Assignment 1:

Problem Statement: Implement multi-threaded client/server process communication using RMI

Codes:

Client.java:

```
import  java.rmi.*;
import java.util.Scanner;


public class Client{
	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);

		try{
			String serverURL = "rmi://localhost/Server";
			Serverintf  serverintf = (Serverintf)
Naming.lookup(serverURL);

			System.out.print("Enter First Number: ");
			double num1 = sc.nextDouble();
			System.out.print("Enter Second Number: ");
			double num2 = sc.nextDouble();

			System.out.println("First Number is: " + num1);
			System.out.println("Second Number is: " + num1);


			System.out.println("----------------------------------Results-
----------------------------------------------");
			System.out.println("Addition is: " +
serverintf.addition(num1,num2));
			System.out.println("Subtraction is: " +
serverintf.subtraction(num1,num2));
			System.out.println("Multiplication is: " +
serverintf.multiplication(num1,num2));
			System.out.println("Division is: " +
serverintf.division(num1,num2));
		}catch(Exception e){
			System.out.println("Exception Occurred at
Client!" + e.getMessage());
		}
	}
}
```

Server.java:

```
import java.rmi.*;

public class Server{
	public static void main(String[] args){

		try{
			Serverimpl serverimpl = new Serverimpl();
			Naming.rebind("Server",serverimpl);
			System.out.println("Server started...");
		}catch(Exception e){
```

```
                        System.out.println("Eception Occurred at server!"
+ e.getMessage());
                }
        }
}
```

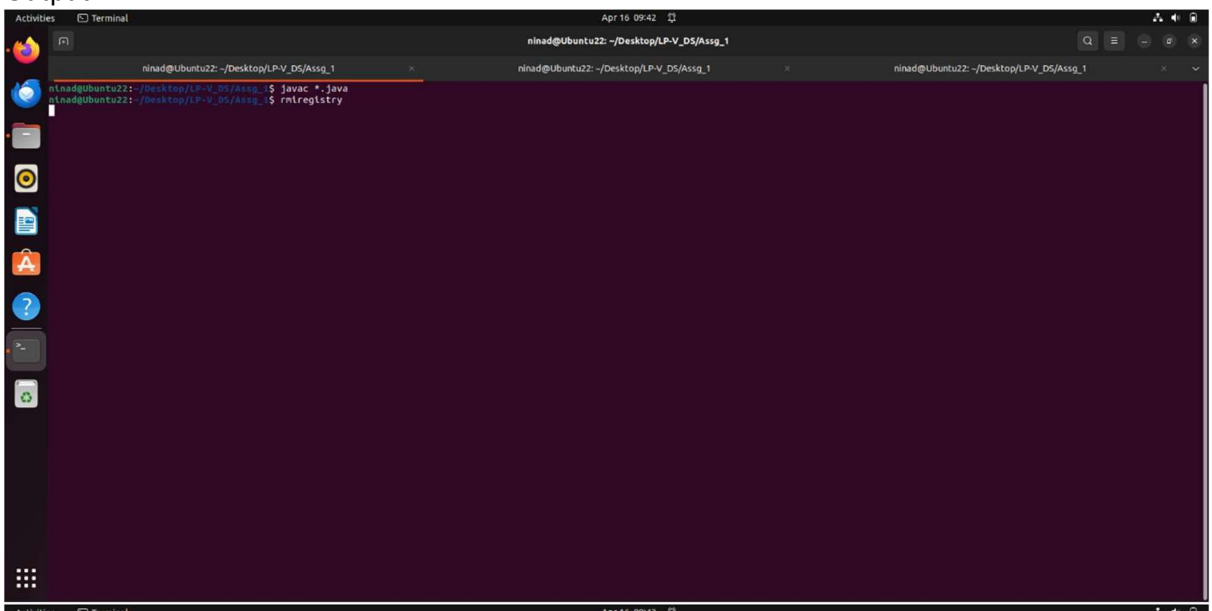
Serverimpl.java:

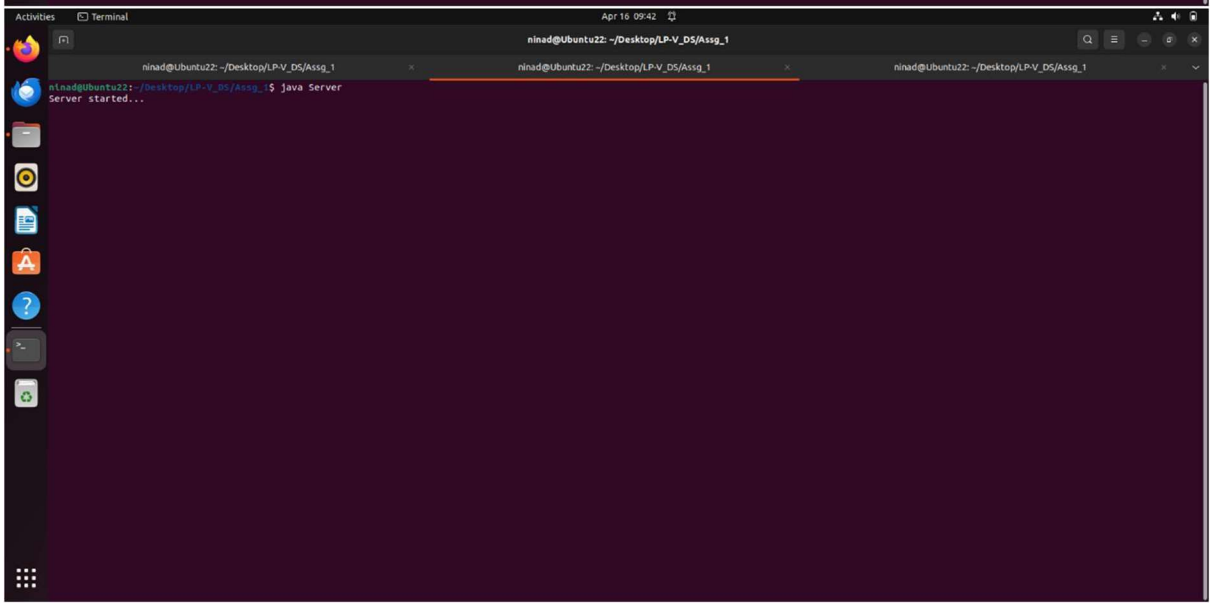
```
import  java.rmi.*;
import java.rmi.server.*;

public class Serverimpl extends UnicastRemoteObject implements
Serverintf{

        public Serverimpl() throws RemoteException{

        }
        public double addition(double num1, double num2) throws
RemoteException{
                return num1 + num2;
        }
        public double subtraction(double num1, double num2) throws
RemoteException{
                return num1 - num2;
        }
        public double multiplication(double num1, double num2) throws
RemoteException{
                return num1 * num2;
        }
        public double division(double num1, double num2) throws
RemoteException{
                if(num2 != 0){
                        return num1 / num2;
                }
                else{
                        System.out.println("Cannot divide a number by
zero!");
                }
                return num1/num2;
        }
}
```

