Program 1

Aim: Write a program to implement Client Server using RPC

ClientRPC.java

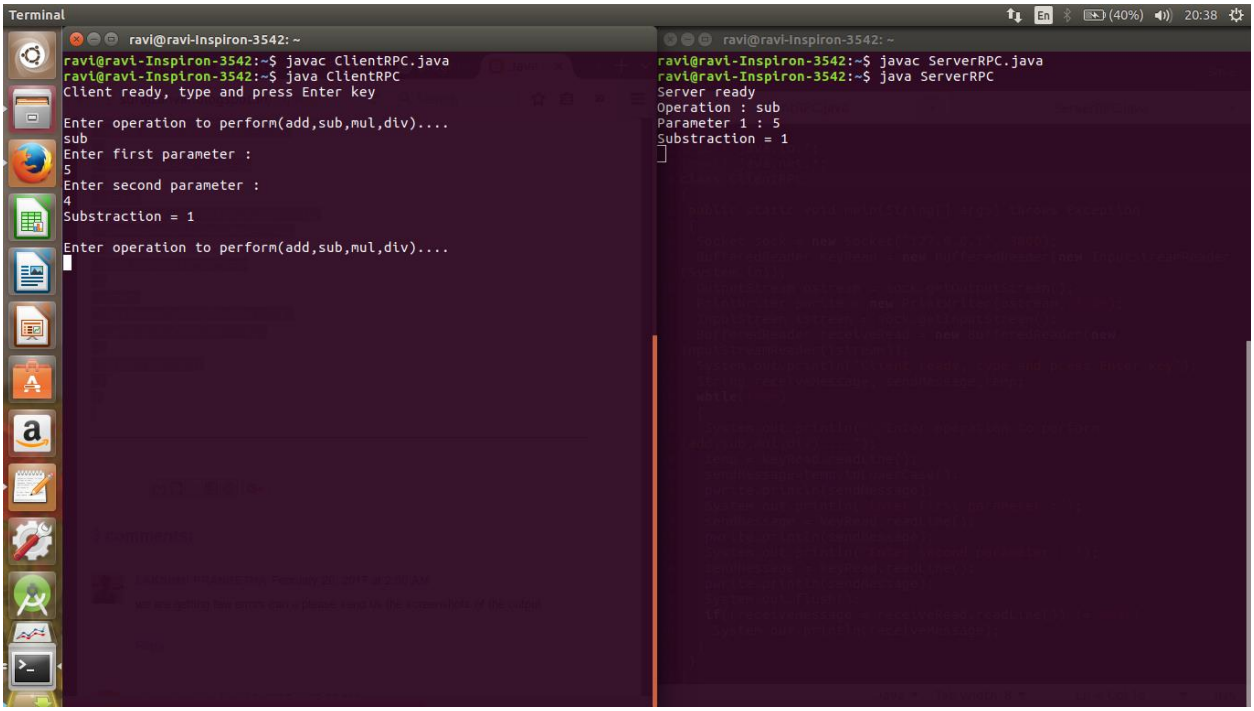
```
import java.io.*;
import java.net.*;
class ClientRPC
{
        public static void main(String[] args) throws Exception
        {
            Socket sock = new Socket("127.0.0.1", 3000);
            BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
            OutputStream ostream = sock.getOutputStream();
            PrintWriter pwrite = new PrintWriter(ostream, true);
            InputStream istream = sock.getInputStream();
            BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));
            System.out.println("Client ready, type and press Enter key");
            String receiveMessage, sendMessage,temp;
            while(true)
            {
                System.out.println("\nEnter operation to perform(add,sub,mul,div)....");
                temp = keyRead.readLine();
                sendMessage=temp.toLowerCase();
                pwrite.println(sendMessage);
                System.out.println("Enter first parameter :");
                sendMessage = keyRead.readLine();
                pwrite.println(sendMessage);
                System.out.println("Enter second parameter : ");
                sendMessage = keyRead.readLine();
                pwrite.println(sendMessage);
                System.out.flush();
                if((receiveMessage = receiveRead.readLine()) != null)
                        System.out.println(receiveMessage);
            }
        }
}
```

ServerRPC.java

```
import java.io.*;
import java.net.*;
class ServerRPC
{
	public static void main(String[] args) throws Exception
	{
		ServerSocket sersock = new ServerSocket(3000);
		System.out.println("Server ready");
		Socket sock = sersock.accept( );
		BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
		OutputStream ostream = sock.getOutputStream();
		PrintWriter pwrite = new PrintWriter(ostream, true);
		InputStream istream = sock.getInputStream();
		BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));
		String receiveMessage, sendMessage,fun;
		int a,b,c;
		while(true)
		{
			fun = receiveRead.readLine();
			if(fun != null)
				System.out.println("Operation : "+fun);
			a = Integer.parseInt(receiveRead.readLine());
			System.out.println("Parameter 1 : "+a);
			b = Integer.parseInt(receiveRead.readLine());
			if(fun.compareTo("add")==0)
			{
				c=a+b;
				System.out.println("Addition = "+c);
				pwrite.println("Addition = "+c);
			}
			if(fun.compareTo("sub")==0)
			{
				c=a-b;
				System.out.println("Substraction = "+c);
				pwrite.println("Substraction = "+c);
			}
			if(fun.compareTo("mul")==0)
			{
				c=a*b;
```

```
                System.out.println("Multiplication = "+c);
                pwrite.println("Multiplication = "+c);
            }
            if(fun.compareTo("div")==0)
            {
                c=a/b;
                System.out.println("Division = "+c);
                pwrite.println("Division = "+c);
            }
            System.out.flush();
        }
    }
}
```

OUTPUT



Program 2

Aim: Write a program to implement Client Server using RPC

ReceiveMessageInterface.java

```
 import java.rmi.*;
public interface ReceiveMessageInterface extends Remote
{
      void receiveMessage(String x) throws RemoteException;
}
```

RmiServer.java

```
import java.rmi.*;
import java.rmi.registry.*;
import java.rmi.server.*;
import java.net.*;
public class RmiServer extends java.rmi.server.UnicastRemoteObject implements
ReceiveMessageInterface{
  String address;
  Registry registry;

  public void receiveMessage(String x) throws RemoteException{
  System.out.println(x);
  }

  public RmiServer() throws RemoteException{
  try{
  address = (InetAddress.getLocalHost()).toString();
  }
  catch(Exception e){
  System.out.println("can't get inet address.");
  }
  int port=3232;
  System.out.println("this address=" + address +  ",port=" + port);
  try{
  registry = LocateRegistry.createRegistry(port);
  registry.rebind("rmiServer", this);
  }
```

