Program1:

Write a program to implement Client Server Using RPC

Sol:-

SERVER CODE:

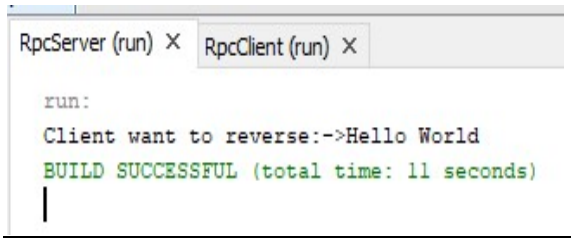
```
package rpcserver;
import java.io.*;
import java.net.*;
public class RpcServer
{
   public static void main(String[] args)
   {
      try
      {
         ServerSocket ss=new ServerSocket(1234);
         Socket s=ss.accept();
         DataInputStream dis=new DataInputStream(s.getInputStream());
         DataOutputStream dout=new DataOutputStream(s.getOutputStream());
         StringBuffer str = new StringBuffer((String)dis.readUTF());
         System.out.println("Client want to reverse:->"+str);
         String k=new String(str.reverse());
         dout.writeUTF(k);
         dout.flush();
         dis.close();
         dout.close();
         s.close();
         ss.close();
       }
      catch(Exception e)
      {
         System.out.println(e);
      }
   }
}
```

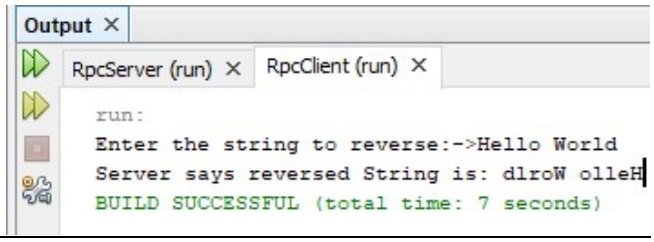
CLIENT CODE:

```
package rpcclient;
import java.net.*;
import java.io.*;
public class RpcClient
{
    public static void main(String[] args)
   {
      try
```

```
    {
      Socket s=new Socket("localhost",1234);
      DataInputStream dis=new DataInputStream(s.getInputStream());
      DataOutputStream dout=new DataOutputStream(s.getOutputStream());
      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
      String str="",str1="";
       System.out.print("Enter the string to reverse:->");
       str=br.readLine();
       dout.writeUTF(str);
       dout.flush();
       str1=dis.readUTF();
       System.out.println("Server says reversed String is: "+str1);

      dout.close();
      dis.close();
      s.close();
    }
    catch(Exception e)
    {
      System.out.println(e);
    }
  }
  }
```

OUTPUT:





Program2:

Write a program to implement Client Server Using RMI

Sol:-

1.CalculatorImpl.java

```
public class CalculatorImpl
   extends
     java.rmi.server.UnicastRemoteObject
   implements Calculator {
  public CalculatorImpl()
      throws java.rmi.RemoteException {
      super();
   }
 public long add(long a, long b)
      throws java.rmi.RemoteException {
      return a + b;
   }
 public long sub(long a, long b)
      throws java.rmi.RemoteException {
      return a - b;
   }
 public long mul(long a, long b)
      throws java.rmi.RemoteException {
      return a * b;
   }
 public long div(long a, long b)
      throws java.rmi.RemoteException {
      return a / b;
   }
}
```

2.Calculator.java

```
public interface Calculator
        extends java.rmi.Remote {
   public long add(long a, long b)
      throws java.rmi.RemoteException;
   public long sub(long a, long b)
      throws java.rmi.RemoteException;
   public long mul(long a, long b)
      throws java.rmi.RemoteException;
   public long div(long a, long b)
      throws java.rmi.RemoteException;
```

```
}
```

3.CalculatorServer.java

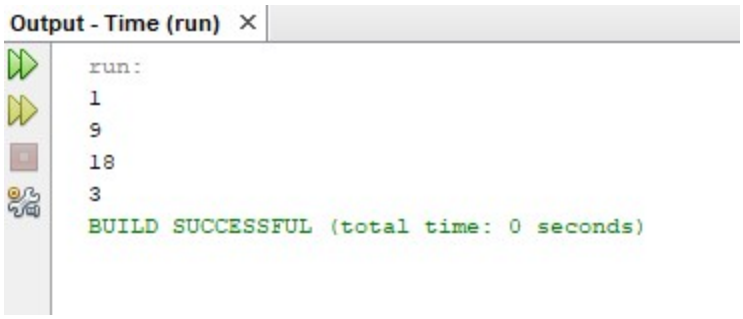
```
import java.rmi.Naming;
public class CalculatorServer {
  public CalculatorServer() {
    try {
      Calculator c = new CalculatorImpl();
      Naming.rebind("rmi://localhost:1099/CalculatorService", c);
    } catch (Exception e) {
      System.out.println("Trouble: " + e);
    }
  }
public static void main(String args[]) {
    new CalculatorServer();
  }
}
```

4.CalculatorClient.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
 public class CalculatorClient {
  public static void main(String[] args) {
      try {
         Calculator c = (Calculator)
                  Naming.lookup("rmi://localhost/CalculatorService");
         System.out.println( c.sub(4, 3) );
         System.out.println( c.add(4, 5) );
         System.out.println( c.mul(3, 6) );
         System.out.println( c.div(9, 3) );
      }
      catch (MalformedURLException murle) {
         System.out.println();
         System.out.println("MalformedURLException");
         System.out.println(murle);
      }
      catch (RemoteException re) {
         System.out.println();
         System.out.println("RemoteException");
         System.out.println(re);
      }
      catch (NotBoundException nbe) {
```