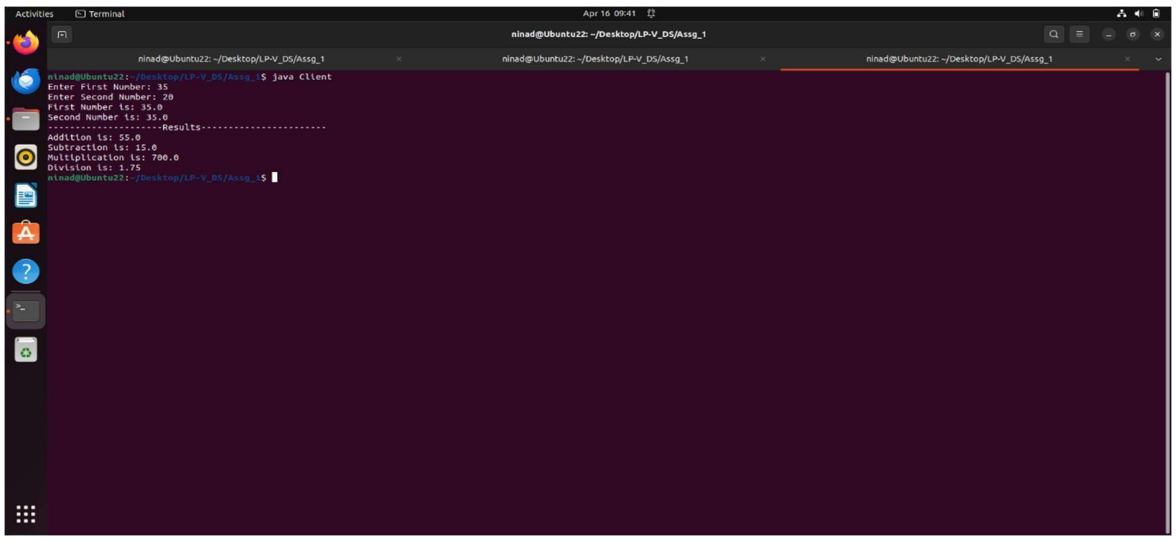
Serverintf.java:

```
import java.rmi.*;


interface Serverintf extends Remote{
        //Syntax for method declaration: access_specifier return_type
method_name(arguments){return_Value}

        public double addition(double num1, double num2) throws
RemoteException;
        public double subtraction(double num1, double num2) throws
RemoteException;
        public double multiplication(double num1, double num2) throws
RemoteException;
        public double division(double num1, double num2) throws
RemoteException; }
```

Output:


Activities
Terminal
Apr 16 09:42
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22:~/Desktop/LP-V_DS/Assg_1$ javac *.java
ninad@Ubuntu22:~/Desktop/LP-V_DS/Assg_1$ rmiregistry


Activities
Terminal
Apr 16 09:42
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22:~/Desktop/LP-V_DS/Assg_1$ java Server
Server started...


Activities
Terminal
Apr 16 09:41
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22: ~/Desktop/LP-V_DS/Assg_1
ninad@Ubuntu22:~/Desktop/LP-V_DS/Assg_1$ java Client
Enter First Number: 35
Enter Second Number: 20
First Number is: 35.0
Second Number is: 35.0
----------Results----------
Addition is: 55.0
Subtraction is: 15.0
Multiplication is: 700.0
Division is: 1.75
ninad@Ubuntu22:~/Desktop/LP-V_DS/Assg_1$

Assignment 2:

Problem Statement: Develop any distributed application using CORBA to demonstrate object brokering (Calculator).

Codes:

ReverseModule.idl:

```
module ReverseModule{

      interface Reverse

      {

            string reverse_string(in string str);

      }

}
```

