As the queue of the respective node is non-empty, a message/request
will have to be send to the next node.
        else if(this.q.size() !=0 )
        {
            obj[node-1].q.add(this.node_number);
            obj[node-1].givePriviledge(obj);
        }

        //As the queue of the respective node is empty, no message/request
will have to be send to the next node.
        else
        {
            obj[node-1].givePriviledge(obj);
        }
    }

    public static void main(String args[])
    {
        int n,i,node_in_cs=0, flag=0;
        Raymond []obj;

        Scanner scanner = new Scanner(System.in);
        System.out.print("\nEnter the number of nodes : ");
        n=scanner.nextInt();
        // Creating an array on n objects where each object represents each
node.
        obj = new Raymond[n];
```

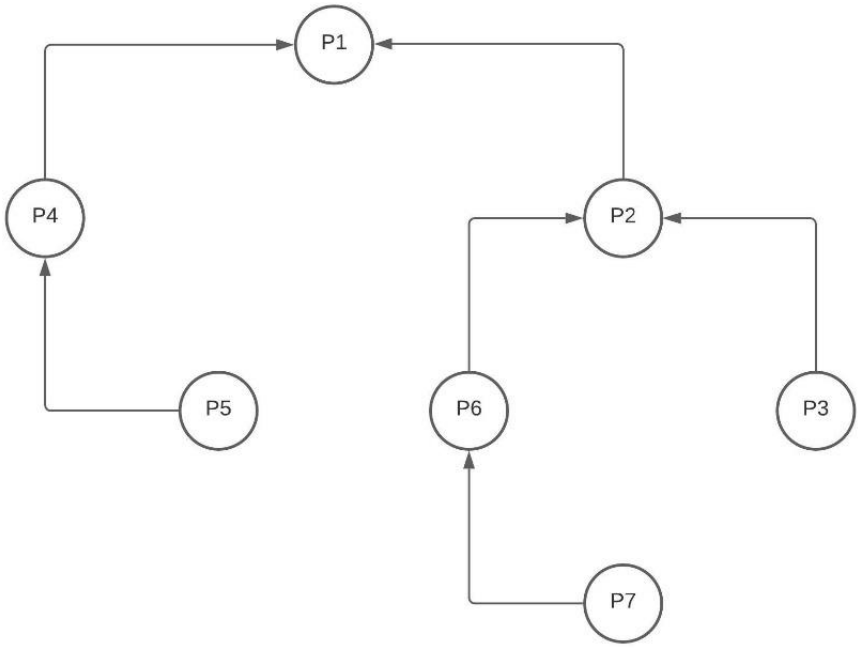
```
        for(i=0;i<n;i++)
        {
            obj[i] = new Raymond(n);
        }

        // To input the holders of each node which is given by the user and to
generate the node number for each node.
        for(i=0;i<n;i++)
        {
            obj[i].inputHolder(i+1);
            obj[i].inputNodeNumber(i+1);
        }

        //To check which node is in Critical Section.
        for(i=0; i<n; i++)
        {
            if(obj[i].holder == obj[i].node_number)
            {
                node_in_cs = i;
                flag = 1;
                break;
            }
        }

        if(flag == 0)
        {
            System.out.println("The given network is invalid as no node is in
Critical Section!");
            return;
        }

        //To display the holders of each node
        for(i=0;i<n;i++)
        {
            System.out.print("\nThe holder of node P"+(i+1)+" is ");
            obj[i].displayHolder();
        }

        Random rand = new Random();
        int rand_cs=0, node_in, no_of_inputs=0, rand_node;
        char ans;
        Set<Integer> hs = new HashSet<Integer>();

        System.out.print("\n\nEnter the number of nodes which wants to enter
into Critical Section : ");
        no_of_inputs = scanner.nextInt();
```

```
        // Since one node is already in CS, the maximum number of nodes
wanting to enter into CS can be n-1
        if(no_of_inputs>n-1 || no_of_inputs<1)
        {
            System.out.println("Invalid Input!");
            scanner.close();
            System.exit(0);
        }