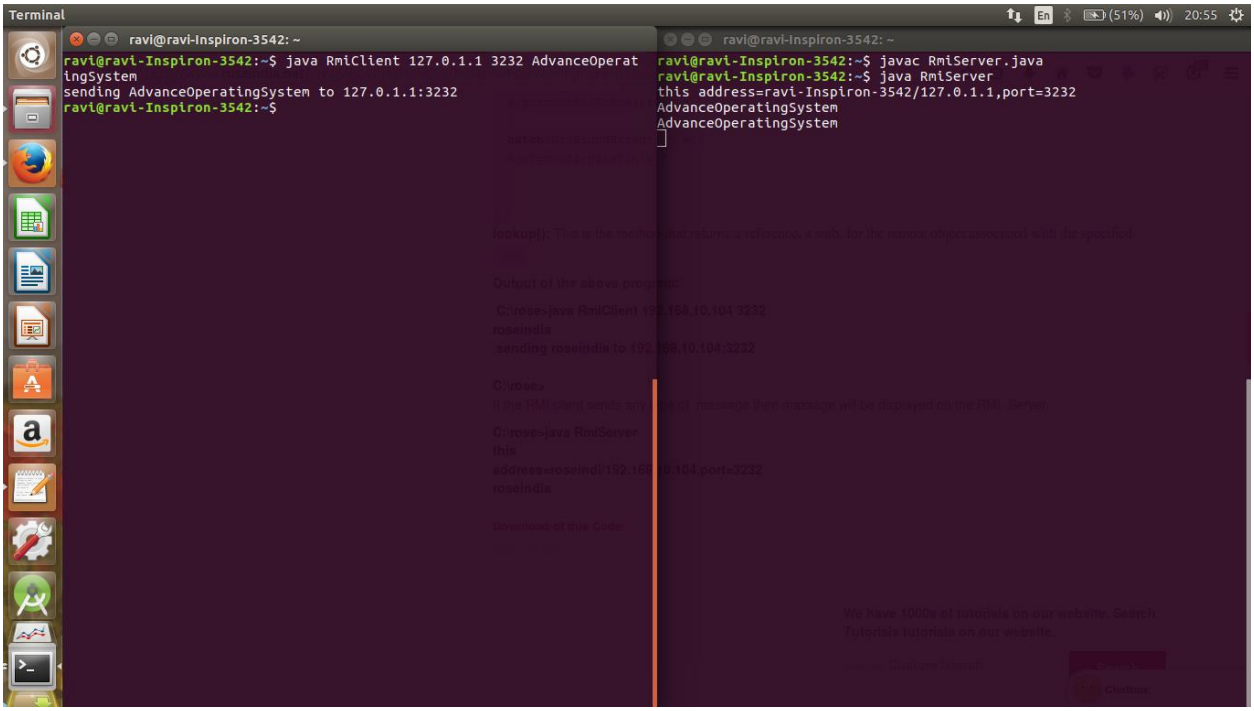
```
  catch(RemoteException e){
  System.out.println("remote exception"+ e);
  }
  }
  static public void main(String args[]){
  try{
  RmiServer server = new RmiServer();
  }
  catch (Exception e){
  e.printStackTrace();
  System.exit(1);
  }
  }
}
```

RmiClient.java

```
import java.rmi.*;
import java.rmi.registry.*;
import java.net.*;
public class RmiClient{
  static public void main(String args[]){
  ReceiveMessageInterface rmiServer;
  Registry registry;
  String serverAddress=args[0];
  String serverPort=args[1];
  String text=args[2];
  System.out.println
   ("sending " + text + " to " +serverAddress + ":" + serverPort);
  try{
  registry=LocateRegistry.getRegistry
  (serverAddress,(new Integer(serverPort)).intValue());
  rmiServer=(ReceiveMessageInterface)(registry.lookup("rmiServer"));
  // call the remote method
  rmiServer.receiveMessage(text);
  }
  catch(RemoteException e){
  e.printStackTrace();
  }
  catch(NotBoundException e){
  System.err.println(e);
```

```
  }
  }
}
```

OUTPUT


Terminal
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ java RmiClient 127.0.1.1 3232 AdvanceOperat
ingSystem
sending AdvanceOperatingSystem to 127.0.1.1:3232
ravi@ravi-Inspiron-3542:~$
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ javac RmiServer.java
ravi@ravi-Inspiron-3542:~$ java RmiServer
this address=ravi-Inspiron-3542/127.0.1.1,port=3232
AdvanceOperatingSystem
AdvanceOperatingSystem

Program 3

<u>Aim</u>: Write a program to implement mutlithread Client Server using process

Server.java

```
import java.io.*;
import java.net.*;
import java.lang.*;
public class Server {
   public static void main(String[] args) throws IOException {
         final int port = 8080;
         System.out.println("Server waiting for connection on port "+port);
         ServerSocket ss = new ServerSocket(port);
         Socket clientSocket = ss.accept();
         System.out.println("Recieved connection from "+clientSocket.getInetAddress()+" on
port "+clientSocket.getPort());
         RecieveFromClientThread recieve = new RecieveFromClientThread(clientSocket);
         Thread thread = new Thread(recieve);
         thread.start();

         SendToClientThread send = new SendToClientThread(clientSocket);
         Thread thread2 = new Thread(send);
         thread2.start();
   }
}

class RecieveFromClientThread implements Runnable
{
   Socket clientSocket=null;
   BufferedReader brBufferedReader = null;

   public RecieveFromClientThread(Socket clientSocket)
   {
         this.clientSocket = clientSocket;
   }
   public void run() {
         try{
```

```
        brBufferedReader=new
        BufferedReader(newInputStreamReader(this.clientSocket.getInputStream());
         String messageString;
         while(true){
         while((messageString = brBufferedReader.readLine())!= null){
                 if(messageString.equals("EXIT"))
                 {
                         break;
                 }
                 System.out.println("From Client: " + messageString);
                 System.out.println("Please enter something to send back to client..");
         }
         this.clientSocket.close();
         System.exit(0);
   }

   }
   catch(Exception ex){System.out.println(ex.getMessage());}
   }
}

class SendToClientThread implements Runnable
{
   PrintWriter pwPrintWriter;
   Socket clientSock = null;
   public SendToClientThread(Socket clientSock)
   {
         this.clientSock = clientSock;
   }
   public void run() {
         try{
          pwPrintWriter=new PrintWriter(new
OutputStreamWriter(this.clientSock.getOutputStream());
         while(true)
         {
                 String msgToClientString = null;
                 BufferedReader input = new BufferedReader(new
InputStreamReader(System.in));
                 msgToClientString = input.readLine();
```