```
        System.out.println();
        System.out.println(
                "NotBoundException");
        System.out.println(nbe);
      }
      catch ( java.lang.ArithmeticException ae) {
        System.out.println();
        System.out.println(
         "java.lang.ArithmeticException");
        System.out.println(ae);
      }
    }
}
```

OUTPUT:



Program3:

Write a program to implement multithreaded Client-Server using processes.

Sol:-

SERVER CODE:

```
import java.io.*;
import java.text.*;
import java.util.*;
import java.net.*;

// Server class
public class Server1
{
   public static void main(String[] args) throws IOException
   {
      // server is listening on port 5056
      ServerSocket ss = new ServerSocket(5056);

      // running infinite loop for getting
      // client request
      while (true)
      {
         Socket s = null;

         try
         {
            // socket object to receive incoming client requests
            s = ss.accept();

            System.out.println("A new client is connected : " + s);

            // obtaining input and out streams
            DataInputStream dis = new DataInputStream(s.getInputStream());
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());

            System.out.println("Assigning new thread for this client");

            // create a new thread object
            Thread t = new ClientHandler(s, dis, dos);

            // Invoking the start() method
            t.start();

         }
```

```
        catch (Exception e){
            s.close();
            e.printStackTrace();
        }
    }
  }
}

// ClientHandler class
class ClientHandler extends Thread
{
  DateFormat fordate = new SimpleDateFormat("yyyy/MM/dd");
  DateFormat fortime = new SimpleDateFormat("hh:mm:ss");
  final DataInputStream dis;
  final DataOutputStream dos;
  final Socket s;


  // Constructor
  public ClientHandler(Socket s, DataInputStream dis, DataOutputStream dos)
  {
    this.s = s;
    this.dis = dis;
    this.dos = dos;
  }

  @Override
  public void run()
  {
    String received;
    String toreturn;
    while (true)
    {
      try {

        // Ask user what he wants
        dos.writeUTF("What do you want?[Date | Time]..\n"+
                "Type Exit to terminate connection.");

        // receive the answer from client
        received = dis.readUTF();

        if(received.equals("Exit"))
        {
          System.out.println("Client " + this.s + " sends exit...");
          System.out.println("Closing this connection.");
          this.s.close();
          System.out.println("Connection closed");
```

```
            break;
        }

        // creating Date object
        Date date = new Date();

        // write on output stream based on the
        // answer from the client
        switch (received) {

            case "Date" :
                toreturn = fordate.format(date);
                dos.writeUTF(toreturn);
                break;

            case "Time" :
                toreturn = fortime.format(date);
                dos.writeUTF(toreturn);
                break;

            default:
                dos.writeUTF("Invalid input");
                break;
        }
      } catch (IOException e) {
        e.printStackTrace();
      }
    }

    try
    {
      // closing resources
      this.dis.close();
      this.dos.close();

    }catch(IOException e){
      e.printStackTrace();
    }
  }
}
```

CLIENT CODE

```
import java.io.*;
import java.net.*;
import java.util.Scanner;

// Client class
public class Client1
```

```
{
   public static void main(String[] args) throws IOException
   {
      try
      {
         Scanner scn = new Scanner(System.in);

         // getting localhost ip
         InetAddress ip = InetAddress.getByName("localhost");

         // establish the connection with server port 5056
         Socket s = new Socket(ip, 5056);

         // obtaining input and out streams
         DataInputStream dis = new DataInputStream(s.getInputStream());
         DataOutputStream dos = new DataOutputStream(s.getOutputStream());

         // the following loop performs the exchange of
         // information between client and client handler
         while (true)
         {
            System.out.println(dis.readUTF());
            String tosend = scn.nextLine();
            dos.writeUTF(tosend);

            // If client sends exit,close this connection
            // and then break from the while loop
            if(tosend.equals("Exit"))
            {
               System.out.println("Closing this connection : " + s);
               s.close();
               System.out.println("Connection closed");
               break;
            }

            // printing date or time as requested by client
            String received = dis.readUTF();
            System.out.println(received);
         }

         // closing resources
         scn.close();
         dis.close();
         dos.close();
      }catch(Exception e){
         e.printStackTrace();
      }
   }
```

}

OUTPUT:

```
Output ×
Server1 (run) ×   Client1 (run) ×
run:
A new client is connected : Socket[addr=/127.0.0.1,port=50463,localport=5056]
Assigning new thread for this client
Client Socket[addr=/127.0.0.1,port=50463,localport=5056] sends exit...
Closing this connection.
Connection closed
```