        // Since a set stores unique values, we will generate nodes wanting to
enter into CS, till the size of the hashset becomes the size of the no of
inputs
        while(hs.size()!=no_of_inputs)
        {
            rand_node=0;
            // The node number can neither be 0 nor the node which is already
in CS.
            while(rand_node==0 || rand_node==node_in_cs)
                rand_node = rand.nextInt(n);
            hs.add(rand_node);
        }

        // Each node from the set requests for CS using the getRequest()
funtion.
        System.out.print("The nodes which are wanting to enter into the
Critical Section are : ");
        for (int node : hs)
        {
            System.out.print("P"+node+" ");
            obj[node-1].getRequest(obj, node);
        }
        System.out.println();
        System.out.println("The node currently in Critical Section is
P"+obj[node_in_cs].node_number);

        //Each node wanting to be in CS is given priviledge using the function
givePriviledge()
        obj[node_in_cs].givePriviledge(obj);
    }
}
```

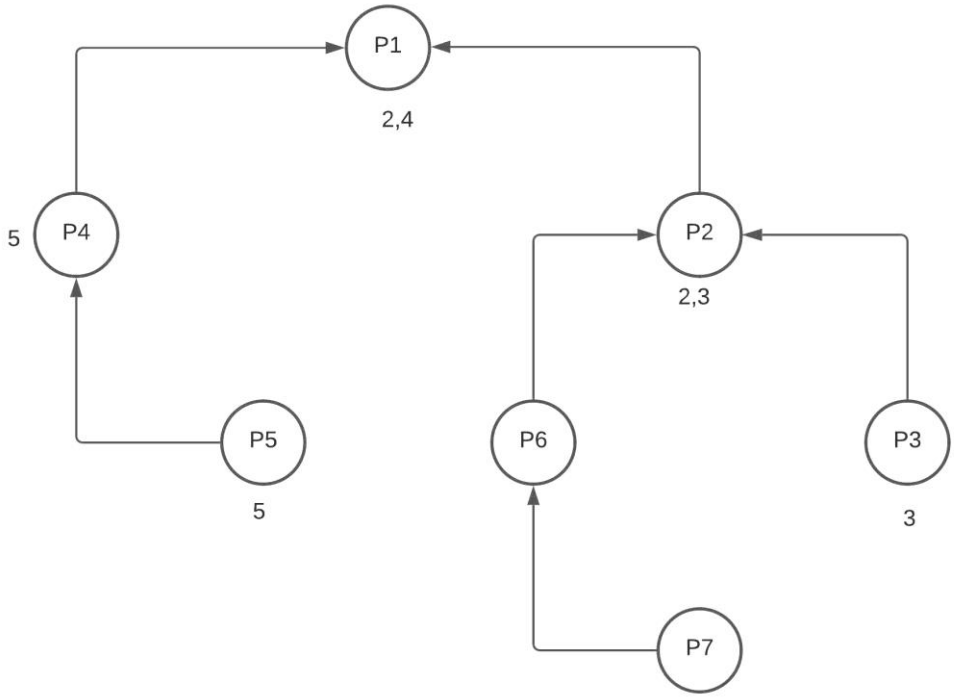
## Dataset Used

## SET 1

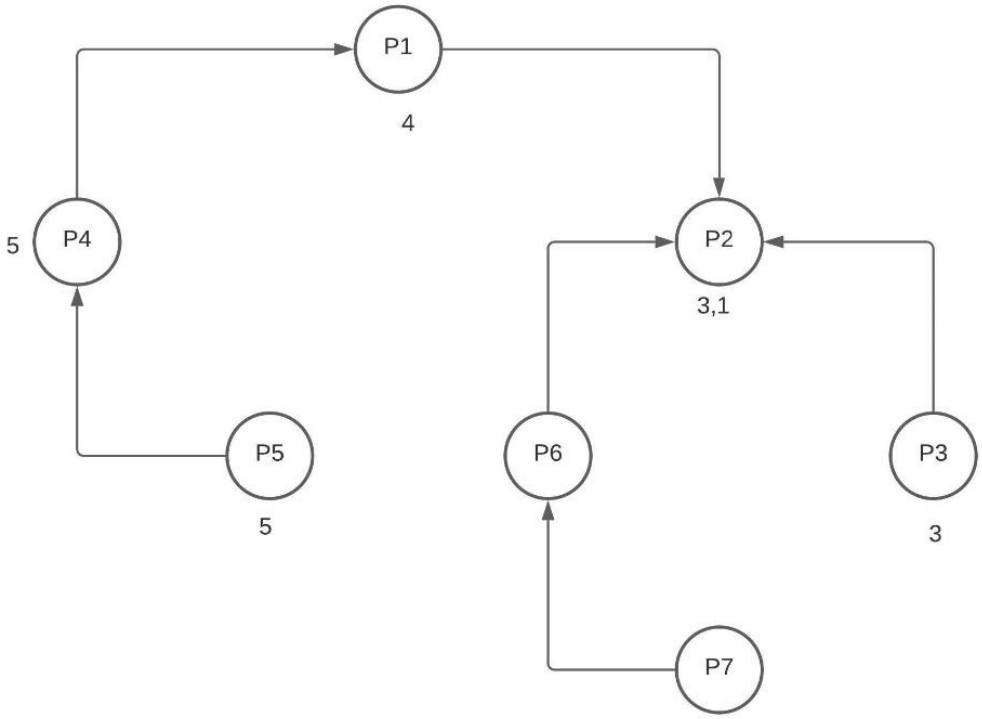
