BABU BANARASI DAS UNIVERSITY
LUCKNOW

Department of Computer Science & Engineering

DISTRIBUTED SYSTEMS

Lab File

(BCS 2851)

Submitted To –

Mr. Abhishek Yadav

Assistant Professor

Dept of CSE, BBDU

Submitted By –

Mansi Kumari

B.Tech (CS-42)

Class Roll.no - 13

University Roll.no -1180432056

S.No	Name of Program	Sign

1	Design a distributed application using RMI for remote computation where client submits two strings to the server and server returns the concatenation of the given strings.	
2	Design a distributed application using socket. Application consists of a server which takes an integer value and returns the factorial to the client.	
3	Design an application using MapReduce to find the coolest year from the given dataset.	
4	Find list of users with maximum file size in the current working directory using mapReduce.	
5	Design a distributed client server application using threads in java.	
6	Design distributed application using RMI in which when a user sends a string, the server reverse it and send back to the client.	
7	Design a program for distributed system using Remote Method Invocation.	
8	Implementation of “Calculator” Service using JAVA RMI	
9	Write a program to simulate the functioning of Lamport’s logical clock	
10	Write a program to simulate the Distributed Mutual Exclusion	

Program 1- Design a distributed application using RMI for remote computation where client submits two strings to the server and server returns the concatenation of the given strings.

```
//RMIServer// import
java.sql.*; import
java.sql.Connection; import
java.rmi.*; import
java.rmi.Naming.*; import
java.rmi.server.*; import
java.rmi.registry.*; import
java.util.Vector;


interface DBInterface extends Remote
{
      public String input(String name1,String name2) throws
RemoteException;
}
public class Server extends UnicastRemoteObject implements DBInterface
{
      int flag=0,n,i,j;
String name3;     ResultSet
r;
      public Server() throws RemoteException{
            try{
                  System.out.println("Initializing Server\nServer Ready");
            }
            catch (Exception e)
            {
                  System.out.println("ERROR: " +e.getMessage());
            }
      }
      public static void main(String[] args)
```

```
      {
try
            {
                  Server rs=new Server();

      java.rmi.registry.LocateRegistry.createRegistry(1030).rebind("DBServ
",rs);


            }
            catch (Exception e)
            {
                  System.out.println("ERROR: " +e.getMessage());
            }
      }
      public String input(String name1,String name2)
      {
            try{
                  name3=name1.concat(name2);
            }
            catch (Exception e) {
                  System.out.println("ERROR: " +e.getMessage());
            }
            return name3;
      }
}
```

Client Code:

```
//RMIClient// import
java.sql.*; import
java.rmi.*; import
java.io.*; import
java.util.*; import
```

```
java.util.Vector.*; import
java.lang.*;

import java.rmi.registry.*;


public class Client{
      static String name1,name2,name3;
public static void main(String args[])
      {
            Client c=new Client();
            BufferedReader b = new BufferedReader(new
InputStreamReader(System.in));


             int ch;
 try {
                  Registry r1 = LocateRegistry.getRegistry ( "localhost",
1030);
                  DBInterface DI=(DBInterface)r1.lookup("DBServ");
                  do{
                       System.out.println("1.Send input
strings\n2.Display concatenated string \nEnter your choice");
                       ch= Integer.parseInt(b.readLine());
                       switch(ch){


                       case 1:
                             System.out.println(" \n Enter first
string:");
                             name1=b.readLine();
                             System.out.println(" \n Enter second
string:");
                             name2=b.readLine();
name3=DI.input(name1,name2);
                             break;
                  case 2:
```

```
                                System.out.println("\n Concatenated String
is : ");
                                int i=0;
                                System.out.println(" " +name3+"");
                                break;
```

```
                    }
                    }while(ch>0);
            }
            catch (Exception e)
            {
                                        System.out.println("ERROR: "
            +e.getMessage());
            }
```

Program -

```
        }
}
```

Program 2 - Design a distributed application using socket. Application consists of a server which takes an integer value and returns the factorial to the client.