```
Output ×
Server1 (run) ×   Client1 (run) ×
run:
What do you want?[Date | Time]..
Type Exit to terminate connection.
Date
2017/10/29
What do you want?[Date | Time]..
Type Exit to terminate connection.
Time
10:30:29
What do you want?[Date | Time]..
Type Exit to terminate connection.
Exit
Closing this connection : Socket[addr=localhost/127.0.0.1,port=5056,localport=50463]
Connection closed
BUILD SUCCESSFUL (total time: 25 seconds)
```

Program4:

Write a program to implement Distributed Chat server using TCP socket.

Sol:-

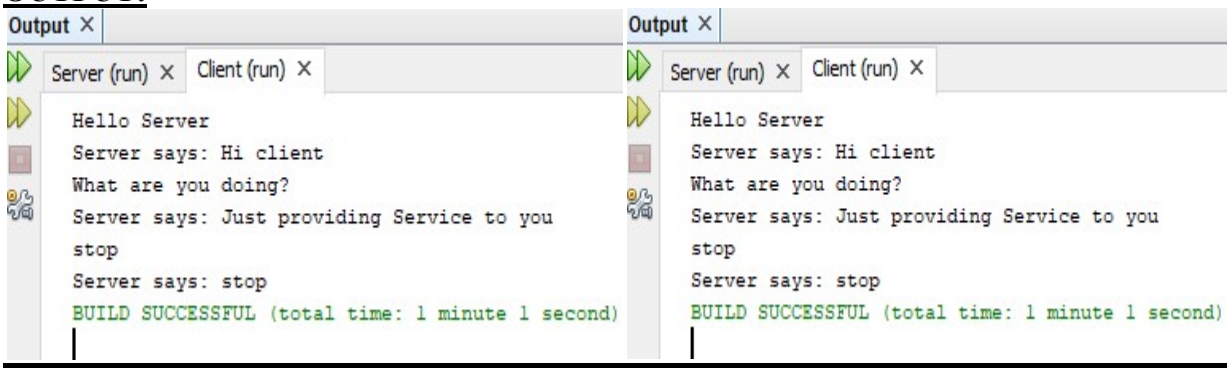
SERVER CODE:

```
import java.io.*;
import java.net.*;

public class Server
{
   public static void main(String[] args)
   {
      try
      {
         ServerSocket ss=new ServerSocket(1234);
         Socket s=ss.accept();  // establish connection
         DataInputStream dis=new DataInputStream(s.getInputStream());
         DataOutputStream dout=new DataOutputStream(s.getOutputStream());// recieve client input
         BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); // send input
stream to client
         String str="",str1="";
         while(!str.equals("stop"))
         {
            str=(String)dis.readUTF();
            System.out.println("Client Says "+str);
            str1=br.readLine();
            dout.writeUTF(str1);
            dout.flush();
         }
         dis.close();
         dout.close();
         s.close();
         ss.close();
      }
      catch(Exception e)
      {
         System.out.println(e);
      }
   }

}
```

CLIENT CODE

```
import java.io.*;
import java.net.*;
public class Client
{
   public static void main(String[] args)
   {
      try
      {
         Socket s=new Socket("localhost",1234);
         DataInputStream dis=new DataInputStream(s.getInputStream());
         DataOutputStream dout=new DataOutputStream(s.getOutputStream());// recieve server input
         BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); // send server
stream to client
         String str="",str1="";
         while(!str.equals("stop"))
         {
          str=(String)br.readLine();
          dout.writeUTF(str);
          dout.flush();
          str1=dis.readUTF();
          System.out.println("Server says: "+str1);

         }
         dout.close();
         dis.close();
         s.close();
      }
      catch(Exception e)
      {
         System.out.println(e);
      }
   }

}
```

OUTPUT:



DATE:-

Program5:

Write a program to implement CORBA Mechanism by using C++ program at one end and java program on Other.

Sol:

Server programs

```
#ifndef __hello_skel_h__
#define __hello_skel_h__
#include <hello.h>

class Hello_skel : virtual public Hello,

virtual public CORBA_Object_skel
{
static CORBA_ULong _ob_num_;

Hello_skel(const Hello_skel&);

void operator=(const Hello_skel&);

protected:

Hello_skel() { }

Hello_skel(const char*);

public:

Hello_ptr _this() { return Hello::_duplicate(this); }

virtual CORBA_ULong _OB_incNumber() const;

virtual OBDispatchStatus _OB_dispatch(const char*, OBFixSeq< CORBA_Octet >&,

bool, CORBA_ULong, CORBA_ULong);
};

#endif
```

```
#include <OB/CORBA.h>

#include <hello_skel.h>

//

IDL:Hello:1.0

CORBA_ULong Hello_skel::_ob_num_ = 0;

Hello_skel::Hello_skel(const char* name)

{
assert_nca(name, OBNCANullString);

try

{
_OB_createObjectKeyWithName(name);
}
catch(...)

{
_OB_setRef(0);

throw;
}

}

CORBA_ULong

Hello_skel::_OB_incNumber() const
{

return Hello_skel::_ob_num_++;
}

OBDispatchStatus

Hello_skel::_OB_dispatch(const char* _ob_op,

OBFixSeq< CORBA_Octet >& _ob_seq,
bool _ob_sw,
CORBA_ULong _ob_offIn,
CORBA_ULong _ob_offOut)
{
```

```
if(strcmp(_ob_op, "hello") == 0)
{

hello();

CORBA_ULong _ob_cnt = _ob_offOut;

_ob_seq.length(0);

_ob_seq.length(_ob_cnt);
#ifdef OB_CLEAR_MEM

memset(_ob_seq.data(), 0, _ob_seq.length());
#endif

return OBDispatchStatusOK;