Initial Network as given by the user.



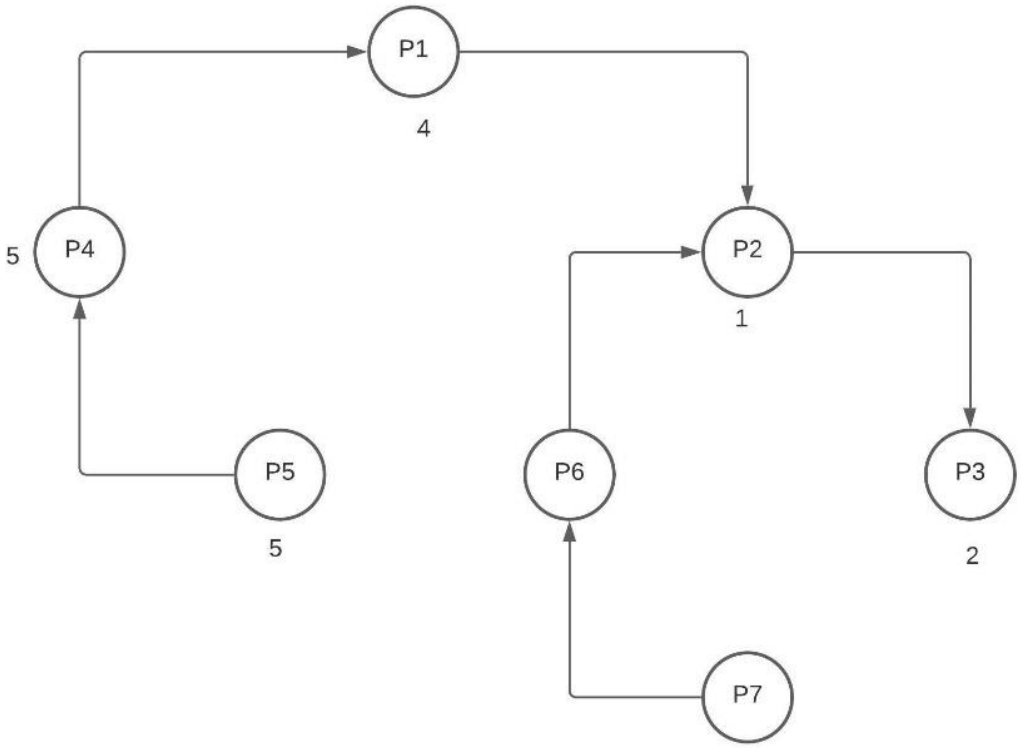

Node P2, P3 and P5 requests for Critical Section.



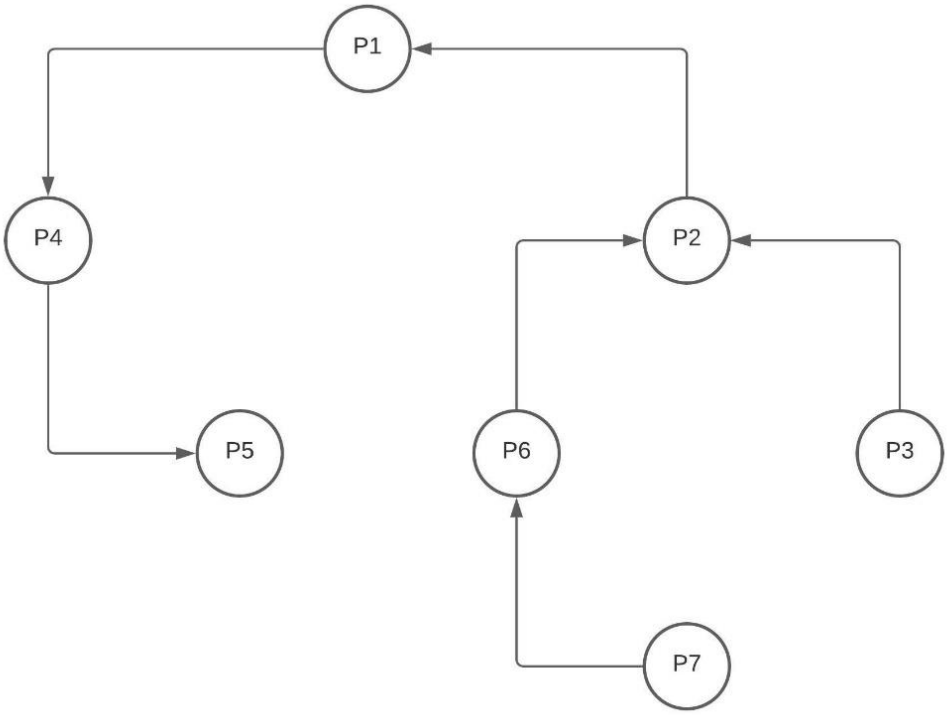
Node P1 comes out of Critical Section and node P2 enters into Critical Section




Node P2 comes out of Critical Section and node P3 enters into Critical Section

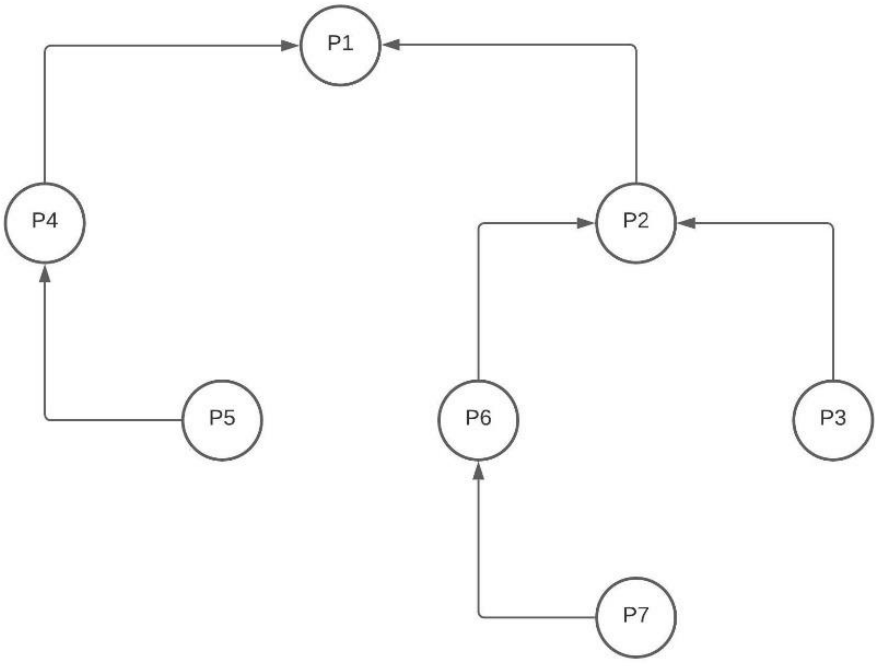