SERVER

```
import java.net.*;
import java.io.*; public
class myserv{
public static void main(String ar[]){ try{
            DatagramSocket s =  new DatagramSocket(1234);
            while ( true ) {
            DatagramPacket packet = new DatagramPacket(new byte[1024],
1024);
            s.receive( packet );
            String message = new String(packet.getData(), 0, 0,
packet.getLength()); int res=1;
int ms=Integer.parseInt(message); for(int
i=1;i<=ms;i++) res=res*i;
String str1=res+" ";
  System.out.println( "Factorial of " +
message + " is " + str1);
}}
catch(Exception e){}
}}
```

CLIENT

```
import java.net.*; import
java.io.*; public class myclient{
public static void main(String ar[])
{   int myPort = 1234; try {

  DatagramSocket ds = new DatagramSocket();
  DatagramPacket pack;
  InetAddress addr = InetAddress.getLocalHost();
  BufferedReader b=new BufferedReader (new
InputStreamReader(System.in));
       {
         System.out.print("Enter the number to find factorial : ");
String message=b.readLine();             byte [] data = new byte [
message.length() ];          message.getBytes(0, data.length,
data, 0);
         pack =  new DatagramPacket(data, data.length, addr, myPort);
ds.send( pack );
  }  }
       catch ( IOException e )  {
System.out.println( e ); }  }}
```

3 Design an application using MapReduce to find the coolest year from the given dataset.

```
import java.io.IOException; import
java.util.*; import
org.apache.hadoop.fs.Path; import
org.apache.hadoop.conf.*; import
org.apache.hadoop.io.*; import
org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.input.TextInputFormat; import
```

Program -

```
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class TempMR2 {

public static class TempMap extends Mapper<LongWritable, Text, Text,
IntWritable> {

public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

String record = value.toString(); String[]
parts = record.split(",");
context.write(new Text(parts[0]), new
IntWritable(Integer.parseInt(parts[1])));
}
}
public static class TempReduce extends Reducer<Text, IntWritable, Text,
IntWritable> {
 public void reduce(Text key, Iterable<IntWritable> values,Context
context) throws IOException, InterruptedException {

int maxValue = 0;

//Looping and calculating Max for each year
for (IntWritable val : values) { maxValue
= Math.max(maxValue, val.get());
}
context.write(key, new IntWritable(maxValue));
}
}
public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();



Job job = new Job(conf, "tempmax");

job.setJarByClass(TempMR2.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setMapperClass(TempMap.class);

job.setReducerClass(TempReduce.class);
```

```
job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job,new Path(args[1]));

job.waitForCompletion(true);
   } }
```

4 Find list of users with maximum file size in the current working directory using mapReduce.

```
public class max {

public static class maxminmapper extends Mapper<LongWritable, Text, Text,
DoubleWritable> { Text t1 = new Text();

public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

String[] colvalue = value.toString().split(“,”); for

(int i = 0; i < colvalue.length; i++) {

t1.set(String.valueOf(i + 1));

context.write(t1, new DoubleWritable(Double.parseDouble(colvalue[i])));

} } }

public static class maxminReducer extends Reducer<Text, DoubleWritable,
Text, DoubleWritable> {

public void reduce(Text key, Iterable<DoubleWritable> values, Context
context) throws IOException, InterruptedExceptio double min =
Integer.MAX_VALUE, max = 0;
```

Program -

```
Iterator<DoubleWritable> iterator = values.iterator(); //Iterating


if (value > max) { //Finding max value
max = value; } }
context.write(new Text(key), new DoubleWritable(min)); context.write(new
Text(key), new DoubleWritable(max));
} }
public static void main(String[] args) throws Exception {
Path inputPath = new Path("hdfs://localhost:54310/home/sortinput");
Path outputDir =new Path("hdfs://localhost:54310/home/MaxMinOutput1");
Configuration conf = new Configuration();
Job job = new Job(conf, "Find Minimum and Maximum");
job.setJarByClass(maxmin.class); FileSystem fs =
FileSystem.get(conf);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(DoubleWritable.class);
job.setMapperClass(maxminmapper.class);
job.setReducerClass(maxminReducer.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.addInputPath(job, inputPath);
FileOutputFormat.setOutputPath(job, outputDir);
System.exit(job.waitForCompletion(true) ? 0 : 1);
      }
}
```

5 Design a distributed client server application using threads in java.

```
import java.io.*;
import java.text.*;
import java.util.*;
import java.net.*; //
Server class public
class Server
{
	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 = s.accept();
				System.out.println("A new client is connected : "
+ s);
				// obtaining input and out streams
```

Program -

```
                        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();
      }
```