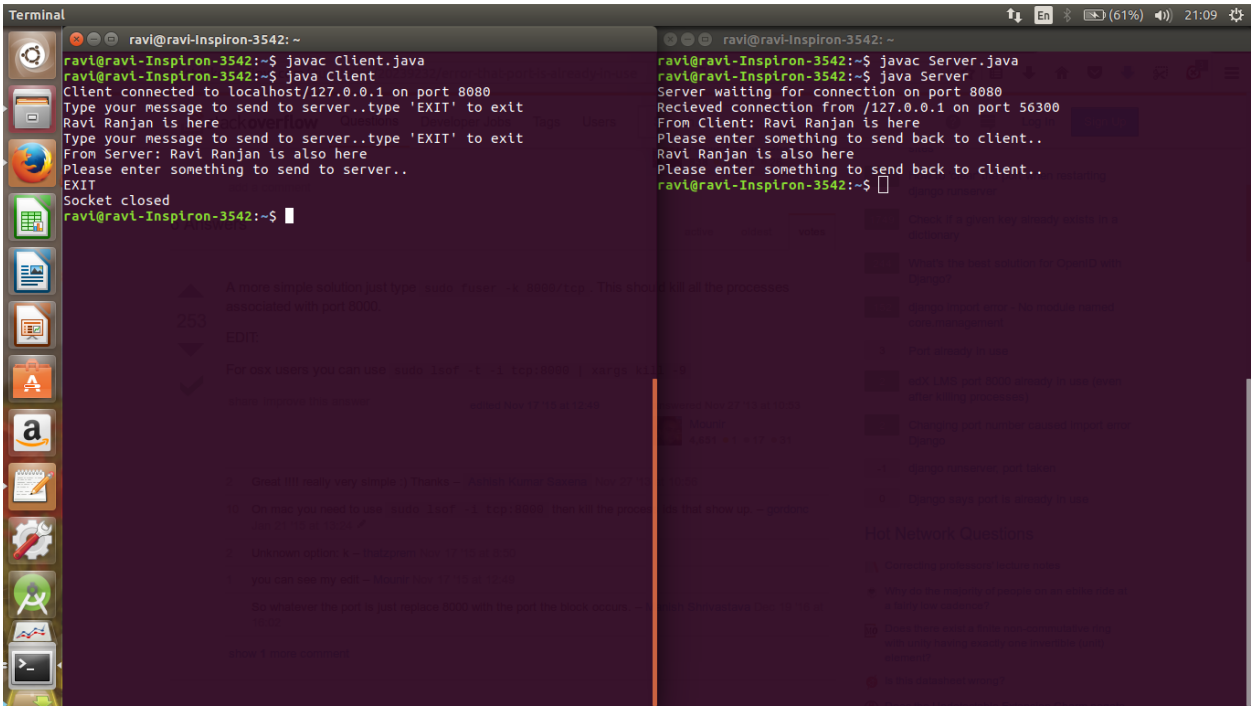
```
                pwPrintWriter.println(msgToClientString);
                pwPrintWriter.flush();
                System.out.println("Please enter something to send back to client..");
            }
        }
        catch(Exception ex){System.out.println(ex.getMessage());}
    }
}
```

Client.java

```
import java.io.*;
import java.net.*;
public class Client
{
        public static void main(String[] args)
        {
                try {
                        Socket sock = new Socket("localhost",8080);
                        SendThread sendThread = new SendThread(sock);
                        Thread thread = new Thread(sendThread);thread.start();
                        RecieveThread recieveThread = new RecieveThread(sock);
                        Thread thread2 =new Thread(recieveThread);thread2.start();
                } catch (Exception e) {System.out.println(e.getMessage());}
        }
}
class RecieveThread implements Runnable
{
        Socket sock=null;
        BufferedReader recieve=null;

        public RecieveThread(Socket sock) {
                this.sock = sock;
        }
        public void run()
        {
                try{
                recieve = new BufferedReader(new
InputStreamReader(this.sock.getInputStream()));
                String msgRecieved = null;
```

```
                while((msgRecieved = recieve.readLine())!= null)
                {
                        System.out.println("From Server: " + msgRecieved);
                        System.out.println("Please enter something to send to server..");
                }
            }catch(Exception e){System.out.println(e.getMessage());}
        }
}
class SendThread implements Runnable
{
        Socket sock=null;
        PrintWriter print=null;
        BufferedReader brinput=null;

        public SendThread(Socket sock)
        {
                this.sock = sock;
        }
        public void run(){
                try{
                if(sock.isConnected())
                {
                        System.out.println("Client connected to "+sock.getInetAddress() + " on
port "+sock.getPort());
                        this.print = new PrintWriter(sock.getOutputStream(), true);
                while(true){
                        System.out.println("Type your message to send to server..type 'EXIT' to
exit");
                        brinput = new BufferedReader(new InputStreamReader(System.in));
                        String msgtoServerString=null;
                        msgtoServerString = brinput.readLine();
                        this.print.println(msgtoServerString);
                        this.print.flush();

                        if(msgtoServerString.equals("EXIT"))
                        break;
                        }
                sock.close();}}catch(Exception e){System.out.println(e.getMessage());}
        }
}
```

OUTPUT


Terminal
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ javac Client.java
ravi@ravi-Inspiron-3542:~$ java Client
Client connected to localhost/127.0.0.1 on port 8080
Type your message to send to server..type 'EXIT' to exit
Ravi Ranjan is here
Type your message to send to server..type 'EXIT' to exit
From Server: Ravi Ranjan is also here
Please enter something to send to server..
EXIT
Socket closed
ravi@ravi-Inspiron-3542:~$
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ javac Server.java
ravi@ravi-Inspiron-3542:~$ java Server
Server waiting for connection on port 8080
Recieved connection from /127.0.0.1 on port 56300
From Client: Ravi Ranjan is here
Please enter something to send back to client..
Ravi Ranjan is also here
Please enter something to send back to client..
ravi@ravi-Inspiron-3542:~$

Program 4

Aim: Write a program to implement distributed chat server using TCP soket

Server.java

```
import java.io.*;
import java.net.*;
import java.lang.*;
public class Server {
   public static void main(String[] args) throws IOException {
         final int port = 8080;
         System.out.println("Server waiting for connection on port "+port);
         ServerSocket ss = new ServerSocket(port);
         Socket clientSocket = ss.accept();
         System.out.println("Recieved connection from "+clientSocket.getInetAddress()+" on
port "+clientSocket.getPort());
         RecieveFromClientThread recieve = new RecieveFromClientThread(clientSocket);
         Thread thread = new Thread(recieve);
         thread.start();

         SendToClientThread send = new SendToClientThread(clientSocket);
         Thread thread2 = new Thread(send);
         thread2.start();
   }
}
class RecieveFromClientThread implements Runnable
{
   Socket clientSocket=null;
   BufferedReader brBufferedReader = null;

   public RecieveFromClientThread(Socket clientSocket)
   {
         this.clientSocket = clientSocket;
   }
   public void run() {
         try{
         brBufferedReader = new BufferedReader(new
InputStreamReader(this.clientSocket.getInputStream()));
```

```
        String messageString;
        while(true){
        while((messageString = brBufferedReader.readLine())!= null){
                if(messageString.equals("EXIT"))
                {
                        break;
                }
                System.out.println("From Client: " + messageString);
                System.out.println("Please enter something to send back to client..");
        }
        this.clientSocket.close();
        System.exit(0);
    }

    }
    catch(Exception ex){System.out.println(ex.getMessage());}
    }
}

class SendToClientThread implements Runnable
{
    PrintWriter pwPrintWriter;
    Socket clientSock = null;

    public SendToClientThread(Socket clientSock)
    {
        this.clientSock = clientSock;
    }
    public void run() {
        try{
        pwPrintWriter =new PrintWriter(new
OutputStreamWriter(this.clientSock.getOutputStream()));

        while(true)
        {
                String msgToClientString = null;
                BufferedReader input = new BufferedReader(new
InputStreamReader(System.in));
```