}

else
return CORBA_Object_skel::_OB_dispatch(_ob_op, _ob_seq, _ob_sw,

_ob_offIn, _ob_offOut);
}

#ifndef __hello_h__

#define __hello_h__

//

IDL:Hello:1.0

class Hello;
typedef Hello* Hello_ptr; typedef Hello* HelloRef;
typedef OBObjVar< Hello > Hello_var;

//

IDL:Hello:1.0


class Hello : virtual public CORBA_Object
{
Hello(const Hello&);
void operator=(const Hello&);

protected:
```

```
Hello() { }
public:

static inline Hello_ptr

_duplicate(Hello_ptr p)
{
CORBA_Object::_duplicate(p);
return p;
}

static inline Hello_ptr

_nil()
{
return 0;

}

static Hello_ptr _narrow(CORBA_Object_ptr); virtual void* _OB_narrowHelp(const char*) const; virtual
const char* _OB_typeId() const;

friend void OBUnmarshal(Hello_ptr&, const CORBA_Octet*&, bool); friend CORBA_Boolean operator>>=(const
CORBA_Any&, Hello_ptr&);

//

IDL:Hello/hello:1.0

virtual void hello();
};

extern const OBTypeCodeConst _tc_Hello;

//

IDL:Hello:1.0


inline void
CORBA_release(Hello_ptr p)
{
CORBA_release((CORBA_Object_ptr)p);
}

inline CORBA_Boolean
```

```
CORBA_is_nil(Hello_ptr p)
{
return p == 0;

}

inline void

OBMarshal(Hello_ptr p, CORBA_Octet*& oct)
{

OBMarshal((CORBA_Object_ptr)p, oct);
}

inline void

OBMarshalCount(Hello_ptr p, CORBA_ULong& count)

{
OBMarshalCount((CORBA_Object_ptr)p, count);
}

void OBUnmarshal(Hello_ptr&, const CORBA_Octet*&, bool);

void operator<<=(CORBA_Any&, Hello_ptr); void operator<<=(CORBA_Any&, Hello_ptr*);

CORBA_Boolean operator>>=(const CORBA_Any&, Hello_ptr&);
inline void

operator<<=(CORBA_Any_var& any, Hello_ptr val)

{
any.inout() <<= val;
}

inline void

operator<<=(CORBA_Any_var& any, Hello_ptr* val)
{
any.inout() <<= val;
}

inline CORBA_Boolean

operator>>=(const CORBA_Any_var& any, Hello_ptr& val)
{
return any.in() >>= val;
}
```

```
#endif



#include <OB/CORBA.h>

#include <OB/TemplateI.h>
#include <hello.h>

//

IDL:Hello:1.0

#ifndef HAVE_NO_EXPLICIT_TEMPLATES template class OBObjVar< Hello >; template class
OBObjForSeq< Hello >; #endif

Hello_ptr

Hello::_narrow(CORBA_Object_ptr p)
{

if(!CORBA_is_nil(p))
{
void* v = p -> _OB_narrowHelp("IDL:Hello:1.0");

if(v)

return _duplicate((Hello_ptr)v);

if(p -> _OB_remoteIsA("IDL:Hello:1.0"))

{
Hello_ptr val = new Hello;

val -> _OB_copyFrom(p);
return val;
}
}

return _nil();

}

void*

Hello::_OB_narrowHelp(const char* _ob_id) const
```

```
{
if(strcmp("IDL:Hello:1.0", _ob_id) == 0)
return (void*)this;
else
return CORBA_Object::_OB_narrowHelp(_ob_id);

}

const char*

Hello::_OB_typeId() const
{
return "IDL:Hello:1.0";
}

void

OBUnmarshal(Hello_ptr& val, const CORBA_Octet*& coct, bool swap)
{
Hello_var old = val;

CORBA_Object_var p;
OBUnmarshal(p.inout(), coct, swap);

if(!CORBA_is_nil(p))

{

void* v = p -> _OB_narrowHelp("IDL:Hello:1.0");

if(v)

val = Hello::_duplicate((Hello_ptr)v);
else

{
assert_nca(!(p -> _is_local() && p -> _is_dynamic()), OBNCADynamicAsStatic);

assert(!p -> _is_local());
val = new Hello;

val -> _OB_copyFrom(p);
}
}
else
val = Hello::_nil();

}
```

```
const OBTypeCodeConst _tc_Hello(

"010000000E0000002200000001000000E00000049444C3A48656C6C6F3A312E3000000006000"
"00048656C6C6F00"

);

void

operator<<=(CORBA_Any& any, Hello_ptr val)
{

OBObjAny* o = new OBObjAny;
o -> b = CORBA_Object::_duplicate(val);
o -> d = CORBA_Object::_duplicate(val);
any.replace(_tc_Hello, o, true);
}

void

operator<<=(CORBA_Any& any, Hello_ptr* val)
{
OBObjAny* o = new OBObjAny;

o -> b = *val;
o -> d = CORBA_Object::_duplicate(*val);
any.replace(_tc_Hello, o, true);
}

CORBA_Boolean

operator>>=(const CORBA_Any& any, Hello_ptr& val)
{
if(any.check_type(_tc_Hello))
{

OBObjAny* o = (OBObjAny*)any.value();
assert(o);

if(!CORBA_is_nil(o -> d))