Program -

```
	}
}
```

6 Design distributed application using RMI in which when a user sends a string, the server reverse it and send back to the client.

```
//RMIServer// import
java.sql.*; import
java.sql.Connection; import
java.rmi.*; import
java.rmi.Naming.*; import
java.rmi.server.*; import
java.rmi.registry.*; import
java.util.Vector;


interface DBInterface extends Remote
{
	public String input(String name1,String name2) throws
RemoteException;
}
public class Server extends UnicastRemoteObject implements DBInterface
{
	int flag=0,n,i,j;
String name3;	ResultSet
r;
	public Server() throws RemoteException{
```

```
	try{
		System.out.println("Initializing Server\nServer Ready");
	}
	catch (Exception e)
	{
		System.out.println("ERROR: " +e.getMessage());
	}
}
public static void main(String[] args)
```

Program -

```
	{
try
		{
			Server rs=new Server();


java.rmi.registry.LocateRegistry.createRegistry(1030).rebind("DBServ
",rs);


		}
		catch (Exception e)
		{
			System.out.println("ERROR: " +e.getMessage());
		}
	}
	public String input(String name1)
	{
		try{
			StringBuilder input1 = new StringBuilder();
			 // append a string into StringBuilder input1
			  input1.append(name1);
			  // reverse StringBuilder input1
			input1.reverse();
			return input1;
		}
		catch (Exception e) {
			System.out.println("ERROR: " +e.getMessage());
		}}}
```

7 Design a program for distributed system using Remote Method Invocation.

```
import java.rmi.registry.Registry;  import
java.rmi.registry.LocateRegistry;  import
java.rmi.RemoteException;  import
java.rmi.server.UnicastRemoteObject;

public class Server extends ImplExample {
public Server() {}
  public static void main(String args[]) {
try {
       ImplExample obj = new ImplExample();
       Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
Registry registry = LocateRegistry.getRegistry();         registry.bind("Hello",
stub);
       System.err.println("Server ready");
     } catch (Exception e) {
       System.err.println("Server exception: " + e.toString());
       e.printStackTrace();
     } }}
```

Client: import

```
java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class Client {      private Client() {}
  public static void main(String[] args) {
try {
       Registry registry = LocateRegistry.getRegistry(null);
```

Program -

```
    Hello stub = (Hello) registry.lookup("Hello");
    stub.printMsg();
        } catch (Exception e) {
    System.err.println("Client exception: " + e.toString());
    e.printStackTrace();
  }}}
```

8 Implementation of calculator using RMI in java.

Additon.java import

java.rmi.Remote;

```
public interface AddInterface extends Remote {       //
Declaring the method prototype       public int add(int x,
int y) throws RemoteException;
}
```

Subtraction.java import

```
java.rmi.Remote;
public interface SubInterface extends Remote {       //
Declaring the method prototype       public int sub(int x,
int y) throws RemoteException;
}
```

Implimentation.java

```
import java.rmi.*; import
java.rmi.server.*;

public class Impl extends UnicastRemoteObject
implements AddInterface, SubInterface
 {
        public Impl() throws Exception { super(); }
public int add(int x, int y) { return x + y; }       public int
sub(int x, int y) { return x - y; }
}
```

Server.java import

```
java.rmi.*; import
java.rmi.registry.*; public
class Server {
        public static void main(String[] args) throws Exception
        {
                Impl obj = new Impl();
```

Program -

```
                Naming.rebind("ADD", obj);
                System.out.println("Server Started");
        }
}
```

Client.java

```
// Program for client application
import java.rmi.*; import
java.util.*; public class Client {
        public static void main(String[] args) throws Exception
        {
                Scanner sc = new Scanner(System.in);
        while (true) {
                        // User Menu
                        System.out.println(
                                "\n1.Addition\n2.Subtraction\n.Exit");
                System.out.println("Enter the option:");
int opt = sc.nextInt();                 if (opt == 5) {
break;
                        }
                        System.out.println(
"Enter the the first number:");                         int a =
sc.nextInt();
  System.out.println("Enter the second number:");    int b = sc.nextInt();
```

```
int n; switch
(opt) { case
1:
                        AddInterface obj
 = (AddInterface)Naming.lookup("ADD");  n = obj.add(a,
b);
                                System.out.println("Addition= " + n);
        break;
case 2:
                        SubInterface obj1
 = (SubInterface)Naming.lookup("ADD");  n = obj1.sub(a,
b);
                                System.out.println("Subtraction= " + n);
                break;
}       }
```

```
	}
}
```

Program 9 - Write a program to simulate the functioning of Lamport's logical clock.

```
#include <bits/stdc++.h>
using namespace std; int
max1(int a, int b)
{
	// Return the greatest of th two
if (a > b)
		return a;
else
return b;
}

// Function to display the logical timestamp void
display(int e1, int e2,
```

```
                  int p1[5], int p2[3])
{
      int i;

      cout << "\nThe time stamps of "
            "events in P1:\n";

      for (i = 0; i < e1; i++) {
cout << p1[i] << " ";
      }

      cout << "\nThe time stamps of "
            "events in P2:\n";

      // Print the array p2[]
for (i = 0; i < e2; i++)
cout << p2[i] << " "; }

// Function to find the timestamp of events void
lamportLogicalClock(int e1, int e2,
                                    int m[5][3])
{
      int i, j, k, p1[e1], p2[e2];

      // Initialize p1[] and p2[]
for (i = 0; i < e1; i++)
p1[i] = i + 1;

      for (i = 0; i < e2; i++)
p2[i] = i + 1;    cout << "\t";
for (i = 0; i < e2; i++)
cout << "\te2" << i + 1;
```

```
      for (i = 0; i < e1; i++) {
cout << "\n e1" << i + 1 << "\t";         for
(j = 0; j < e2; j++)                cout <<
m[i][j] << "\t";
      }

      for (i = 0; i < e1; i++) {          for (j = 0; j
< e2; j++) {                   if (m[i][j] == 1) {
            p2[j] = max1(p2[j], p1[i] + 1);
      for (k = j + 1; k < e2; k++)
      p2[k] = p2[k - 1] + 1;
                  }
                  if (m[i][j] == -1) {
p1[i] = max1(p1[i], p2[j] + 1);
for (k = i + 1; k < e1; k++)
p1[k] = p1[k - 1] + 1;
                  }
            }
      }
      display(e1, e2, p1, p2);
}

// Driver Code
int main() {
      int e1 = 5, e2 = 3,
m[5][3];    m[0][0] = 0;
m[0][1] = 0;      m[0][2] = 0;
m[1][0] = 0;      m[1][1] = 0;
m[1][2] = 1;      m[2][0] = 0;
m[2][1] = 0;      m[2][2] = 0;
m[3][0] = 0;      m[3][1] = 0;
m[3][2] = 0;      m[4][0] = 0;
m[4][1] = -1;     m[4][2] = 0;
```

```
      // Function Call
lamportLogicalClock(e1, e2, m);


      return 0;
}
```

Program 10 - Write a program to simulate the Distributed Mutual Exclusion

```
import java.util.*; public
class Mutex {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
int opt0,opt1;        int p1 = 1;        int
p2 = 2;         int p3 = 3;         int flag
= 0;         int cs  =  0;
        Queue<Integer> q = new LinkedList<>();
do
        {
            System.out.println("....menu...");
            System.out.println("1.Request the critical section");
System.out.println("2.Release the critical section");
System.out.println("3.Exit");             opt0 = sc.nextInt();
switch(opt0)
            {
case 1:
                {
                    System.out.println("Select the process.");
                    System.out.println("1.p1");
                    System.out.println("2.p2");
System.out.println("3.p3");                     opt1
= sc.nextInt();                     switch(opt1)
                    {
case 1:
{
if(flag==0)
{
```

```
cs = 1;
flag = 1;
                            }
else
                            {
                                System.out.println("process p"+cs+"is
already in critical section.");
                                q.add(p1);
                            }

                            System.out.println("System Status:");
                            System.out.println("critical section is
occupoied by:"+cs);
                            System.out.println("process waiting is: "+q);
break;                          }                           case 2:
{                           if(flag==0)                             {
cs  = 2;                                flag = 1;
                            }
else
                            {
                                 System.out.println("process p"+cs+"is
                                    already in critical section.");
                                q.add(p2);
                            }
                            System.out.println("System Status:");
                            System.out.println("critical section is
occupoied by:"+cs);
                            System.out.println("process waiting is: "+q);
break;                          }                           case 3:
{                           if(flag==0)                             {
cs  = 3;                                flag = 1;
                            }
else
                            {
                                 System.out.println("process p"+cs+"is
                                    already in critical section.");
```

```
                                q.add(p3);
                            }
                            System.out.println("System Status:");
System.out.println("critical section is
                              occupoied by:"+cs);
                            System.out.println("process waiting is: "+q);
break;
                        }
}
break;                 }
case 2:
                {
                    System.out.println("the process p"+cs+"is removed from
                        section.");
                    if(!q.isEmpty())
                    {
                        cs = q.peek();
                        q.remove();
                        System.out.println("System status:");
                        System.out.println("Critical Section occupied by
p"+cs);
                    }
else
                    {
                        System.out.println("No Process is waiting in the
                        queue");
                        flag = 0;
                    }}
case 3:
break;
            }}while(3!=opt0);}}
```