Node P3 comes out of Critical Section and node P5 enters into Critical Section

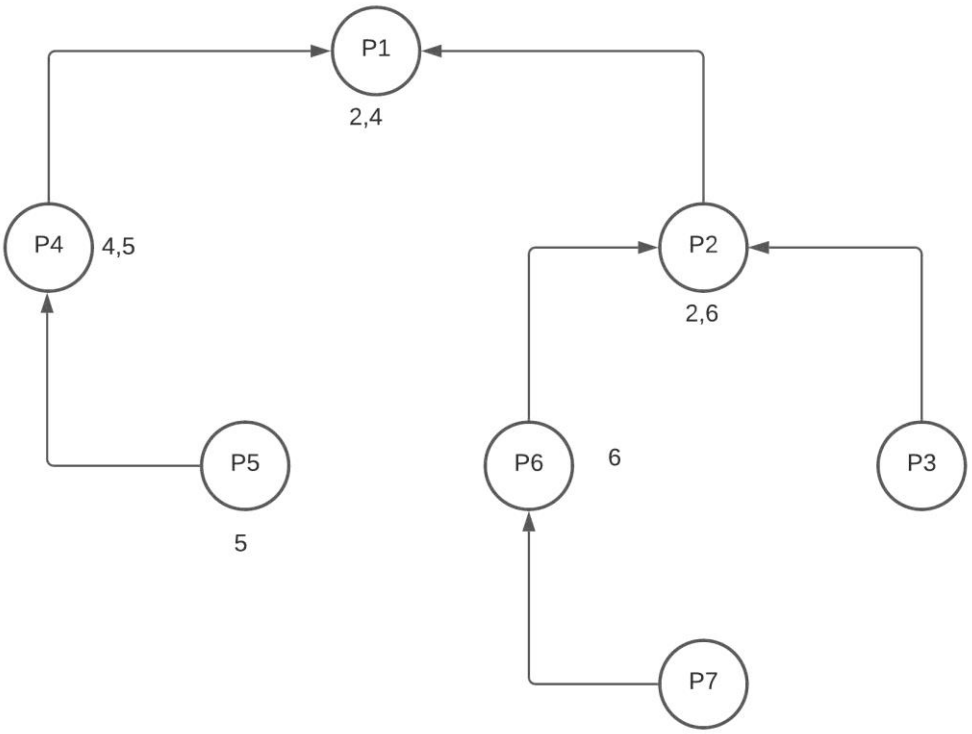



**SET 2**

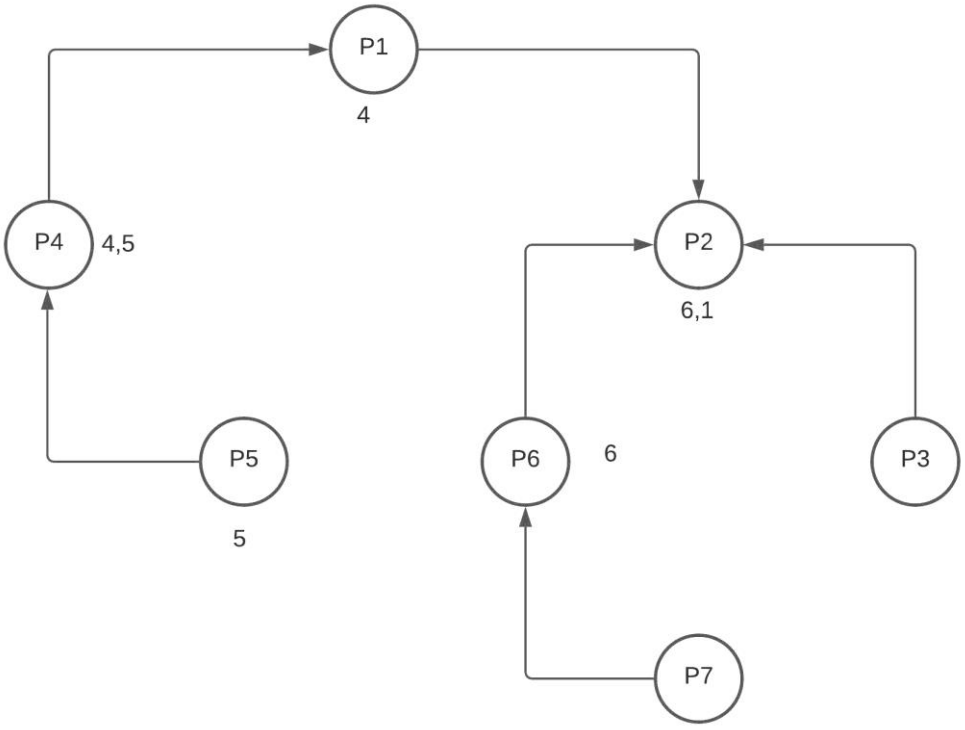
Initial Network as given by the user.



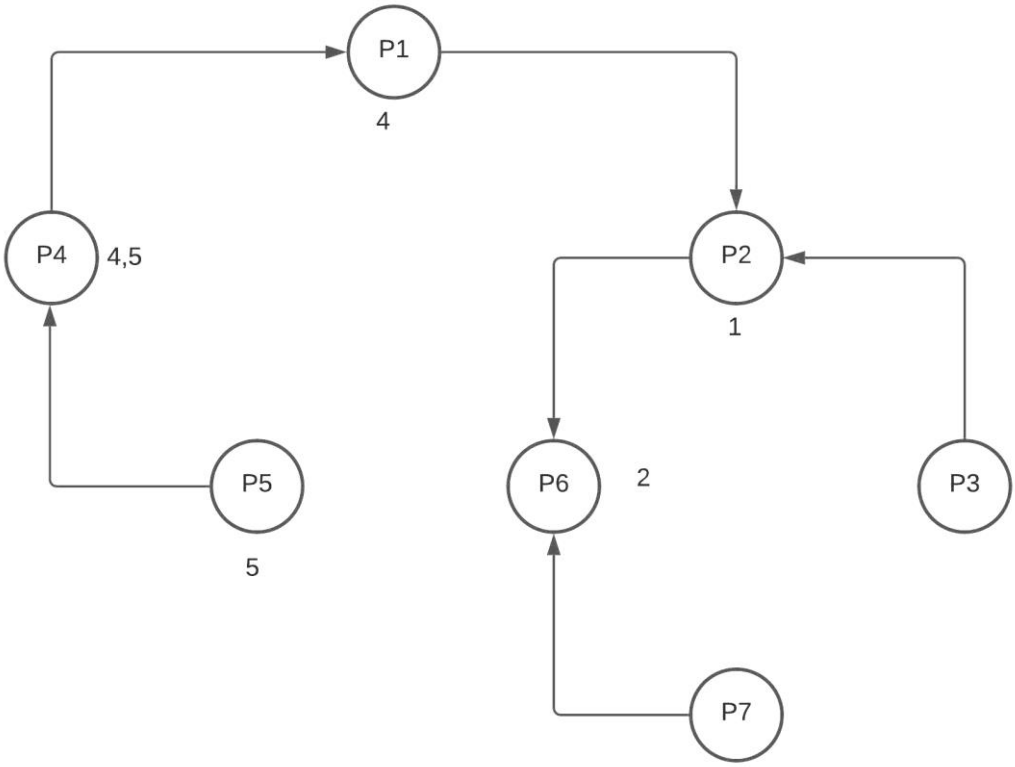
Node P2, P4, P5 and P6 requests for Critical Section.



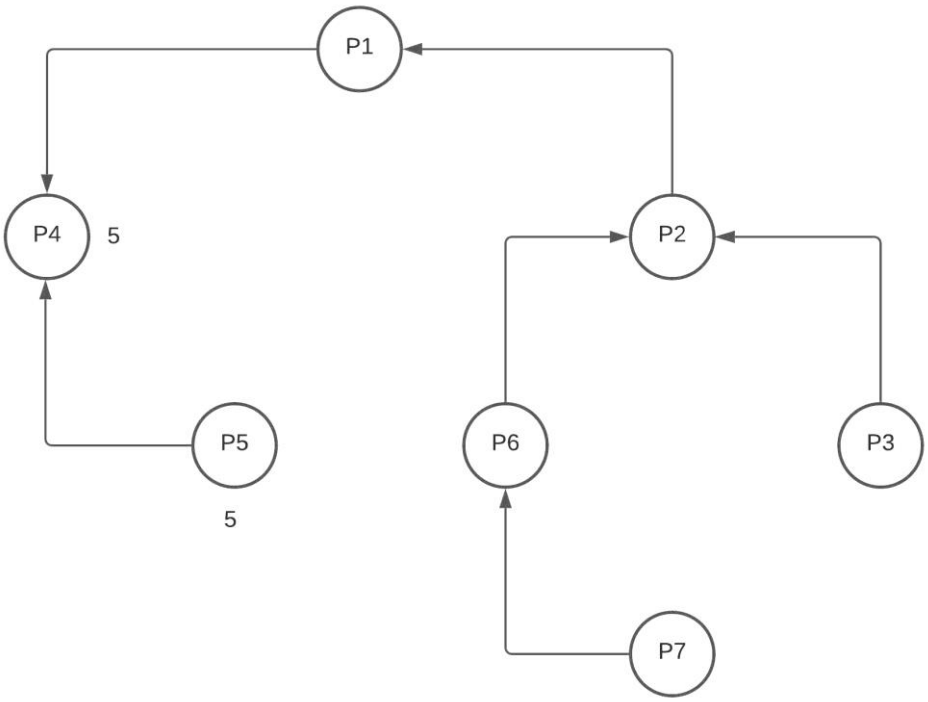

Node P1 comes out of Critical Section and node P2 enters into Critical Section



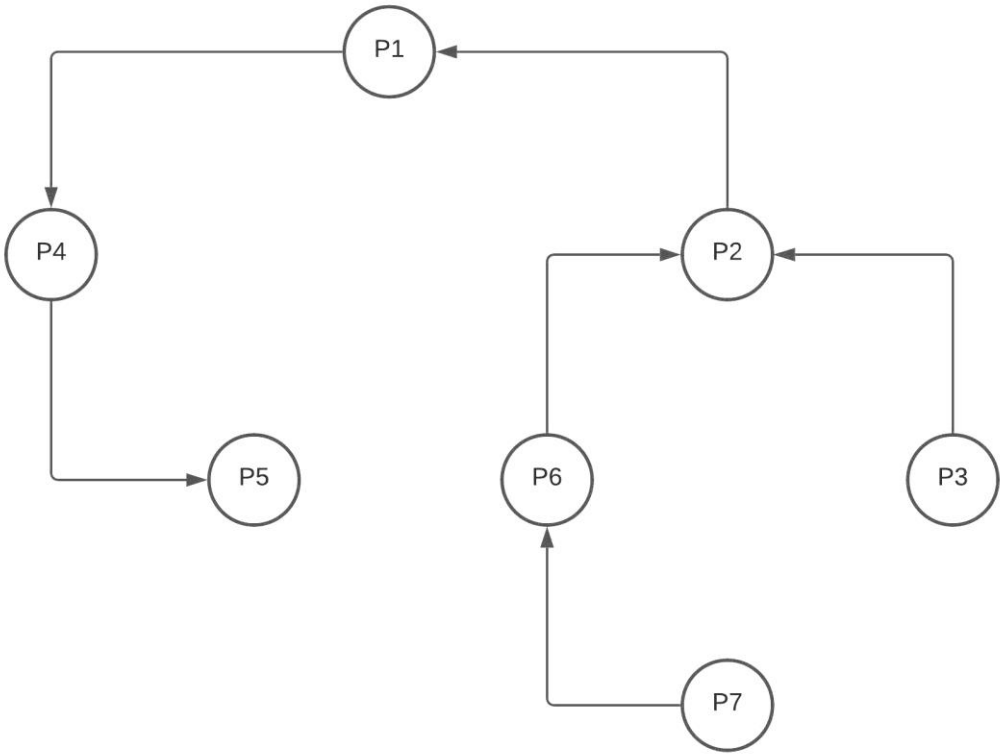
Node P2 comes out of Critical Section and node P6 enters into Critical Section




Node P6 comes out of Critical Section and node P4 enters into Critical Section



Node P4 comes out of Critical Section and node P5 enters into Critical Section



## Output Obtained

## SET 1

```
PS C:\Users\debal\Documents\Assignments\msc-sem3-AOS\Assignment3> java Raymond

Enter the number of nodes : 7

Enter the holder of the node P1 : 1

Enter the holder of the node P2 : 1

Enter the holder of the node P3 : 2

Enter the holder of the node P4 : 1

Enter the holder of the node P5 : 4

Enter the holder of the node P6 : 2

Enter the holder of the node P7 : 6

The holder of node P1 is 1
The holder of node P2 is 1
The holder of node P3 is 2
The holder of node P4 is 1
The holder of node P5 is 4
The holder of node P6 is 2
The holder of node P7 is 6

Enter the number of nodes which wants to enter into Critical Section : 3
The nodes which are wanting to enter into the Critical Section are : P2 P3 P5
The node currently in Critical Section is P1
```

```
The node P2 enters into the Critical Section
The contents of the queue of each node are :
Node P1 : [4]
Node P2 : [3, 1]
Node P3 : [3]
Node P4 : [5]
Node P5 : [5]
Node P6 : []
Node P7 : []
The node P2 comes out of the Critical Section

The node P3 enters into the Critical Section
The contents of the queue of each node are :
Node P1 : [4]
Node P2 : [1]
Node P3 : [2]
Node P4 : [5]
Node P5 : [5]
Node P6 : []
Node P7 : []
The node P3 comes out of the Critical Section

The node P5 enters into the Critical Section
The contents of the queue of each node are :
Node P1 : []
Node P2 : []
Node P3 : []
Node P4 : []
Node P5 : []
Node P6 : []
Node P7 : []
The node P5 comes out of the Critical Section
PS C:\Users\debal\Documents\Assignments\msc-sem3-AOS\Assignment3>
```

## SET 2

```
PS C:\Users\debal\Documents\Assignments\msc-sem3-AOS\Assignment3> java Raymond

Enter the number of nodes : 7

Enter the holder of the node P1 : 1

Enter the holder of the node P2 : 1

Enter the holder of the node P3 : 2

Enter the holder of the node P4 : 1

Enter the holder of the node P5 : 4

Enter the holder of the node P6 : 2

Enter the holder of the node P7 : 6

The holder of node P1 is 1
The holder of node P2 is 1
The holder of node P3 is 2
The holder of node P4 is 1
The holder of node P5 is 4
The holder of node P6 is 2
The holder of node P7 is 6

Enter the number of nodes which wants to enter into Critical Section : 4
The nodes which are wanting to enter into the Critical Section are : P2 P4 P5 P6
The node currently in Critical Section is P1
```

```
The node P2 enters into the Critical Section
The contents of the queue of each node are :
Node P1 : [4]
Node P2 : [6, 1]
Node P3 : []
Node P4 : [4, 5]
Node P5 : [5]
Node P6 : [6]
Node P7 : []
The node P2 comes out of the Critical Section

The node P6 enters into the Critical Section
The contents of the queue of each node are :
Node P1 : [4]
Node P2 : [1]
Node P3 : []
Node P4 : [4, 5]
Node P5 : [5]
Node P6 : [2]
Node P7 : []
The node P6 comes out of the Critical Section

The node P4 enters into the Critical Section
The contents of the queue of each node are :
Node P1 : []
Node P2 : []
Node P3 : []
Node P4 : [5]
Node P5 : [5]
Node P6 : []
Node P7 : []
The node P4 comes out of the Critical Section
```

```
The node P5 enters into the Critical Section
The contents of the queue of each node are :
Node P1 : []
Node P2 : []
Node P3 : []
Node P4 : []
Node P5 : []
Node P6 : []
Node P7 : []
The node P5 comes out of the Critical Section
PS C:\Users\debal\Documents\Assignments\msc-sem3-AOS\Assignment3>
```