ReverseServer.java:

```
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;

class ReverseServer
{
       public static void main(String[] args)
       {
              try
              {
                     org.omg.CORBA.ORB orb =
org.omg.CORBA.ORB.init(args,null);

                     POA rootPOA =
POAHelper.narrow(orb.resolve_inital_refernces("RootPOA"));
                     rootPOA.the_POAManager().activate();

                     Reverseimpl rvr = new Reverseimpl();

                     org.omg.CORBA.ORB.Object ref =
rootPOA.servant_to_referncec(rvr);

                     System.out.println("Step 1");
                     Reverse h_ref =
ReverseModule.ReverseHelper.narroe(ref);

                     System.out.println("Step 2");
                     org.omg.CORBA.ORB.Object objRef =
orb.resolve_initial_references("NameService");

                     System.out.println("Step 3");
                     NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);

                     System.out.println("Step 4");
                     String name = "Reverse";
                     NameComponent path[] = ncRef.to_name(name);
```

```
                        ncRef.rebind(path,h_ref);

                        System.out.println("Reverse Server Reading and
Waiting...");
                        orb.run();
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Reverseimpl.java:

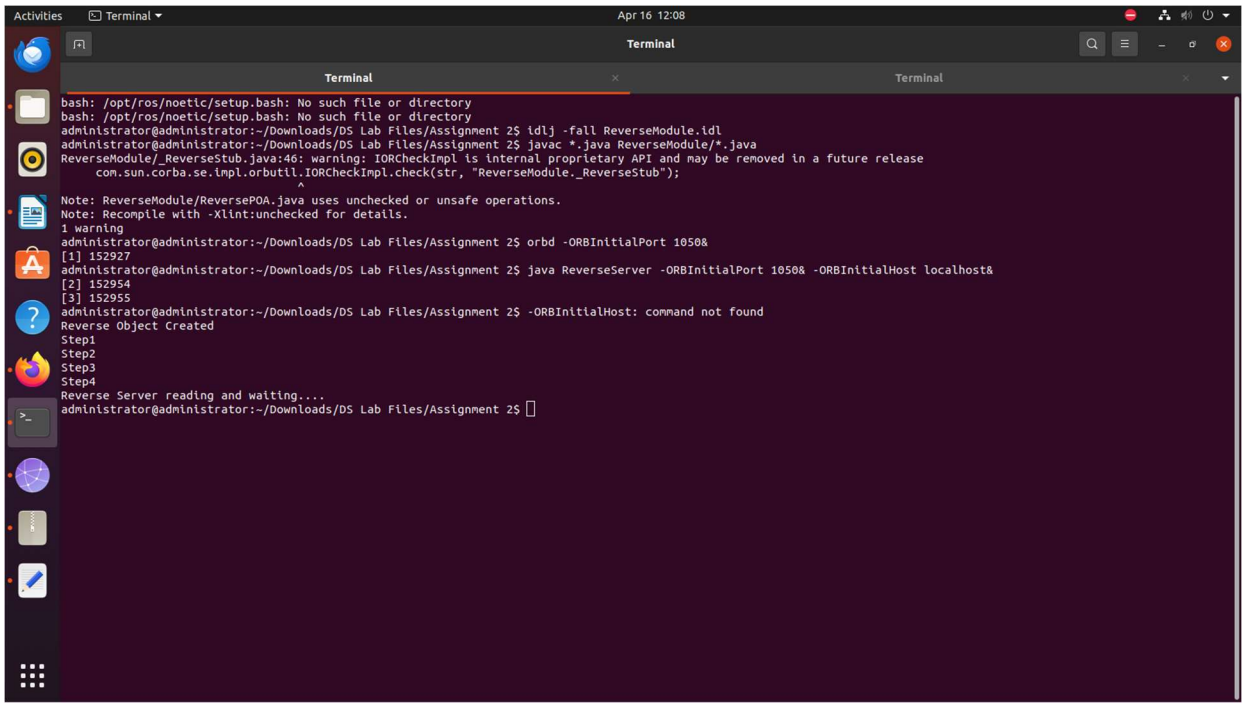
```
import ReverseModule.ReversePOA;
import java.lang.String;

class Reverseimpl extends ReversePOA
{
        Reverseimpl()
        {
                super();
                System.out.println("Reverse Object Created");
        }

        public String reverse_string(String name)
        {
                StringBuffer str = new StringBuffer(name);
                str.reverse();
                return(("Server send: " + str));
        }
}
```

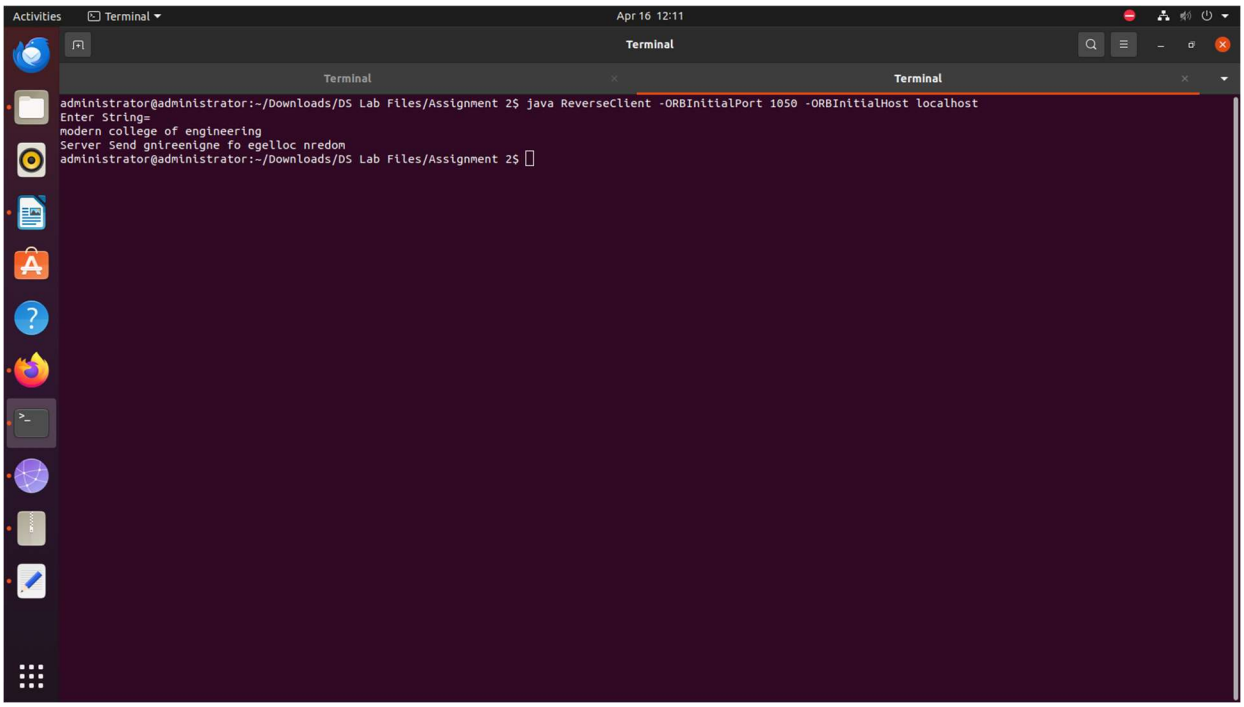
ReverseClient.java:

```
import ReverseModule.*;
import org.omg.CosNaming.*;
import org.omg.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.io.*;

class ReverseClient
{
        public static void main(String[] args)
        {
                Reverse Reverseimpl = null;

                try
                {
                        org.omg.CORBA.ORB orb =
org.omg.CORBA.ORB.init(args,null);

                        org.omg.CORBA.ORB.Object objRef =
orb.resolve_initial_references("NameService");
                        NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);

                        String name = "Reverse";
```

```
                        Reverseimpl =
ReverseHelper.narrow(ncRef.resolve_str(name));

                        System.out.println("Enter String: ");
                        BufferReader br = new BufferReader(new
InputStreamReader(System.in));
                        String str = br.readLine();

                        String tempStr = Reverseimpl.reverse_string(str);

                        System.out.println(tempStr);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Output:




Activities
Terminal
Apr 16 12:11
Terminal
Terminal
Terminal
administrator@administrator:~/Downloads/DS Lab Files/Assignment 2$ java ReverseClient -ORBInitialPort 1050 -ORBInitialHost localhost
Enter String=
modern college of engineering
Server Send gnireenigne fo egelloc nredom
administrator@administrator:~/Downloads/DS Lab Files/Assignment 2$

Assignment 3:

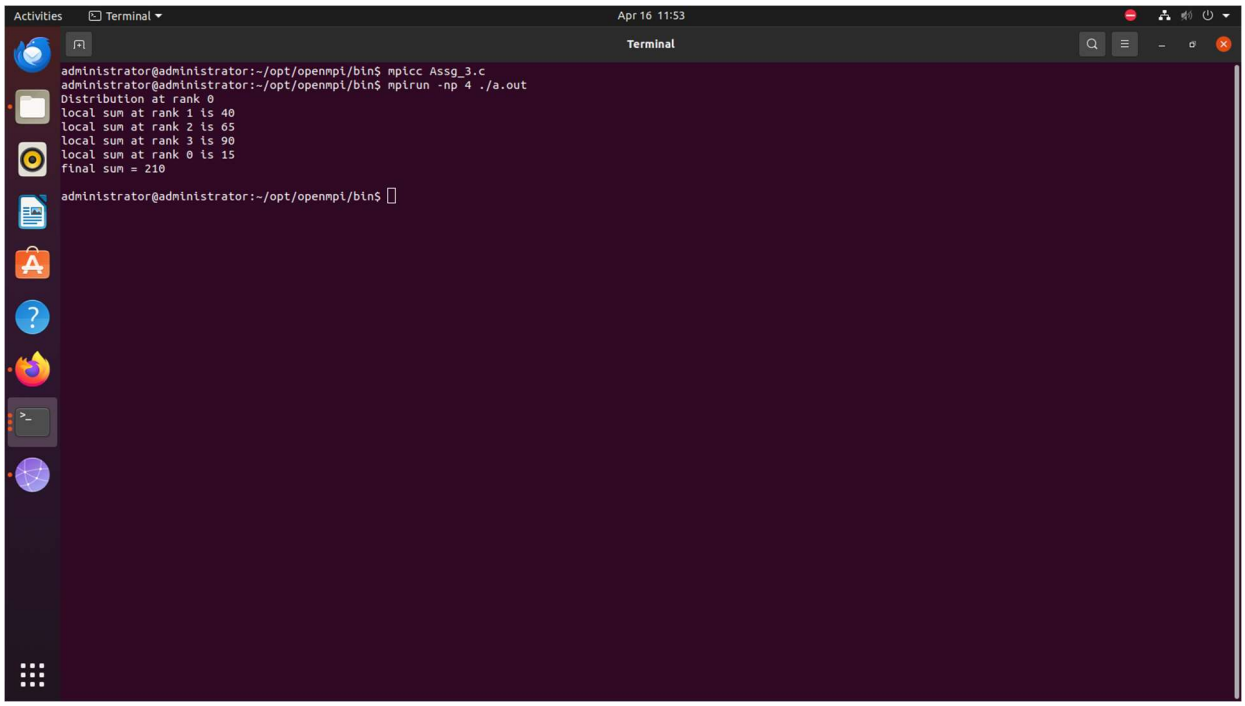
Problem Statement: Develop a distributed system, to find sum of N elements in an array by distributing N/n elements to n number of processors MPI or OpenMP. Demonstrate by displaying the intermediate sums calculated at different processors.

Codes:

Assg_3.c:

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char* argv[])
{
	int rank, size;
	int num[20]; //N=20, n=4
	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	MPI_Comm_size(MPI_COMM_WORLD, &size);
	for(int i = 0;i < 20;i++)
		num[i] = i + 1;
	if(rank == 0){
		int s[4];
		printf("Distribution at rank %d \n", rank);
		for(int i = 1;i < 4;i++)
			MPI_Send(&num[i * 5], 5, MPI_INT, i, 1,
MPI_COMM_WORLD); //N/n i.e. 20/4=5
		int sum = 0, local_sum = 0;
		for(int i = 0;i < 5;i++)
		{
			local_sum = local_sum + num[i];
		}
		for(int i = 1;i < 4;i++)
		{
			MPI_Recv(&s[i], 1, MPI_INT, i, 1, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
		}
		printf("local sum at rank %d is %d\n", rank,local_sum);
		sum=local_sum;
		for(int i = 1;i < 4;i++)
			sum = sum + s[i];
		printf("final sum = %d\n\n",sum);
	}
	else
	{
		int k[5];
		MPI_Recv(k, 5, MPI_INT, 0, 1, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
		int local_sum = 0;
		for(int i = 0;i < 5;i++)
		{
			local_sum = local_sum + k[i];
		}
		printf("local sum at rank %d is %d\n", rank, local_sum);
		MPI_Send(&local_sum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
	}
	MPI_Finalize();
	return 0;
}
```

Output:


Activities
Terminal
Apr 16 11:53
Terminal
administrator@administrator:~/opt/openmpi/bin$ mpicc Assg_3.c
administrator@administrator:~/opt/openmpi/bin$ mpirun -np 4 ./a.out
Distribution at rank 0
local sum at rank 1 is 40
local sum at rank 2 is 65
local sum at rank 3 is 90
local sum at rank 0 is 15
final sum = 210
administrator@administrator:~/opt/openmpi/bin$

Assignment 4:

Problem Statement: Implement Berkeley algorithm for clock synchronization.

Codes:

Server.py:

```
# Python3 program imitating a clock server

from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time
# datastructure used to store client address and clock data
client_data = {}
''' nested thread function used to receive
      clock time from a connected client '''
def startReceivingClockTime(connector, address):
      while True:
            # receive clock time
            clock_time_string = connector.recv(1024).decode()
            clock_time = parser.parse(clock_time_string)
            clock_time_diff = datetime.datetime.now() - \

      clock_time

            client_data[address] = {
                        "clock_time"      : clock_time,
                        "time_difference" : clock_time_diff,
                        "connector"  : connector
                        }

            print("Client Data updated with: "+ str(address),
                                                                  end =
"\n\n")
            time.sleep(5)


''' master thread function used to open portal for
      accepting clients over given port '''
def startConnecting(master_server):

      # fetch clock time at slaves / clients
      while True:
            # accepting a client / slave clock client
            master_slave_connector, addr = master_server.accept()
            slave_address = str(addr[0]) + ":" + str(addr[1])

            print(slave_address + " got connected successfully")

            current_thread = threading.Thread(
                              target = startReceivingClockTime,
```

```
                                    args = (master_slave_connector,
                                                          slave_address,
))
            current_thread.start()


# subroutine function used to fetch average clock difference
def getAverageClockDiff():

      current_client_data = client_data.copy()

      time_difference_list = list(client['time_difference']
                                                for client_addr, client
                                                      in
client_data.items())


      sum_of_clock_difference = sum(time_difference_list, \
                                                datetime.timedelta(0, 0))

      average_clock_difference = sum_of_clock_difference \
                                                            /
len(client_data)

      return average_clock_difference


''' master sync thread function used to generate
      cycles of clock synchronization in the network '''
def synchronizeAllClocks():

      while True:

            print("New synchronization cycle started.")
            print("Number of clients to be synchronized: " + \

      str(len(client_data)))

            if len(client_data) > 0:

                  average_clock_difference = getAverageClockDiff()

                  for client_addr, client in client_data.items():
                        try:
                              synchronized_time = \
                                    datetime.datetime.now() + \

      average_clock_difference

                              client['connector'].send(str(
                                          synchronized_time).encode())

                        except Exception as e:
                              print("Something went wrong while " + \
```

```
                                    "sending synchronized time " + \
                                    "through " + str(client_addr))

            else :
                  print("No client data." + \
                                    " Synchronization not applicable.")

            print("\n\n")

            time.sleep(5)


# function used to initiate the Clock Server / Master Node
def initiateClockServer(port = 8080):

      master_server = socket.socket()
      master_server.setsockopt(socket.SOL_SOCKET,
                                                socket.SO_REUSEADDR, 1)

      print("Socket at master node created successfully\n")

      master_server.bind(('', port))

      # Start listening to requests
      master_server.listen(10)
      print("Clock server started...\n")

      # start  making  connections
      print("Starting to make connections...\n")
      master_thread = threading.Thread(
                                    target = startConnecting,
                                    args = (master_server, ))
      master_thread.start()

      # start synchronization
      print("Starting synchronization parallelly...\n")
      sync_thread = threading.Thread(
                                    target = synchronizeAllClocks,
                                    args = ())
      sync_thread.start()