{
void* v = o -> d -> _OB_narrowHelp("IDL:Hello:1.0");

if(v)

val = (Hello_ptr)v;
```

```
else
{
assert_nca(!(o -> d -> _is_local() && o -> d -> _is_dynamic()), OBNCADynamicAsStatic);
assert(!o -> d -> _is_local());
val = new Hello;
val -> _OB_copyFrom(o -> d);
OBObjAny* no = new OBObjAny;

no -> b = CORBA_Object::_duplicate(o -> b);
no -> d = val;
((CORBA_Any&)any).replace(_tc_Hello, no, true);
}
}

else
val = Hello::_nil();

return true;

}

else
return false;
}
void
Hello::hello()

{
if(CORBA_is_nil(_ob_con_))
throw CORBA_NO_IMPLEMENT();
CORBA_ULong _ob_off = _ob_con_ -> offset(this, "hello"); CORBA_ULong _ob_cnt = _ob_off;

OBFixSeq< CORBA_Octet > _ob_seq(_ob_cnt); _ob_seq.length(_ob_cnt);
#ifdef OB_CLEAR_MEM
memset(_ob_seq.data(), 0, _ob_seq.length()); #endif

bool _ob_sw, _ob_ex, _ob_fo;

_ob_off = _ob_con_ -> request(this, "hello", _ob_seq, _ob_sw, _ob_ex, _ob_fo, _ob_tout_);

if(_ob_fo)

{
const CORBA_Octet* _ob_co = _ob_seq.data() + _ob_off; _OB_forward(_ob_co, _ob_sw); hello();

return;

}
```

```
if(_ob_ex)

throw CORBA_UNKNOWN();
}
#include <hello_skel.h>

class Hello_impl : public Hello_skel

{
public:

Hello_impl();

virtual void hello();

};


#include <CORBA.h>

#include <hello_impl.h>
Hello_impl::Hello_impl()
{
}

void
Hello_impl::hello()
{
cout << "Hello World!" << endl;
}


#include <CORBA.h>

#include <hello_impl.h>

#include <fstream.h>
int
main(int argc, char* argv[], char*[])
{
CORBA_ORB_var orb = CORBA_ORB_init(argc, argv); CORBA_BOA_var boa = orb -> BOA_init(argc, argv);
Hello_var p = new Hello_impl;

CORBA_String_var s = orb -> object_to_string(p); const char* refFile = "Hello.ref"; ofstream out(refFile);
out << s << endl;
out.close();
```

```
boa -> impl_is_ready(CORBA_ImplementationDef::_nil());

}
```

Client programs

```
public interface Hello extends org.omg.CORBA.Object {
void hello();
public void hello(); }
abstract public class _sk_Hello extends org.omg.CORBA.portable.Skeleton implements Hello {
protected _sk_Hello(java.lang.String name)
{

super(name)
; }
protected _sk_Hello() { super(); }
public java.lang.String[] _ids() { return __ids; }

private static java.lang.String[] __ids = { "IDL:Hello:1.0" }; public
org.omg.CORBA.portable.MethodPointer[] _methods()

{ org.omg.CORBA.portable.MethodPointer[] methods = { new
org.omg.CORBA.portable.MethodPointer("hello", 0, 0), }; return methods; }

public boolean _execute(org.omg.CORBA.portable.MethodPointer method,
org.omg.CORBA.portable.InputStream input, org.omg.CORBA.portable.OutputStream output) {
switch(method.interface_id)

{

case 0:

{
return _sk_Hello._execute(this, method.method_id, input, output);
}
}
throw new org.omg.CORBA.MARSHAL(); }
public static boolean _execute(Hello _self, int _method_id,
org.omg.CORBA.portable.InputStream _input, org.omg.CORBA.portable.OutputStream
_output)
{
```

```
switch(_method_id) { case 0: { _self.hello(); return false; } } throw new
org.omg.CORBA.MARSHAL(); } }
class hello_client {
public static void main( String args[] ) { try{
System.out.println( "Initializing the orb."); org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
IORHolder ior_holder = new IORHolder();
String iorString = ior_holder.readIORFile( "Hello.ref" );

org.omg.CORBA.Object object = orb.string_to_object( iorString ); Hello hello = HelloHelper.narrow(
object ); hello.hello();

} catch ( org.omg.CORBA.SystemException e ) {

System.err.println( "System Exception ");
System.err.println( e );} }}
```

Program6:

Write a program to Simulate the Functioning of Lamport's Logical Clock

Sol:-

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
        int n,e;
        cout<<"Enter the number of processes:->";
        cin>>n;
        cout<<"\nEnter the number of events by each process:->";
        cin>>e;

        cout<<"Enter the order of occurrence of events in Matrix along with type of event";
        cout<<"\nEnter r along with process for receivng process for e.g. t\n";
        string s[n][e];
        int timestamp[n][e]={0};
        for(int i=0;i<n;i++)
        {
                for(int j=0;j<e;j++)
                  cin>>s[i][j];

        }
int min=99,len,max=6,max1,max2,MAX;
while(min!=max)
{
for(int i=0;i<n;i++)
    {
      for(int j=0;j<e;j++)
       {
            if(s[i][j].length()<2&&int(s[i][j][0])-'0'<=min+1)
             {
          min=int(s[i][j][0])-'0';
             if(s[i][j]=="5")
               timestamp[i][j]=timestamp[i][j-1]+1;
             else
               timestamp[i][j]=timestamp[i][i]+1;
           cout<<"Time Stamp of Process "<<i+1<<" is "<<timestamp[i][j]<<"\n";
                  s[i][j]="9";
          }
```

```
        else if(s[i][j].length()==2&&int(s[i][j][0])-'0'<=min+1)
            {
                min=int(s[i][j][0])-'0';
            if(s[i][j]=="6r")
                    timestamp[i][j]=timestamp[i-1][j-1]+1;
                else
                    timestamp[i][j]=timestamp[i+1][j]+1;
                    max1=timestamp[i][j];
                     max2=timestamp[i][j]-1;
                if(max1>max2)
                    MAX=max1;
                else
                    MAX=max2;
                    MAX=MAX+1;
                timestamp[i][j]=MAX;
                    cout<<"Time Stamp of Process"<<i+1<<" is "<<timestamp[i][j]<<"\n";
                    s[i][j]="9";
                }

            }

        } }

    return 0;
}
```

OUTPUT:

```
C:\Users\Vineet\Documents\LamportClock.exe
Enter the number of processes:->3

Enter the number of events by each process:->2
Enter the order of occurrence of events in Matrix along with type of event
Enter r along with process for receivng process
1 3
4r 5
2 6r
Time Stamp of Process 1 is 1
Time Stamp of Process 3 is 1
Time Stamp of Process 1 is 2
Time Stamp of Process2 is 3
Time Stamp of Process 2 is 4
Time Stamp of Process3 is 5

--------------------------------
Process exited after 11.36 seconds with return value 0
Press any key to continue . . .
```

Program7:

Write a program to Implement Cristians Algorithm?

Sol:-

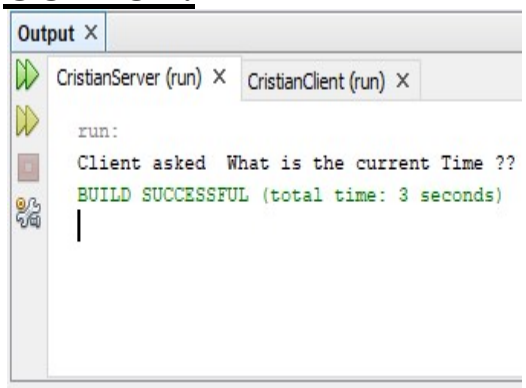
SERVER CODE:

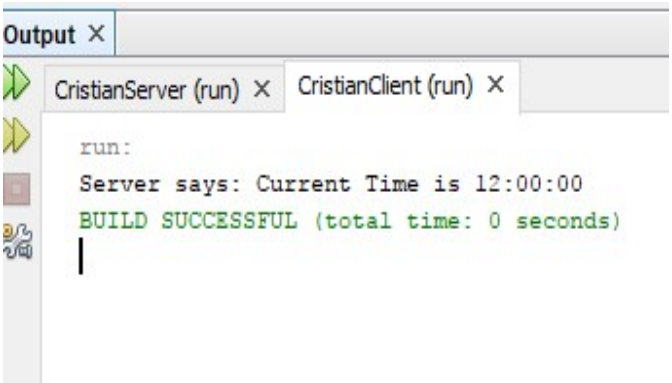
```
package cristianserver;
import java.io.*;
import java.net.*;
import java.text.SimpleDateFormat;
import java.time.LocalTime;
import java.util.Date;
public class CristianServer
{
public static void main(String[] args)
   {
      try
      {
         ServerSocket ss=new ServerSocket(1234);
         Socket s=ss.accept();
         DataOutputStream dout=new DataOutputStream(s.getOutputStream());
          DataInputStream dis=new DataInputStream(s.getInputStream());
         String time1="12:00:00";
         System.out.println("Client asked "+dis.readUTF());
         dout.writeUTF("Current Time is "+time1);
         dout.flush();
         dout.close();
          s.close();
          ss.close();
       }
      catch(Exception e)
      {
         System.out.println(e);
      }
   }

}
```

CLIENT CODE:

```
package cristianclient;
import java.io.*;
import java.net.*;
public class CristianClient
{
  public static void main(String[] args)
   {
     try
     {
        Socket s=new Socket("localhost",1234);
        DataInputStream dis=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());// recieve server input

        String str="",str1="";
        dout.writeUTF(" What is the current Time ??");
         str1=dis.readUTF();
        System.out.println("Server says: "+str1);

         dis.close();
         dout.close();
        s.close();
     }
     catch(Exception e)
     {
        System.out.println(e);
     }
   }

}
```

OUTPUT:





Program8:

Write a program to Implement Berkeley Algorithm?

Sol:

SERVER CODE:

```
package berkerlyserver;
import java.io.*;
import java.net.*;
import java.util.*;
import java.text.SimpleDateFormat;
public class BerkerlyServer
{

  public static void main(String[] args)
   {
      try
      {
         ServerSocket ss=new ServerSocket(1234);
         Socket s=ss.accept();  // establish connection
         DataInputStream dis=new DataInputStream(s.getInputStream());
         DataOutputStream dout=new DataOutputStream(s.getOutputStream());// recieve client input
         BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); // send input
stream to client
         String str="",str1="",str2="",k1="",time1="12:00:00";

          dout.writeUTF("My time is "+ time1+" Tell Me Your Time Client");

          str=dis.readUTF();

          System.out.println("Client Says My time is:"+str);
          k1=str;

          SimpleDateFormat format=new SimpleDateFormat("HH:mm:ss");
          Date d1=format.parse(time1);
          Date d2=format.parse(k1);
          long Diff;
          Diff=((d2.getTime()-d1.getTime())/60000);  // take average
          System.out.println(Diff);
          long avg;
          avg=Diff/2;


          dout.writeUTF("You should decrease your time by "+(Diff-avg));
```

```
            dout.flush();
            dis.close();
            dout.close();
            s.close();
            ss.close();
        }
       catch(Exception e)
       {
          System.out.println(e);
       }
    }

  }
```

CLIENT CODE:

```
package berkerlyclient;
import java.io.*;
import java.net.*;

public class BerkerlyClient
{
   public static void main(String[] args)
   {
      try
      {
         Socket s=new Socket("localhost",1234);
         DataInputStream dis=new DataInputStream(s.getInputStream());
         DataOutputStream dout=new DataOutputStream(s.getOutputStream());// recieve server input
         BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); // send server
stream to client
         String str="",str1="";
          str=dis.readUTF();
          System.out.println("Server says: "+str);
          dout.writeUTF("12:15:00");
          dout.flush();
          str1=dis.readUTF();
          System.out.println("Server says: "+str1);
          dis.close();
          dout.close();
         s.close();
      }
      catch(Exception e)
      {
         System.out.println(e);
      }
   }
   }
```

DATE:-

OUTPUT:

```
Output
BerkerlyServer (run)   BerkerlyClient (run)
run:
Client Says My time is:12:15:00
15
BUILD SUCCESSFUL (total time: 5 seconds)
```

```
Output
BerkerlyServer (run)   BerkerlyClient (run)
run:
Server says: My time is 12:00:00 Tell Me Your Time Client
Server says: You should decrease your time by 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

Program9:

Write a program to Simulate Distributed Mutual Exclusion Program using Centralized algorithm?

Sol:

```
#include<stdio.h>
#include<dos.h>
#include<time.h>
#include<conio.h>
int main()
{
int cs=0,pro=0;
double run=5;
char key='a';
time_t t1,t2;
printf("\nPress a key(except q) to enter a process into critical section.");
printf(" \nPress q at any time to exit.");
t1 = time(NULL) -5;
char ch;
while(key!='q')
{
while((ch=getche())!='s')
if(cs!=0)
{
t2 = time(NULL);
if(t2
-
t1 > run)
{
printf("Process%d ",pro-1);
printf(" exits critical section.\n");
cs=0;
}
}
key = getch();
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 sectio\n");
cs=1;
```

```
pro++;
t1 = time(NULL);
}
}
}
}
```

OUTPUT:

Select C:\Users\Vineet\Documents\aiyann.exe

```
Press a key(except q) to enter a process into critical section.
Press q at any time to exit.sProcess 0  entered critical sectio
aProcess0  exits critical section.
sProcess 1  entered critical sectio
sError: Another process is currently executing critical section Please wait till its execution is over.
sError: Another process is currently executing critical section Please wait till its execution is over.
qProcess1  exits critical section.
```

Program10:

Write a program to Simulate RING BASED election algorithm.

Sol:-

```
#include<string.h>
#include<iostream>
#include<stdio.h>
#include<stdlib.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;
	}
// sorting
	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");
                                  // cout<<"\n"<<i<<" "<<proc[i].id<<"deactivate\n";
                                      }
                 }
                                       break;
                break;
  }
 }
 return 0;
}
```

OUTPUT:

C:\Users\Vineet\Documents\Ring.exe

```
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    5

enter id of process    1

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

Program11:

Write a program to Simulate BULLY election algorithm.

Sol:-

```
#include<stdio.h>
#include<string.h>
#include<iostream>
#include<stdlib.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;
                   }
                   }
                  //cout<<"\n"<<pr;
                  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);

                   }
            }
        return 0;
}
```

OUTPUT:

```
C:\Users\Vineet\Documents\BullyAlgo.exe

 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
```

Program12:

Write a program to Simulate the Non Token based algorithm Ricart's Agrawala

Sol:-

```
import java.io.*;
public class RicartAgrawala {
        public boolean bRequestingCS;
        public int outstandingReplies;
        public int highestSeqNum;
        public int seqNum;
        public int nodeNum;
        public Driver driverModule;
        public PrintWriter[] w;
        public int channelCount = 3;
          public boolean[] replyDeferred;
  public RicartAgrawala(int nodeNum, int seqNum, Driver      driverModule)
{
                    bRequestingCS = false;
                    outstandingReplies = channelCount;
                    highestSeqNum = 0;
                   this.seqNum = seqNum;
                    this.driverModule = driverModule;
                    w = new PrintWriter[channelCount];
                    this.nodeNum = nodeNum;
                    replyDeferred = new boolean[channelCount];
        }
         public boolean invocation()
         {
         bRequestingCS = true;
         seqNum = highestSeqNum + 1;

        outstandingReplies = channelCount;
                for(int i = 1; i <= channelCount + 1; i++)
{
                            if(i != nodeNum)
                              {    requestTo(seqNum, nodeNum, i);
                              }
          }
                while(outstandingReplies > 0)
                            {
```

```
                    try
                {
                    Thread.sleep(5);                                    }
            catch(Exception e){
                }
                }
            return true;
        }



public void releaseCS() {

bRequestingCS = false;
  for(int i = 0; i < channelCount; i++){

     if(replyDeferred[i])
     {
        replyDeferred[i] = false;
        if(i < (nodeNum – 1))
        replyTo(i + 1);
        else
        replyTo(i + 2);
     }
  }
}
public void receiveRequest(int j, int k)
{
   System.out.println("Received request from node " + k);
   boolean bDefer = false;
   highestSeqNum = Math.max(highestSeqNum, j);
   bDefer = bRequestingCS && ((j > seqNum) || (j == seqNum && k > nodeNum));
   if(bDefer)
   {
      System.out.println("Deferred sending message to " + k);
      if(k > nodeNum)
         replyDeferred[k – 2] = true;
      else
      replyDeferred[k – 1] = true;
   }
   else
   {
      System.out.println("Sent reply message to " + k);
      replyTo(k);
```

```
    }
}
public void receiveReply()
{
    outstandingReplies = Math.max((outstandingReplies – 1), 0);
}
public void replyTo(int k)
{
    System.out.println("Sending REPLY to node " + k);
    if(k > nodeNum)
    {
        w[k–2].println("REPLY," + k);
    }
    else
    {
        w[k–1].println("REPLY," + k);
    }
}
public void requestTo(int seqNum, int nodeNum, int i)
{
    System.out.println("Sending REQUEST to node " + (((i))));
    if(i > nodeNum)
    {
        w[i–2].println("REQUEST," + seqNum + "," + nodeNum);
    }
    else
    {
        w[i–1].println("REQUEST," + seqNum + ",nodeNum);
    }
}
}
```

DATE:-

Program13:

Write a program to Simulate the Non Token based Algorithm Maekawa's

Sol:-

Maekawa.java

```
package Permission_Arbitre;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.LinkedList;


public class Maekawa extends Thread {
int[] ports = {1111,2222,3333,4444};
boolean voted=false,accessing=false;
private int answers = 0;
LinkedList<Client> Ri = new LinkedList<Client>();
LinkedList<Integer> onhold = new LinkedList<>();

int ID;
int port;

public static void main(String[] args) {
int port = Integer.parseInt(args[0]);
Maekawa p = new Maekawa(port);
p.start();
p.createServer();

}
```

```
public Maekawa(int port){
this.port=port;
this.ID=port;
}
public void run() {
PrintWriter pw;
try {
sleep(5000);
} catch (Exception e) {
e.printStackTrace();
}
for(int port:ports){
try {
if(port != ID){
Socket s = new Socket("127.0.0.1", port);
Client p = new Client(s, port);
pw = new PrintWriter(s.getOutputStream(), true);
pw.println(ID);
p.start();
}
} catch (Exception e) {
System.out.println("Error Connecting with Other processes");
}
}
while(true){
try {
System.in.read();
voted=true;
sendtoRi("ask");
System.out.println("Asking Processes... ");
```

```
while(true){
sleep(1500);
if(answers == ports.length-1)
{
accessing=true;
System.out.println("Accessing Critical Section...");
sleep(5000);
System.out.println("Done Working on Critical Section!");
sendtoRi("free");
if(onhold.isEmpty())
voted = false;
else{
int p = onhold.removeFirst();
sendTo(p,"ok");
}
answers = 0;
accessing=false;
break;
}

}
} catch (Exception e) {
e.getMessage();
}
}

}
public void createServer() {
try {
ServerSocket server = new ServerSocket(port);
Client p;

while (true) {
```

```
Socket s = server.accept();
BufferedReader input = new BufferedReader(new InputStreamReader(s.getInputStream()));
int id = Integer.parseInt(input.readLine());
p = new Client(s,id);
Ri.add(p);
System.out.println(id + " is Successfully Connected.");
sleep(500);
p.start();
}
} catch (Exception e) {
e.printStackTrace();
}
}
public void sendtoRi(String message) {
for (Client p : Ri)
p.sendMessage(ID+":"+message);
}
public void sendTo(int x,String message){
for (Client p : Ri)
if(p.getIdP() == x){
p.sendMessage(ID+":"+message);
break;
}
}
class Client extends Thread {
BufferedReader input;
PrintWriter output;
String msg;
int id;

public Client(Socket client, int id) {
```

```
this.id = id;
try {
output = new PrintWriter(client.getOutputStream(), true);
input = new BufferedReader(new InputStreamReader(client.getInputStream()));
} catch (IOException e) {
e.printStackTrace();
}
}
public int getIdP() {
return id;
}
public void sendMessage(String str) {
output.println(str);
}
public void run() {
while (true) {
try {
String msg = input.readLine();
System.out.println("Message Received : " + msg);
String messageArray[] = msg.split(":");

if (messageArray[1].equals("ask")) {
if(voted || accessing){
onhold.add(Integer.parseInt(messageArray[0]));
System.err.println(onhold.toString());
}
else
sendTo(Integer.parseInt(messageArray[0]),"ok");
}
else if(messageArray[1].equals("ok")){
answers++;
```

```
System.out.println("Incrementing Answer");
}
else if(messageArray[1].equals("free")){
if(onhold.isEmpty())
voted = false;
else{
int p = onhold.removeFirst();
sendTo(p,"ok");
}
}
} catch (IOException e) {
System.out.println(id+"Error! Socket will be closed immediatly");
break;
}

}
}

}
}
```