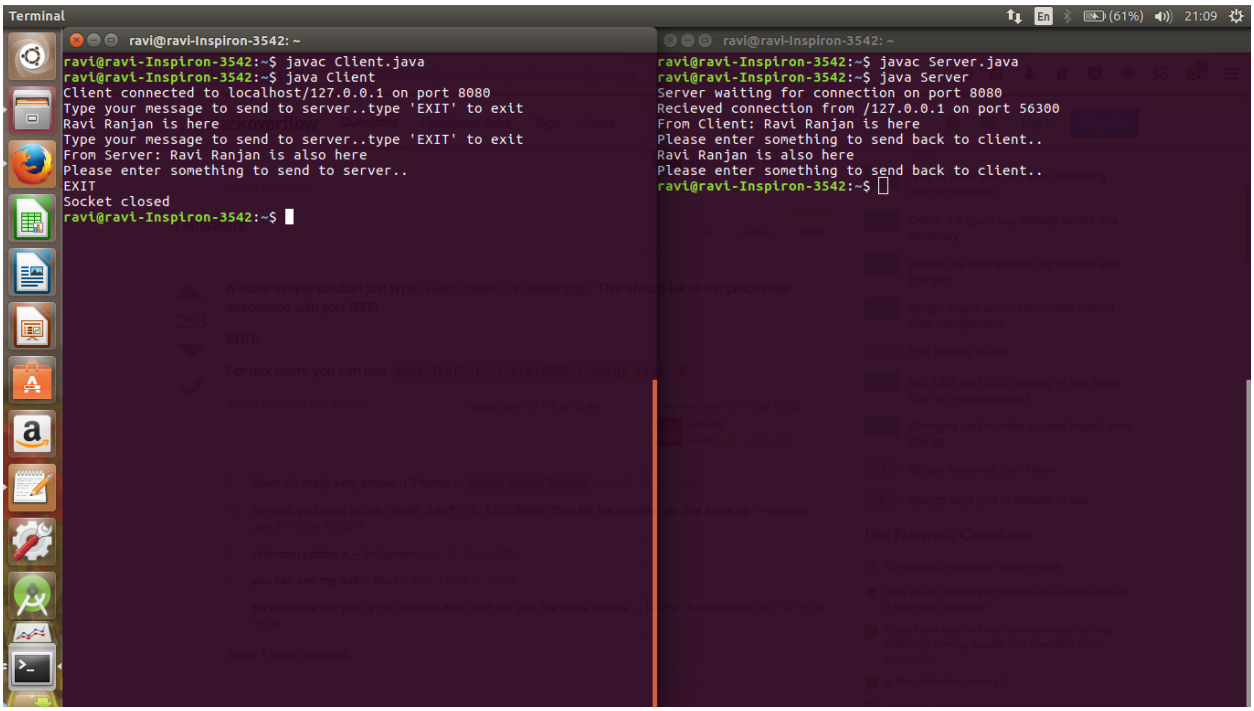
```
                msgToClientString = input.readLine();

                pwPrintWriter.println(msgToClientString);
                pwPrintWriter.flush();
                System.out.println("Please enter something to send back to client..");
        }
        }
        catch(Exception ex){System.out.println(ex.getMessage());}
    }
}
```

Client.java

```
import java.io.*;
import java.net.*;
public class Client {
        public static void main(String[] args)
        {
                try {
                        Socket sock = new Socket("localhost",8080);
                        SendThread sendThread = new SendThread(sock);
                        Thread thread = new Thread(sendThread);thread.start();
                        RecieveThread recieveThread = new RecieveThread(sock);
                        Thread thread2 =new Thread(recieveThread);thread2.start();
                } catch (Exception e) {System.out.println(e.getMessage());}
        }
}
class RecieveThread implements Runnable
{
        Socket sock=null;
        BufferedReader recieve=null;

        public RecieveThread(Socket sock) {
                this.sock = sock;
        }
        public void run() {
                try{
                recieve = new BufferedReader(new
InputStreamReader(this.sock.getInputStream()));
                String msgRecieved = null;
```

```
		while((msgRecieved = recieve.readLine())!= null)
		{
			System.out.println("From Server: " + msgRecieved);
			System.out.println("Please enter something to send to server..");
		}
		}catch(Exception e){System.out.println(e.getMessage());}
	}
}
class SendThread implements Runnable
{
	Socket sock=null;
	PrintWriter print=null;
	BufferedReader brinput=null;

	public SendThread(Socket sock)
	{
		this.sock = sock;
	}
	public void run(){
		try{
		if(sock.isConnected())
		{
			System.out.println("Client connected to "+sock.getInetAddress() + " on
port "+sock.getPort());
			this.print = new PrintWriter(sock.getOutputStream(), true);
		while(true){
			System.out.println("Type your message to send to server..type 'EXIT' to
exit");
			brinput = new BufferedReader(new InputStreamReader(System.in));
			String msgtoServerString=null;
			msgtoServerString = brinput.readLine();
			this.print.println(msgtoServerString);
			this.print.flush();

			if(msgtoServerString.equals("EXIT"))
			break;
			}
		sock.close();}}catch(Exception e){System.out.println(e.getMessage());}
	}
}
```

Output


Terminal
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ javac Client.java
ravi@ravi-Inspiron-3542:~$ java Client
Client connected to localhost/127.0.0.1 on port 8080
Type your message to send to server..type 'EXIT' to exit
Ravi Ranjan is here
Type your message to send to server..type 'EXIT' to exit
From Server: Ravi Ranjan is also here
Please enter something to send to server..
EXIT
Socket closed
ravi@ravi-Inspiron-3542:~$
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ javac Server.java
ravi@ravi-Inspiron-3542:~$ java Server
Server waiting for connection on port 8080
Recieved connection from /127.0.0.1 on port 56300
From Client: Ravi Ranjan is here
Please enter something to send back to client..
Ravi Ranjan is also here
Please enter something to send back to client..
ravi@ravi-Inspiron-3542:~$

Program 6

Aim: Write a program to simulate the functioning of Lamport's Logical clock.

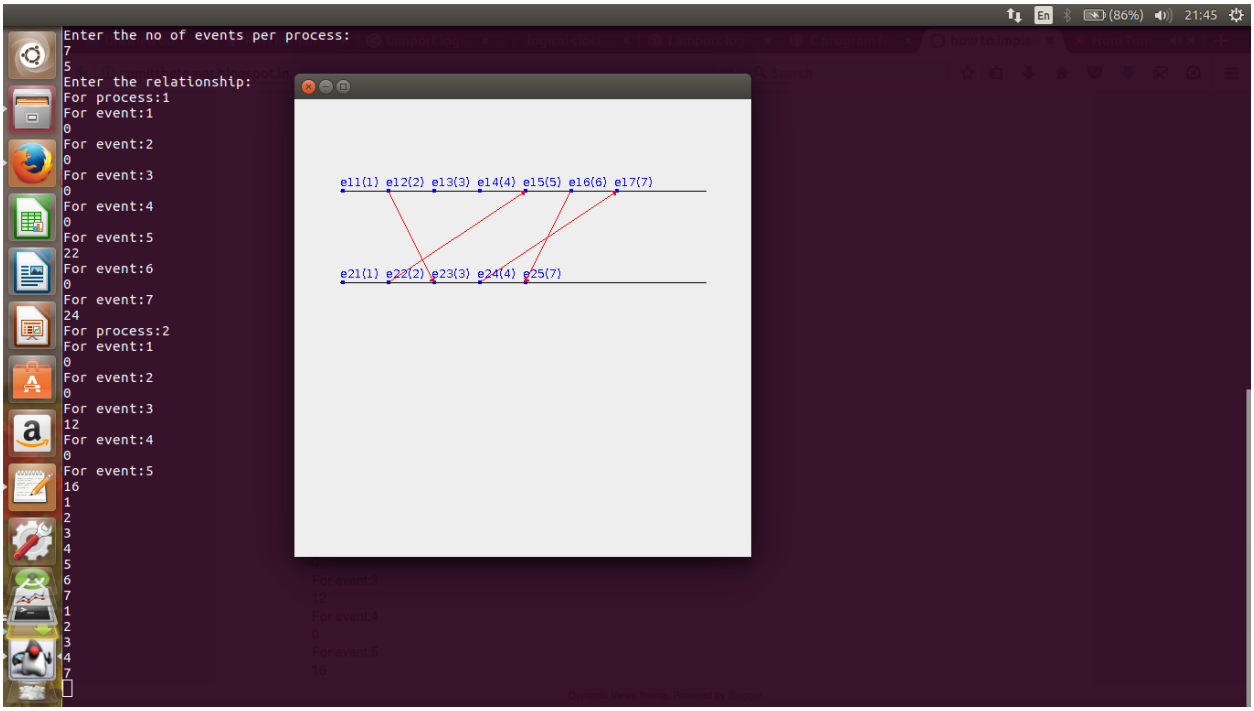
```
import java.util.*;
import java.util.Scanner;
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
public class lamport
{
   int e[][]=new int[10][10];
   int en[][]=new int[10][10];
   int ev[]=new int[10];
   int i,p,j,k;
   HashMap<Integer,Integer> hm=new HashMap<Integer,Integer>();
   int xpoints[] =new int[5];
   int ypoints[] =new int[5];
   class draw extends JFrame
   {
                 private final int ARR_SIZE = 4;
         void drawArrow(Graphics g1, int x1, int y1, int x2, int y2)
         {
                Graphics2D g = (Graphics2D) g1.create();

                double dx = x2 - x1, dy = y2 - y1;
                double angle = Math.atan2(dy, dx);
                int len = (int) Math.sqrt(dx*dx + dy*dy);
                AffineTransform at = AffineTransform.getTranslateInstance(x1, y1);
                at.concatenate(AffineTransform.getRotateInstance(angle));
                g.transform(at);

                // Draw horizontal arrow starting in (0, 0)
                g.drawLine(0, 0, len, 0);
                g.fillPolygon(new int[] {len, len-ARR_SIZE, len-ARR_SIZE, len},
                        new int[] {0, -ARR_SIZE, ARR_SIZE, 0}, 4);
         }

         public void paintComponent(Graphics g) {
```

```
            for (int x = 15; x < 200; x += 16)
            drawArrow(g, x, x, x, 150);
            drawArrow(g, 30, 300, 300, 190);
        }
             public void paint(Graphics g){
                    int h1,h11,h12;
                    Graphics2D go=(Graphics2D)g;
                    go.setPaint(Color.black);
                    for(i=1;i<=p;i++)
                    {
                    go.drawLine(50,100*i,450,100*i);
                    }
                         for(i=1;i<=p;i++)
                    {
                      for(j=1;j<=ev[i];j++)
                        {
                              k=i*10+j;
                              go.setPaint(Color.blue);
                              go.fillOval(50*j,100*i-3,5,5);
                             go.drawString("e"+i+j+"("+en[i][j]+")",50*j,100*i-5);
                             h1=hm.get(k);
                             if(h1!=0)
                              {
                                h11=h1/10;
                                h12=h1%10;
                                go.setPaint(Color.red);
                                drawArrow(go,50*h12+2,100*h11,50*j+2,100*i);
                              }
                         }
                     }

             }
}
public void calc()
{
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number of process:");
        p=sc.nextInt();
        System.out.println("Enter the no of events per process:");
        for(i=1;i<=p;i++)
```

```
{
  ev[i]=sc.nextInt();
}
System.out.println("Enter the relationship:");
for(i=1;i<=p;i++)
{
        System.out.println("For process:"+i);
        for(j=1;j<=ev[i];j++)
        {
                System.out.println("For event:"+(j));
                int input=sc.nextInt();
                k=i*10+j;
                hm.put(k,input);
                if(j==1)
               en[i][j]=1;
        }
}

for(i=1;i<=p;i++)
{
        for(j=2;j<=ev[i];j++)
        {
                k=i*10+j;
                if(hm.get(k)==0)
                {
                        en[i][j]=en[i][j-1]+1;
                }
                else
                {
                        int a=hm.get(k);
                       int p1=a/10;
                        int e1=a%10;
                       if(en[p1][e1]>en[i][j-1])
                       en[i][j]=en[p1][e1]+1;
                       else
                               en[i][j]=en[i][j-1]+1;
        }
}
 }
for(i=1;i<=p;i++)
```

```
        {
                for(j=1;j<=ev[i];j++)
                {
                        System.out.println(en[i][j]);

                }
        }
        JFrame jf=new draw();
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jf.setSize(500,500);
        jf.setVisible(true);
    }
    public static void main(String[] args)
    {

        lamport lam=new lamport();
        lam.calc();
    }
}
```

OUTPUT



Program 7

Aim: Write a program to implement Cristian's Algorithm

ClockServer.java

```
import java.io.*;
import java.net.*;
import java.util.*;
public class ClockServer {
	public static void main(String[] args) throws IOException {
	String port;
	BufferedReader stdIn =new BufferedReader(new InputStreamReader(System.in));
	System.out.println("Enter the port no");
	port=stdIn.readLine();
	int portNumber = Integer.parseInt(port);
	try (
	ServerSocket serverSocket =
		new ServerSocket(portNumber);
	Socket clientSocket = serverSocket.accept();
	PrintWriter out =
		new PrintWriter(clientSocket.getOutputStream(), true);
	BufferedReader in = new BufferedReader(
		new InputStreamReader(clientSocket.getInputStream()));

	) {
	String inputLine;
	System.out.println("Server Started");
	while (true) {
		inputLine = in.readLine();
		if(inputLine.equalsIgnoreCase("Exit"))
		{
		System.out.println("Exiting");
		out.println("Server Exiting");
		break;
		}
		out.println(System.currentTimeMillis()+5000);
	}
	} catch (IOException e) {
```

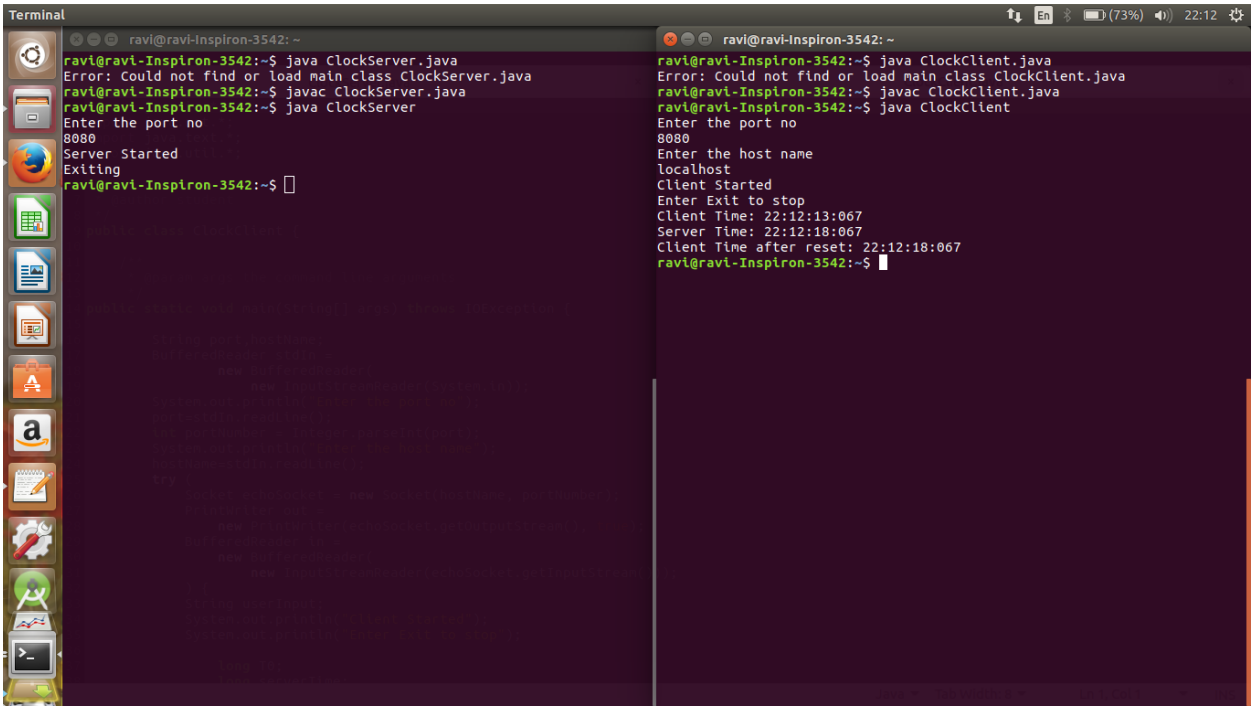
```
        System.out.println("Exception caught when trying to listen on port "
                + portNumber + " or listening for a connection");
        System.out.println(e.getMessage());
        }
        }

}
```

ClockClient.java

```
import java.io.*;
import java.net.*;
import java.text.*;
import java.util.*;
public class ClockClient {
public static void main(String[] args) throws IOException {

        String port,hostName;
        BufferedReader stdIn =new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the port no");
        port=stdIn.readLine();
        int portNumber = Integer.parseInt(port);
        System.out.println("Enter the host name");
        hostName=stdIn.readLine();
        try (
        Socket echoSocket = new Socket(hostName, portNumber);
        PrintWriter out =
                new PrintWriter(echoSocket.getOutputStream(), true);
        BufferedReader in =
                new BufferedReader(
                new InputStreamReader(echoSocket.getInputStream()));
        ) {
        String userInput;
        System.out.println("Client Started");
        System.out.println("Enter Exit to stop");

                long T0;
                long serverTime;
                long T1;
                long finalTime;
                out.println(T0=System.currentTimeMillis());
```

```
                serverTime = Long.parseLong(in.readLine());
                T1 =System.currentTimeMillis();
                finalTime =  serverTime + (T1-T0)/2;
                DateFormat formatter = new SimpleDateFormat("HH:mm:ss:SSS");
                System.out.println("Client Time: " + formatter.format(new Date(T1)));
                System.out.println("Server Time: " + formatter.format(new Date(serverTime)));
                System.out.println("Client Time after reset: " + formatter.format(new
Date(finalTime)));
                out.println("EXit");
        } catch (UnknownHostException e) {
        System.err.println("Don't know about host " + hostName);
        System.exit(1);
        } catch (IOException e) {
        System.err.println("Couldn't get I/O for the connection to " +
                hostName);
        System.exit(1);
        }
        }

}
```

OUTPUT



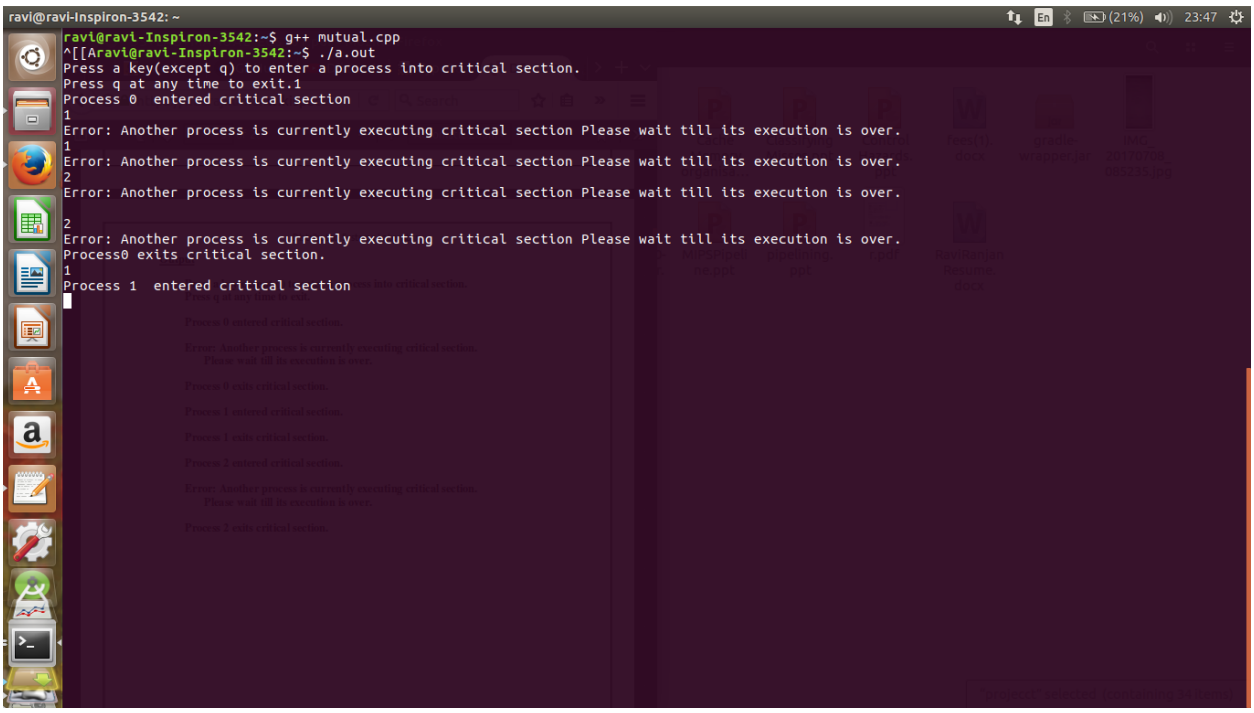
Program 9

Aim: Write a program to simulate the Distributed Mutual exclusion using Centralized algorithm

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int cs=0,pro=0;
   double run=5;
   char key='a';
   time_t t1,t2;
   cout<<"Press a key(except q) to enter a process into critical section.";
   cout<<"\nPress q at any time to exit.";
   t1 = time(NULL)-5;
   while(key!='q')
   {
         if(cs!=0)
         {
                  t2 = time(NULL);
                  if(t2-t1 > run)
                  {
                           cout<<"Process"<<pro-1;
                           cout<<" exits critical section."<<endl;
                           cs=0;
                  }
         }
         cin>>key;
         if(key!='q')
         {
                  if(cs!=0)
                  printf("Error: Another process is currently executing critical section Please wait till its execution is over.\n");
                  else
                  {
                           printf("Process %d ",pro);
```

```
                        printf(" entered critical section\n");
                        cs=1;
                        pro++;
                        t1 = time(NULL);
                }
        }
    }
}
```

OUTPUT



Program 10

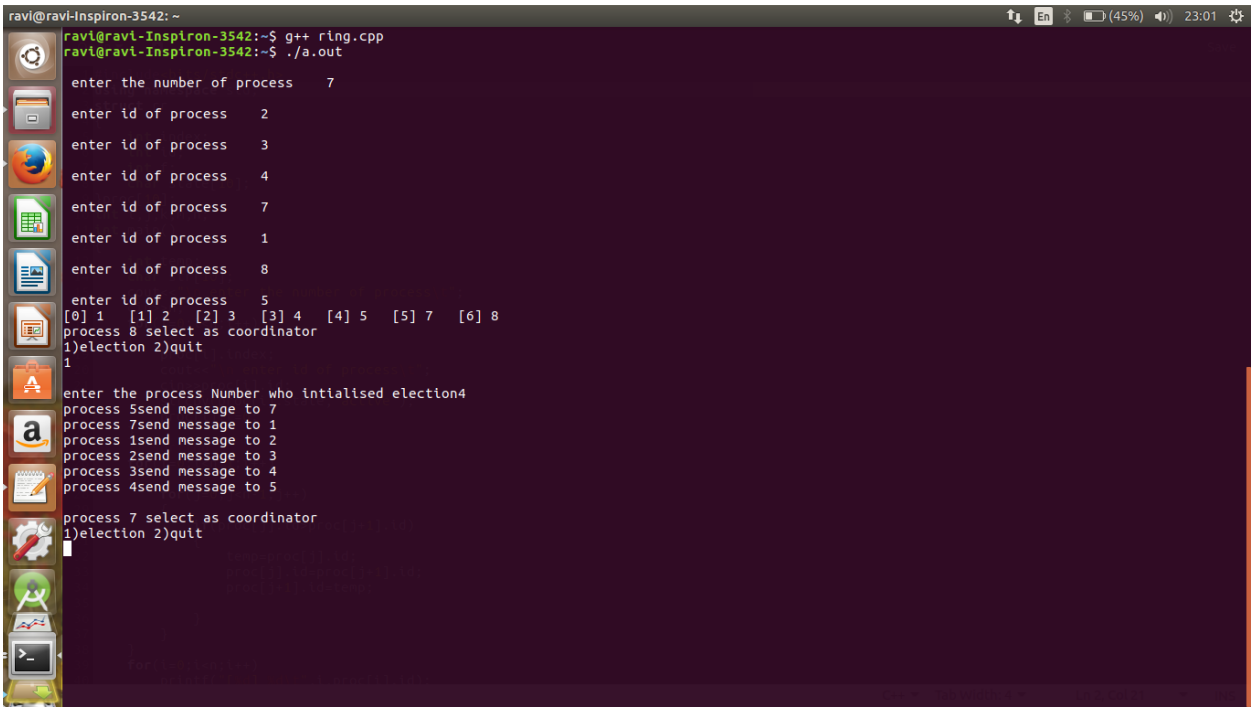
Aim: Write a program to implement simulate ring based election

```
#include<bits/stdc++.h>
using namespace std;
struct rr
{
   int index;
   int id;
   int f;
   char state[10];
}proc[10];
int i,j,k,m,n;
int main()
{
   int temp;
   char str[10];
   cout<<"\n enter the number of process\t";
   cin>>n;
   for(i=0;i<n;i++)
   {
         proc[i].index;
         cout<<"\n enter id of process\t";
         cin>>proc[i].id;
         strcpy(proc[i].state,"active");
         proc[i].f=0;
   }
   for(i=0;i<n-1;i++)
   {
         for(j=0;j<n-1;j++)
         {
               if(proc[j].id>proc[j+1].id)
               {
                     temp=proc[j].id;
                     proc[j].id=proc[j+1].id;
                     proc[j+1].id=temp;
```

```
                }
        }
    }
    for(i=0;i<n;i++)
        printf("[%d] %d\t",i,proc[i].id);
    int init;
    int ch;
    int temp1;
    int temp2;
    int arr[10];
    strcpy(proc[n-1].state,"inactive");
    cout<<"\nprocess "<<proc[n-1].id<<" select as coordinator";
    while(1)
    {
        cout<<"\n1)election 2)quit\n";
        scanf("%d",&ch);
        for(i=0;i<n;i++)
        {
                proc[i].f=0;
        }
        switch(ch)
        {
                case 1:
                {
                        cout<<"\nenter the process Number who intialised election";
                        scanf("%d",&init);
                        temp2=init;
                        temp1=init+1;
                    i=0;
                    while(temp2!=temp1)
                    {
                        if(strcmp(proc[temp1].state,"active")==0 && proc[temp1].f==0 )
                                {
                                        cout<<"process "<<proc[init].id<<"send message to
"<<proc[temp1].id<<"\n";
                                        proc[temp1].f=1;
                                        init=temp1;
                                        arr[i]=proc[temp1].id;
                                        i++;
                                }
```

```
                    if(temp1==n)
                    temp1=0;
                    else
                    temp1++;

            }

                cout<<"process "<<proc[init].id<<"send message to
"<<proc[temp1].id<<"\n";
                arr[i]=proc[temp1].id;
                i++;
                int max=-1;
                for(j=0;j<i;j++)
                {
                        if(max<arr[j])
                        max=arr[j];
                }
            cout<<"\nprocess "<<max<<" select as coordinator";
            for(i=0;i<n;i++)
            {
                if(proc[i].id==max)
                {
                        strcpy(proc[i].state,"inactive");
                        }
                }
                break;
        }
    }
  }
  return 0;
}
```

OUTPUT



```
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ g++ ring.cpp
ravi@ravi-Inspiron-3542:~$ ./a.out

 enter the number of process    7

 enter id of process    2

 enter id of process    3

 enter id of process    4

 enter id of process    7

 enter id of process    1

 enter id of process    8

 enter id of process    5
[0] 1   [1] 2   [2] 3   [3] 4   [4] 5   [5] 7   [6] 8
process 8 select as coordinator
1)election 2)quit
1

enter the process Number who intialised election4
process 5send message to 7
process 7send message to 1
process 1send message to 2
process 2send message to 3
process 3send message to 4
process 4send message to 5

process 7 select as coordinator
1)election 2)quit
```

Program 11

Aim: Write a program to implement Bully Election Algorithm

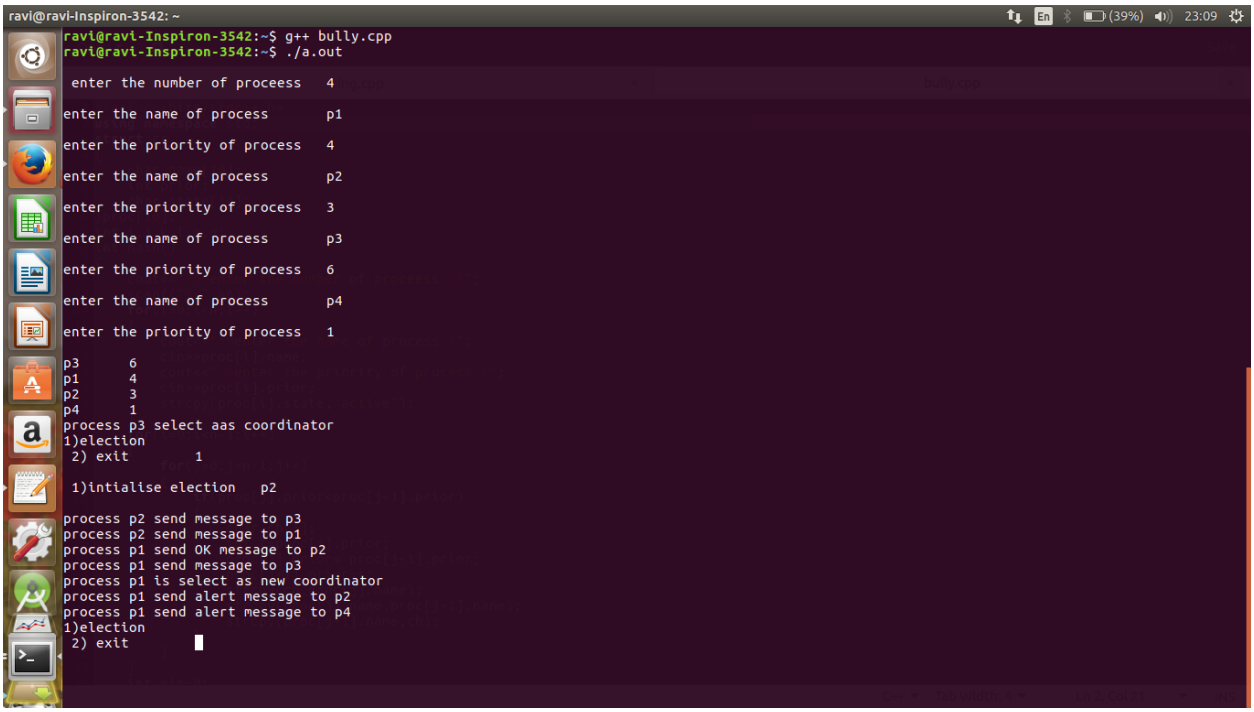
```
#include<bits/stdc++.h>
using namespace std;
struct rr
{
   char name[10];
   int prior;
   char state[10];
}proc[10];
int i,j,k,l,m,n;
int main()
{
   cout<<"\n enter the number of proceess \t";
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
         cout<<"\nenter the name of process\t";
         cin>>proc[i].name;
         cout<<"\nenter the priority of process\t";
         cin>>proc[i].prior;
         strcpy(proc[i].state,"active");
   }
   for(i=0;i<n-1;i++)
   {
         for(j=0;j<n-1;j++)
         {
               if(proc[j].prior<proc[j+1].prior)
               {
                     char ch[5];
                     int t=proc[j].prior;
                     proc[j].prior= proc[j+1].prior;
                     proc[j+1].prior=t;
                     strcpy(ch,proc[j].name);
                     strcpy(proc[j].name,proc[j+1].name);
                     strcpy(proc[j+1].name,ch);
               }
```

```
        }
}
int min=0;
for(i=0;i<n;i++)
        cout<<"\n"<<proc[i].name<<"\t"<<proc[i].prior;
for(i=0;i<n;i++)
{
        for(i=0;i<n;i++)
        {
                if(min<proc[i].prior)
                        min=proc[i].prior;
        }
}
for(i=0;i<n;i++)
{
        if(proc[i].prior==min)
        {
                cout<<"\nprocess "<<proc[i].name<<" select aas coordinator";
                strcpy(proc[i].state,"iactive");
                break;
        }
}
int pr;
while(1)
{
        int ch;
        cout<<"\n1)election\t";
        cout<<"\n 2) exit  \t";
        cin>>ch;
        int max=0;
        int ar[20];
        k=0;
        int fl=0;
        switch(ch)
        {
                case 1: char str[5];
                        cout<<"\n 1)intialise election\t";
                        cin>>str;
                          fl=0;
                        l1: for(i=0;i<n;i++)
```

```
{
        if(strcmp(str,proc[i].name)==0)
        {
                pr=proc[i].prior;
        }
}
for(i=0;i<n;i++)
{
        if(pr<proc[i].prior)
        {
                cout<<"\nprocess "<<str<<" send message to "<<proc[i].name;
        }
}
for(i=0;i<n;i++)
{
        if(pr<proc[i].prior && strcmp(proc[i].state,"active")==0 )
        {
                if(fl==0)
                {
                        ar[k]= proc[i].prior;
                        k++;
                }
                cout<<"\nprocess "<<proc[i].name<<" send OK message to "<<str;
                if(proc[i].prior>max)
                        max=proc[i].prior;
        }
}
fl=1;
if(k!=0)
{
        k=k-1;
        for(i=0;i<n;i++)
        {
                if(ar[k]==proc[i].prior)
                        strcpy(str,proc[i].name);
        }
        goto l1;
}
```

```
                m=0;
                for(j=0;j<n;j++)
                {
                        if(proc[j].prior>m && strcmp(proc[j].state,"active")==0 )
                        {
                                cout<<"\nprocess "<<proc[j].name <<" is select as new
coordinator";
                                strcpy(proc[j].state,"inactive");
                                break;
                        }
                }
                for(i=0;i<n;i++)
                {
                        if(strcmp(proc[i].state,"active")==0                    &&
proc[j].prior>proc[i].prior)
                        {
                                cout<<"\nprocess "<<proc[j].name<<" send alert message
to "<<proc[i].name;
                        }
                }
                break;
        case 2:
                exit(1);

        }
    }
}
```

OUTPUT



```
ravi@ravi-Inspiron-3542: ~
ravi@ravi-Inspiron-3542:~$ g++ bully.cpp
ravi@ravi-Inspiron-3542:~$ ./a.out

 enter the number of proceess   4

enter the name of process       p1

enter the priority of process   4

enter the name of process       p2

enter the priority of process   3

enter the name of process       p3

enter the priority of process   6

enter the name of process       p4

enter the priority of process   1

p3      6
p1      4
p2      3
p4      1
process p3 select aas coordinator
1)election
 2) exit        1

 1)intialise election   p2

process p2 send message to p3
process p2 send message to p1
process p1 send OK message to p2
process p1 send message to p3
process p1 is select as new coordinator
process p1 send alert message to p2
process p1 send alert message to p4
1)election
 2) exit
```