# Driver function
if__name__ == '__main__':

      # Trigger the Clock Server
      initiateClockServer(port = 8080)
```

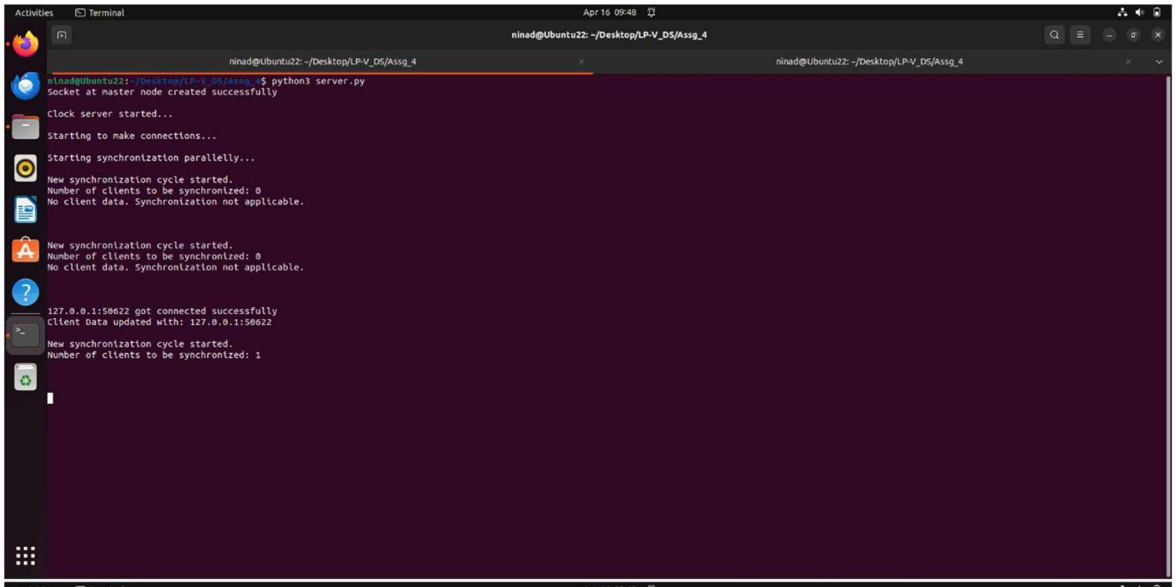
Client.py:

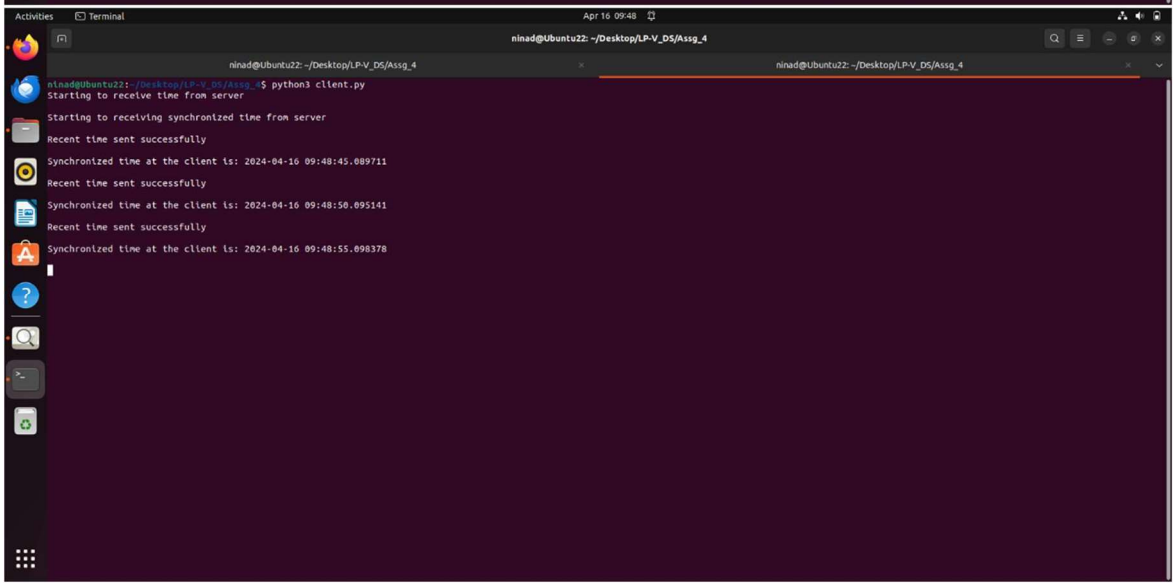
```
# Python3 program imitating a client process

from timeit import default_timer as timer
```

```
from dateutil import parser
import threading
import datetime
import socket
import time


# client thread function used to send time at client side
def startSendingTime(slave_client):

      while True:
            # provide server with clock time at the client
            slave_client.send(str(
                              datetime.datetime.now()).encode())

            print("Recent time sent successfully",
                                                      end = "\n\n")
            time.sleep(5)


# client thread function used to receive synchronized time
def startReceivingTime(slave_client):

      while True:
            # receive data from the server
            Synchronized_time = parser.parse(
                                    slave_client.recv(1024).decode())

            print("Synchronized time at the client is: " + \

      str(Synchronized_time),
                                                   end = "\n\n")


# function used to Synchronize client process time
def initiateSlaveClient(port = 8080):

      slave_client = socket.socket()

      # connect to the clock server on local computer
      slave_client.connect(('127.0.0.1', port))

      # start sending time to server
      print("Starting to receive time from server\n")
      send_time_thread = threading.Thread(
                              target = startSendingTime,
                              args = (slave_client, ))
      send_time_thread.start()


      # start receiving synchronized from server
      print("Starting to receiving " + \
                                    "synchronized time from server\n")
      receive_time_thread = threading.Thread(
```

```
                                target = startReceivingTime,
                                args = (slave_client, ))
        receive_time_thread.start()


# Driver function
if __name__ == '__main__':

        # initialize the Slave / Client
        initiateSlaveClient(port = 8080)
```

Output:





Assignment 5:

Problem Statement: Implement token ring based mutual exclusion algorithm.

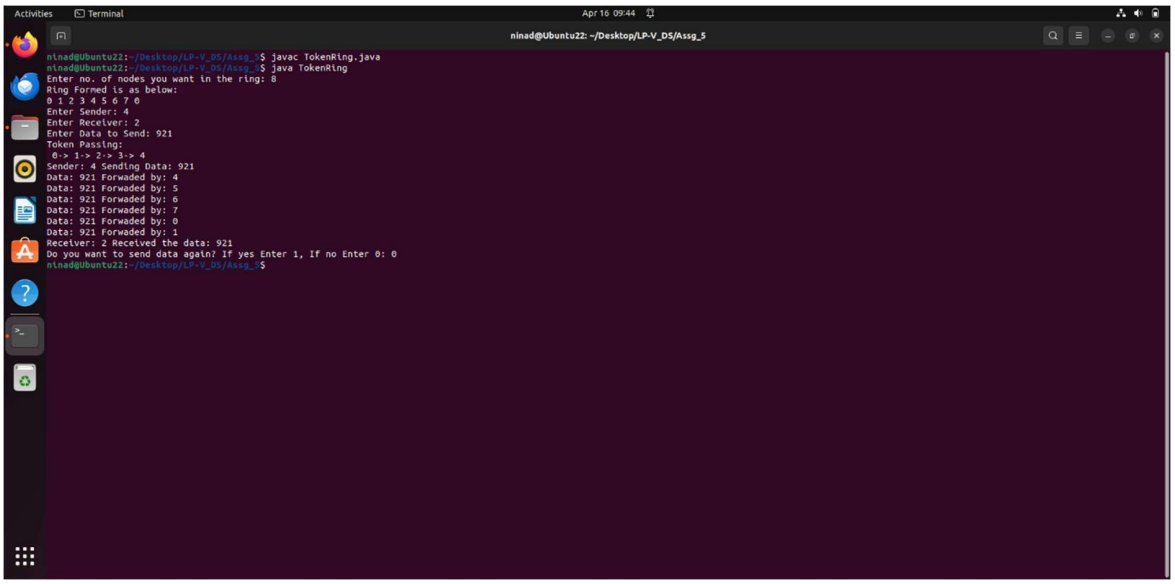
Codes:

TokenRing.java:

```
import java.util.*;


public class TokenRing{

	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);

		System.out.print("Enter no. of nodes you want in the
ring: ");
		int n = sc.nextInt();

		System.out.println("Ring Formed is as below: ");
		for(int i=0; i<n; i++){
			System.out.print(i + " ");
		}

		System.out.println("0");

		int choice = 0;

		do{
			System.out.print("Enter Sender: ");
			int sender = sc.nextInt();

			System.out.print("Enter Receiver: ");
			int receiver = sc.nextInt();

			System.out.print("Enter Data to Send: ");
			int data = sc.nextInt();

			int token = 0;
			System.out.println("Token Passing: ");


			for(int i=token; i<sender; i++){
				System.out.print(" " + i + "->");

			}

			System.out.println(" " + sender);
			System.out.println("Sender: " + sender + "
Sending Data: " + data);
			for(int i=sender; i!=receiver; i = (i+1)%n){
				System.out.println("Data: " + data + "
Forwaded by: " + i);
			}
```

```
                        System.out.println("Receiver: " + receiver + "
Received the data: " + data);
                        token = sender;

                        System.out.print("Do you want to send data again?
If yes Enter 1, If no Enter 0: ");
                        choice = sc.nextInt();
                }while(choice == 1);
        }
}
```

Output:



Assignment 6:

Problem Statement: Implement Bully and Ring algorithm for leader election.

Codes:

BullyAlgoExample.java:

```
import java.io.*;
import java.util.Scanner;
// create class BullyAlgoExample to understand how bully
algorithms works
class BullyAlgoExample{
// declare variables and arrays for process and their
status
static int numberOfProcess;
static int priorities[] = new int[100];
static int status[] = new int[100];
static int cord;
// main() method start
public static void main(String args[])throws IOException
// handle IOException
{
// get input from the user for the number of processes
System.out.println("Enter total number of processes:");
// create scanner class object to get input from user
Scanner sc = new Scanner(System.in);
numberOfProcess = sc.nextInt();
int i;
// use for loop to set priority and status of each
process
for(i = 0; i<numberOfProcess; i++)
{
System.out.println("Status for process "+(i+1)+":");
status[i] = sc.nextInt();
System.out.println("Priority of process "+(i+1)+":");
priorities[i] = sc.nextInt();
}
System.out.println("Enter proces which will initiate
election");
int ele = sc.nextInt();
```

```
sc.close();
// call electProcess() method
electProcess(ele);
System.out.println("After electing process the final
coordinator is "+cord);
}
// create electProcess() method
static void electProcess(int ele)

{
ele = ele - 1;
cord = ele + 1;
for(int i = 0; i<numberOfProcess; i++)
{
if(priorities[ele]<priorities[i])
{
System.out.println("Election message is sent from
"+(ele+1)+" to "+(i+1));
if(status[i]==1)
electProcess(i+1);
}
}
}
}
```

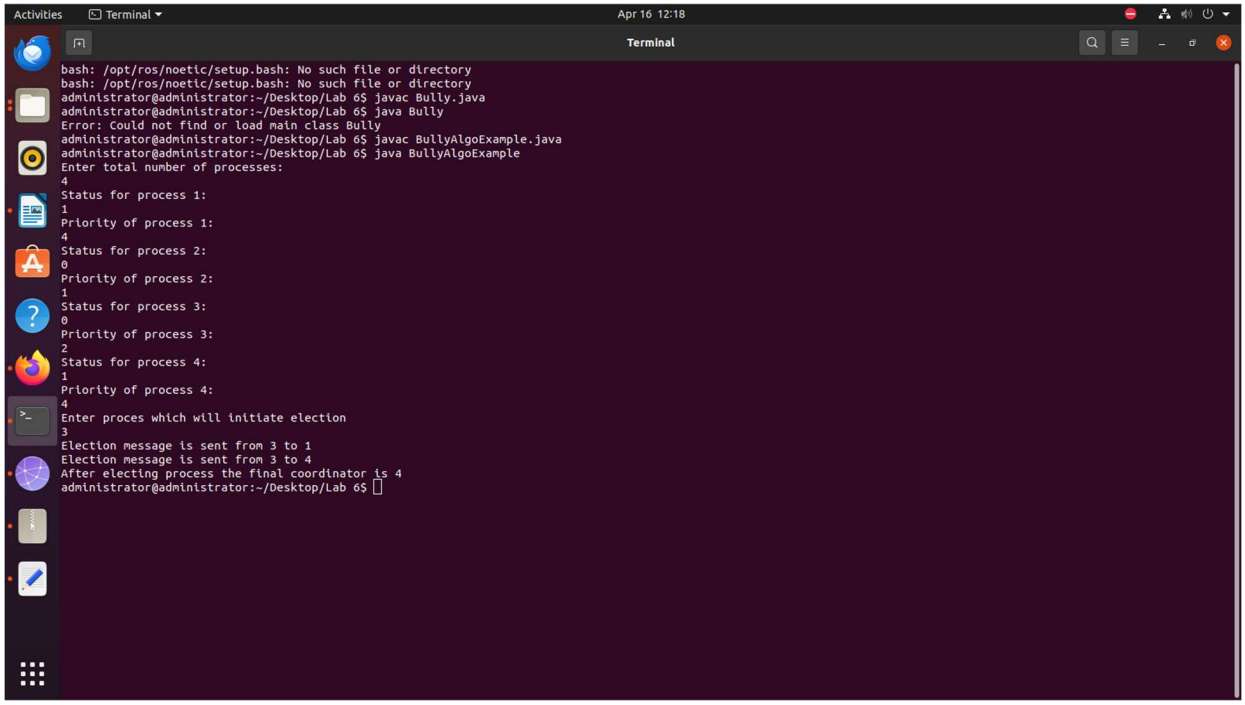
RingAlgorithm.java:

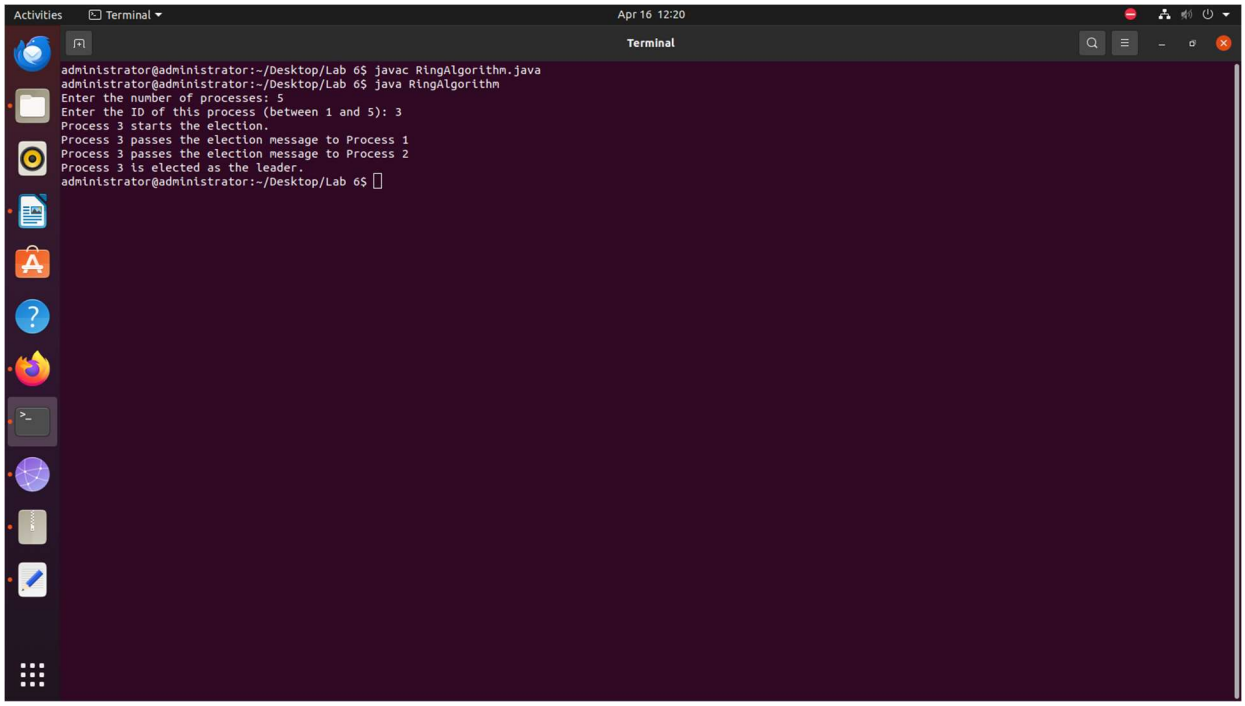
```
import java.util.Scanner;
public class RingAlgorithm {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the number of processes: ");
int numProcesses = scanner.nextInt();
System.out.print("Enter the ID of this process (between 1
and " + numProcesses + "): ");
int thisProcessId = scanner.nextInt();
// Initialize the ring
RingProcess[] ring = new RingProcess[numProcesses];
for (int i = 0; i < numProcesses; i++) {
```

```
ring[i] = new RingProcess(i + 1);
}
// Set the next process in the ring for each process
for (int i = 0; i < numProcesses; i++) {
ring[i].setNextProcess(ring[(i + 1) % numProcesses]);
}
// Start the election
ring[thisProcessId - 1].startElection();
}
}
class RingProcess {
private int processId;
private RingProcess nextProcess;
private boolean isLeader;
public RingProcess(int processId) {
this.processId = processId;
this.isLeader = false;
}
public void setNextProcess(RingProcess nextProcess) {
this.nextProcess = nextProcess;
}
public void startElection() {
System.out.println("Process " + processId + " starts the
election.");
if (isLeader) {
System.out.println("Process " + processId + " is already
the leader.");
return;
}
RingProcess currentProcess = this;
while (true) {
if (currentProcess.nextProcess.processId == processId) {
currentProcess.isLeader = true;

System.out.println("Process " + processId + " is elected
as the leader.");
break;
} else if (currentProcess.nextProcess.processId >
```

```
processId) {
currentProcess = currentProcess.nextProcess;
} else {
System.out.println("Process " + processId + " passes the
election message to Process " +
currentProcess.nextProcess.processId);
currentProcess = currentProcess.nextProcess;
}
}
}
}
```

Output:




Activities
Terminal
Apr 16 12:20
Terminal
administrator@administrator:~/Desktop/Lab 6$ javac RingAlgorithm.java
administrator@administrator:~/Desktop/Lab 6$ java RingAlgorithm
Enter the number of processes: 5
Enter the ID of this process (between 1 and 5): 3
Process 3 starts the election.
Process 3 passes the election message to Process 1
Process 3 passes the election message to Process 2
Process 3 is elected as the leader.
administrator@administrator:~/Desktop/Lab 6$

Assignment 7:

Problem Statement: Create a simple web service and write any distributed application to consume the web service.

Codes:

Output: