# ASSIGNMENT 1

Name-------

Roll no:------

# Class- BE(IT)

AddClient.java

import java.rmi.\*; import java.net.\*; import java.io.\*; import java.util.\*; public class AddClient

{public static void main(String args[]){ String host="localhost";

Scanner sc = new Scanner(System.in);

System.out.println("Enter 1st Parameter"); int a =sc.nextInt();

System.out.println("Enter 2nd Parameter"); int b = sc.nextInt();

try{AddRem remobj =(AddRem)Naming.lookup("rmi://"+host+"/AddRem"); System.out.println(remobj.addNum(a,b)); }

catch(RemoteException re)

{re.printStackTrace();}

catch(NotBoundException nbe){ nbe.printStackTrace();}

catch(MalformedURLException mfe)

{mfe.printStackTrace(); }}}

# AddRem.java

import java.rmi.\*;

public interface AddRem extends Remote

{public int addNum(int a, int b) throws RemoteException;}

# AddRemImpl.java

import java.rmi.\*;

import java.rmi.server.UnicastRemoteObject;

public class AddRemImpl extends UnicastRemoteObject implements AddRem { public AddRemImpl() throws RemoteException {

// Constructor implementation

}@Override

public int addNum(int a, int b) throws RemoteException {

// Method implementation return a + b;}}

AddServer.java import java.rmi.\*; import java.net.\*;

public class AddServer

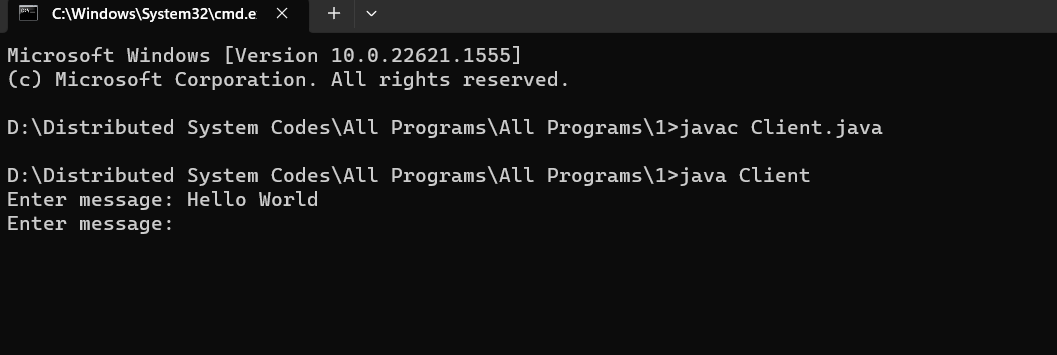
{public static void main(String[] args) {

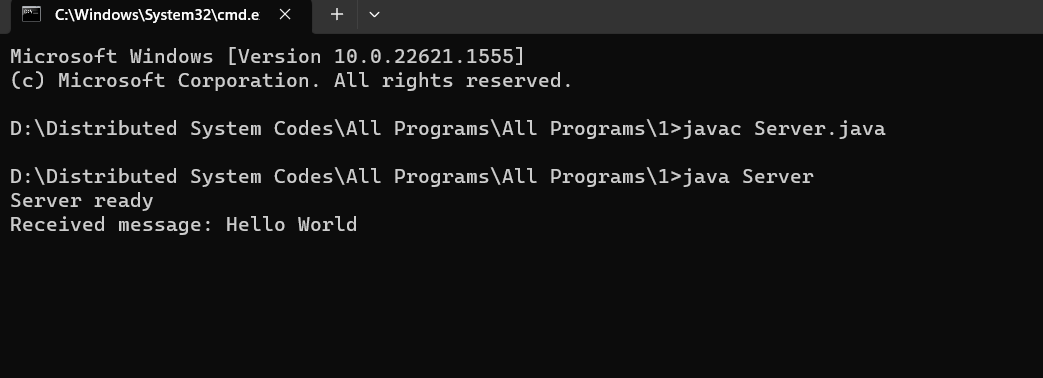
try{AddRemImpl localobj = new AddRemImpl(); Naming.rebind("rmi:///AddRem",localobj);}

catch(RemoteException re)

{re.printStackTrace();} catch(MalformedURLException mfe)

{mfe.printStackTrace(); }





# ASSIGNMENT 2

Name- ------

Roll no:------

# Class- BE(IT)

ReverseModule.idl

module ReverseModule //module ReverseModule is the name of the module

{interface Reverse{

string reverse\_string(in string str); }; };

# 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.CosNaming.NamingContextPackage.\*; import org.omg.CORBA.\*;

import java.io.\*; class ReverseClient

{ public static void main(String args[]){ Reverse ReverseImpl=null;

try{ // initialize the ORB

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

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService"); NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

String name = "Reverse";

//Helper class provides narrow method that cast corba object reference (ref) into the java

// System.out.println("Step2");

// Look ups "Reverse" in the naming context

ReverseImpl = ReverseHelper.narrow(ncRef.resolve\_str(name)); System.out.println("Enter String=");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); String str= br.readLine();

String tempStr= ReverseImpl.reverse\_string(str); System.out.println(tempStr);

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

# ReverseServer.java

import ReverseModule.Reverse; import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*; import org.omg.CORBA.\*;

import org.omg.PortableServer.\*; class ReverseServer

{public static void main(String[] args)

{try{

// initialize the ORB

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

// initialize the portable object adaptor (BOA/POA) connects client request using object reference

//uses orb method as resolve\_initial\_references

POA rootPOA = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA")); rootPOA.the\_POAManager().activate();

// creating an object of ReverseImpl class ReverseImpl rvr = new ReverseImpl();

//server consist of 2 classes ,servent and server. The servent is the subclass of ReversePOA which is generated by the idlj compiler

// The servent ReverseImpl is the implementation of the ReverseModule idl interface

// get the object reference from the servant class

//use root POA class and its method servant\_to\_reference org.omg.CORBA.Object ref = rootPOA.servant\_to\_reference(rvr);

// System.out.println("Step1");

Reverse h\_ref = ReverseModule.ReverseHelper.narrow(ref);// Helper class provides narrow method that cast corba object reference (ref) into the java interface

// System.out.println("Step2");

// orb layer uses resolve\_initial\_references method to take initial reference as NameService org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

//Register new object in the naming context under the Reverse

// System.out.println("Step3");

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

//System.out.println("Step4"); String name = "Reverse";

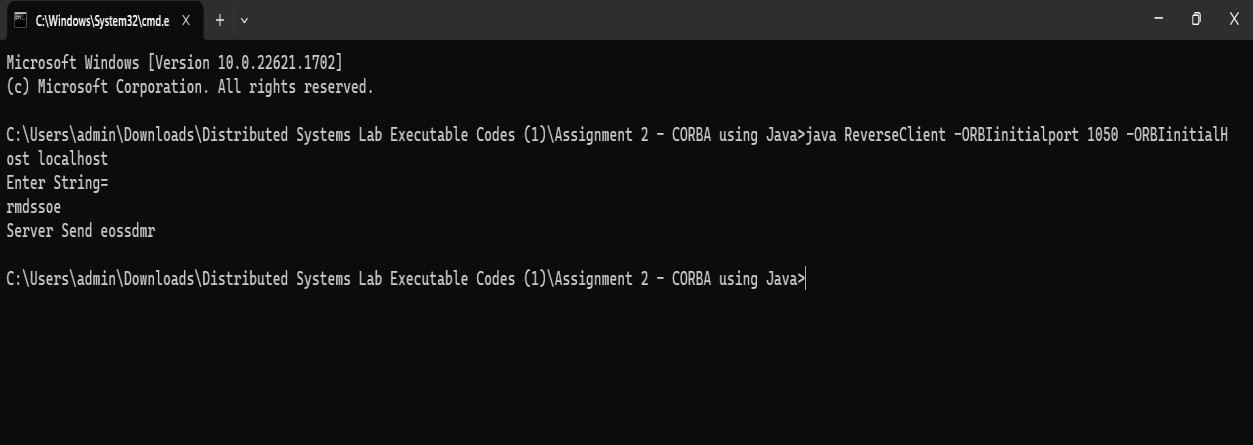
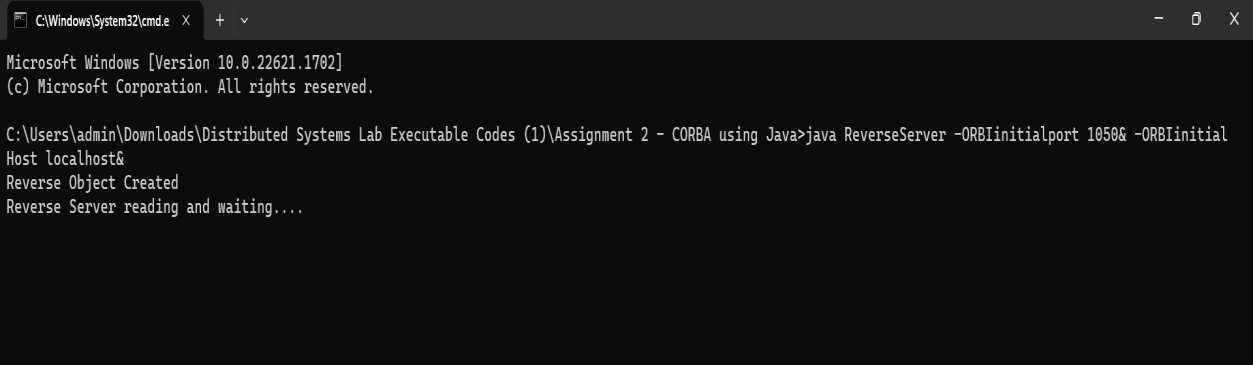
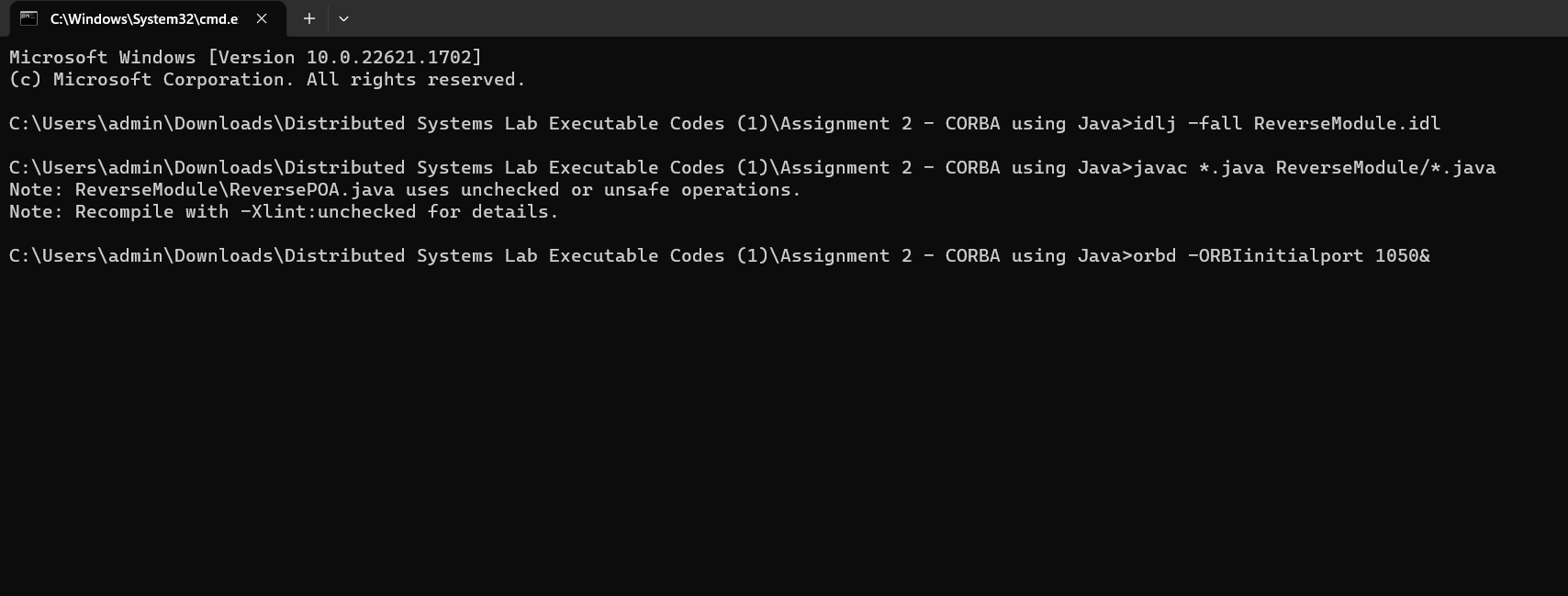
NameComponent path[] = ncRef.to\_name(name); ncRef.rebind(path,h\_ref);

//Server run and waits for invocations of the new object from the client System.out.println("Reverse Server reading and waiting ");

orb.run();} catch(Exception e){

e.printStackTrace(); }}}

# OUTPUT-



ASSIGNMENT 3

# Name- ------

# Roll no:------

Class- BE(IT)

# arr\_sum\_mpi.c

#include<stdio.h> #include<mpi.h> #define arr\_size 15

int main(int argc, char \*argv[]){ int rank, size; MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

//Code that will execute inside process 0 or rank 0 if(rank == 0){

int arr[]= {12,4,6,3,21,15,3,5,7,8,9,1,5,3,5};

int global\_sum = 0, local\_sum = 0, recv\_local\_sum;

//If the array size is perfectly divisible by number of process. if(arr\_size%size == 0){

int array\_element\_per\_process = arr\_size/size; int sub\_arr[array\_element\_per\_process]; for(int i=1; i<size; i++){

//Copying the sub array

for(int j=0; j<array\_element\_per\_process;j++){ sub\_arr[j] = arr[i\*array\_element\_per\_process+j];}

//Sending array chunk of equal size to all the process.

MPI\_Send(sub\_arr, array\_element\_per\_process, MPI\_INT, i, 1, MPI\_COMM\_WORLD);

MPI\_Send(&array\_element\_per\_process, 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD);}

//Calculating the local sum of rank 0 itself for(int j=0; j<array\_element\_per\_process; j++){

local\_sum += arr[j];}

printf("Rank %d: local sum: %d\n", rank, local\_sum); global\_sum += local\_sum;

//When the array size is not perfectly divisible by number of process.

}else{

int array\_element\_per\_process = arr\_size/size + 1; int sub\_arr[array\_element\_per\_process];

for(int i=1; i<size; i++){ if(i == size - 1){

//last sub array will have the size less than other process array size

int total\_array\_size\_of\_last\_process = arr\_size - array\_element\_per\_process \* i; for(int j=0; j< total\_array\_size\_of\_last\_process; j++){

sub\_arr[j] = arr[i\*array\_element\_per\_process+j];}

MPI\_Send(&sub\_arr, total\_array\_size\_of\_last\_process, MPI\_INT, i, 1, MPI\_COMM\_WORLD);

MPI\_Send(&total\_array\_size\_of\_last\_process, 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD);

}else{

//Copying the sub array

for(int j=0; j<array\_element\_per\_process;j++){ sub\_arr[j] = arr[i\*array\_element\_per\_process+j];}

MPI\_Send(&sub\_arr, array\_element\_per\_process, MPI\_INT, i, 1, MPI\_COMM\_WORLD); MPI\_Send(&array\_element\_per\_process, 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD}

Calculating the local sum of rank 0 itself

for(int j=0; j<array\_element\_per\_process; j++){ local\_sum += arr[j];}

printf("Rank %d: local sum: %d\n", rank, local\_sum); global\_sum += local\_sum;

}//calculating the global sum of the array

//Receving the local sum from the other process and updating the global sum for(int i=1; i<size; i++){

MPI\_Recv(&recv\_local\_sum, 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE); global\_sum += recv\_local\_sum;

}//Printing the output

printf("The sum of the array is %d\n", global\_sum);

//Code that will get executed inside other than process 0 or rank 0.

}else{//The other process will receive the chunck of array int array\_element\_per\_process = arr\_size/size + 1;

int recv\_sub\_arr[array\_element\_per\_process];

int recv\_array\_element\_per\_process, local\_sum = 0;

MPI\_Recv(recv\_sub\_arr, recv\_array\_element\_per\_process, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Recv(&recv\_array\_element\_per\_process, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

//Calculating local sum for the sub array

for(int j=0; j<recv\_array\_element\_per\_process; j++){ local\_sum += recv\_sub\_arr[j];}

//Printing the local sum

printf("Rank %d: local sum: %d\n", rank, local\_sum);

//Sending back the local sum to the rank 0 or process 0.

MPI\_Send(&local\_sum, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD);} MPI\_Finalize();

return 0;}

# arr\_sum.c

#include <mpi.h> #include <stdio.h> #include <stdlib.h> #define ARRAY\_SIZE 16

int main(int argc, char\*\* argv) { int rank, size;

int sum = 0;

int array[ARRAY\_SIZE];

// Initialize MPI MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

// Populate the array on the root process if (rank == 0) {

for (int i = 0; i < ARRAY\_SIZE; i++) { array[i] = i + 1;}}

// Scatter the array to all processes int subarray\_size = ARRAY\_SIZE / size; int subarray[subarray\_size];

MPI\_Scatter(array, subarray\_size, MPI\_INT, subarray, subarray\_size, MPI\_INT, 0, MPI\_COMM\_WORLD);

// Sum the local elements int local\_sum = 0;

for (int i = 0; i < subarray\_size; i++) { local\_sum += subarray[i];}

// Display the local sum of each process

printf("Process %d local sum is %d\n", rank, local\_sum);

// Reduce the local sums to get the final sum on the root process MPI\_Reduce(&local\_sum, &sum, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

// Print the result on the root process if (rank == 0) {

printf("The sum of the elements is %d\n", sum);}

// Finalize MPI MPI\_Finalize(); return 0;}

# Output-



ASSIGNMENT 4

# Name- ------ Roll no:------

Class- BE(IT)

# 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()

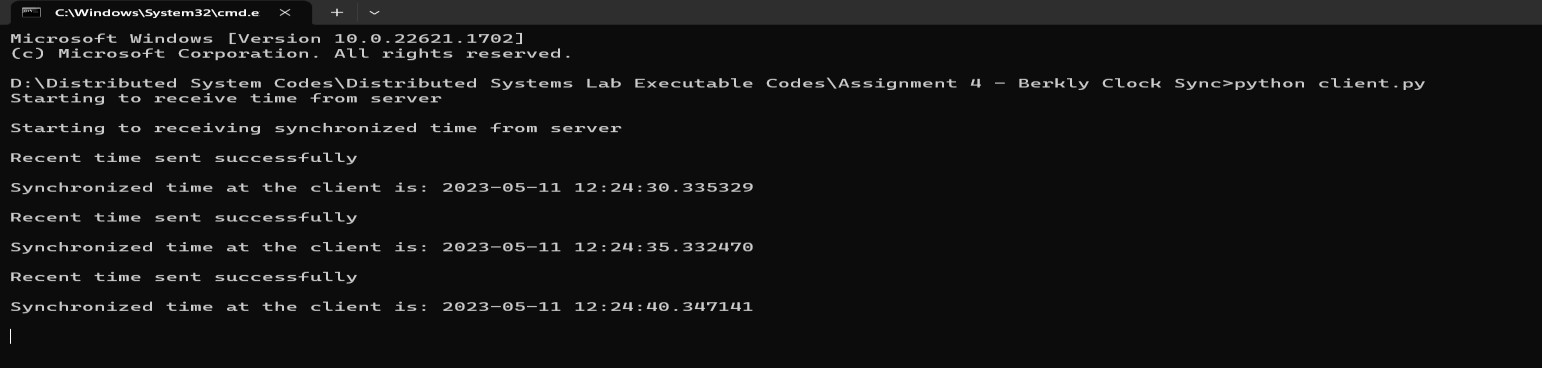
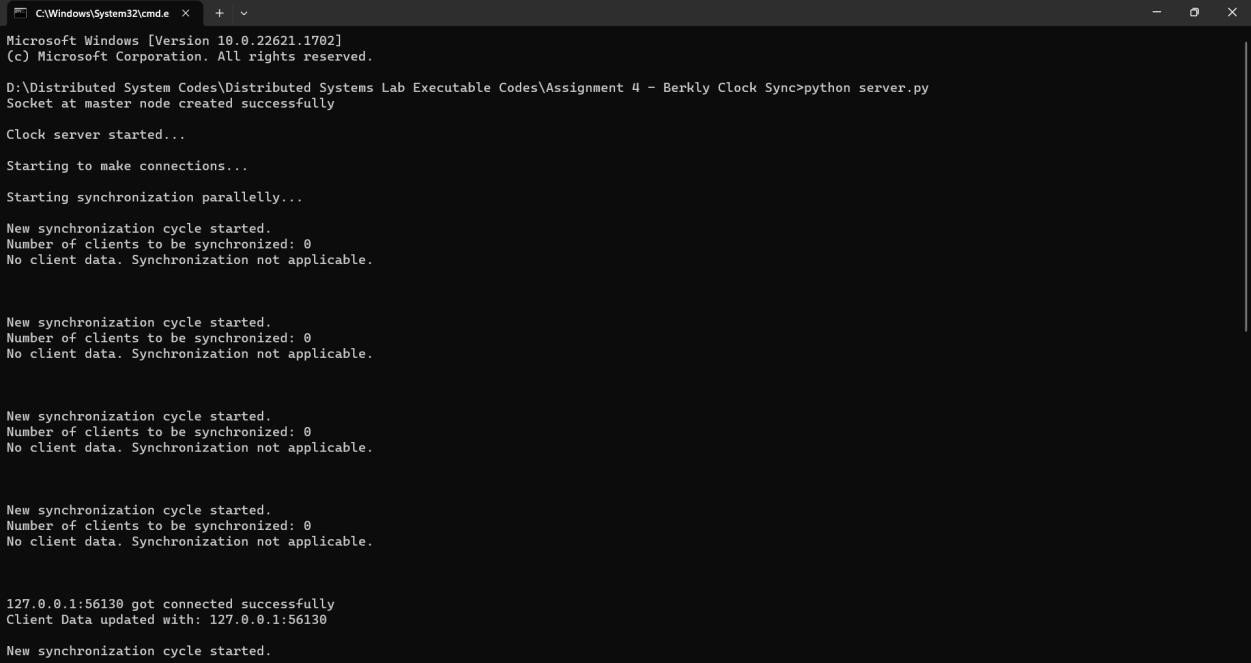
# start receiving synchronized from server print("Starting to receiving " +

"synchronized time from server\n") receive\_time\_thread = threading.Thread(

target=startReceivingTime, args=(slave\_client, ))

receive\_time\_thread.start() if name == ' main ': initiateSlaveClient(port=8080)

# Output-



ASSIGNMENT 5

# Name- ------

# Roll no:------

Class- BE(IT)

# Ring-token.c

#include<stdio.h> #include<conio.h> #include<dos.h> #include<time.h> void main(){

int cs=0,pro=0; double run=5; char key='a'; time\_t t1,t2;

printf("Press a key(except q) to enter a process into critical section."); printf(" \nPress q at any time to exit.");

t1 = time(NULL) - 5; while(key!='q')

{while(!kbhit()) 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)

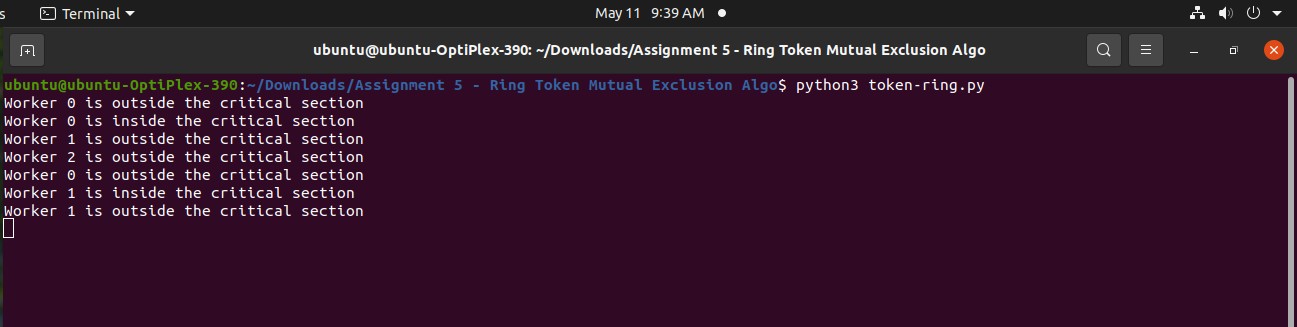
printf("Error: Another process is currently executing critical section Please wait till its n"); else printf("Process %d ",pro);

printf(" entered critical section\n"); cs=1;

pro++;

t1 = time(NULL); }}}}

# Output-



ASSIGNMENT 6

# Name- ------

# Roll no:------

Class- BE(IT)

# Bully\_ring.cpp

// first we include the necessary header files #include <iostream>

#include <cstdlib>

// we define MAX as the maximum number of processes our program can simulate

// we declare array pStatus[MAX] to store the process status; 0 for dead and 1 for alive

// we declare n as the number of processes

// we declare coordinator to store the winner of election int pStatus[MAX], n, coordinator;

using namespace std; void bully();

void ring();

// void ring\_(); // this is also another approach ring implementation, and works well. void display();

int main()

{int i, j, fchoice;

cout << "Enter number of processes: "; cin >> n;

for (i = 1; i <= n; i++)

{cout << "Enter Process " << i << " is alive or not(0/1): "; cin >> pStatus[i];

if (pStatus[i]) coordinator = i;}

display();

do{cout << "- ";

cout << "\n1.BULLY ALGORITHM\n2.RING\n3.DISPLAY\n4.EXIT\n";

cout << "--- \n\n"; cout << "Enter your choice: ";

cin >> fchoice; switch (fchoice){ case 1:

bully(); break;

case 2:

ring();

// ring\_() break;

case 3:

display(); break;

case 4:

exit(1); break;}

} while (fchoice != 3); return 0;}

void display()

{ int i;

// we display the processes, their status and the coordinator cout << "--- \n";

cout << "Processes: ";

for (i = 1; i <= n; i++) // PID from 1 to n cout << i << "\t";

cout << endl

<< "Alive: "; for (i = 1; i <= n; i++)

cout << pStatus[i] << "\t";

cout << "\n- \n"; cout << "COORDINATOR IS " << coordinator << endl;}

void bully()

{int schoice, crash, activate, i, gid, flag, subcoordinator; do

{cout << "-- -";

cout << "\n1.CRASH\n2.ACTIVATE\n3.DISPLAY\n4.EXIT\n";

cout << "--- \n"; cout << "Enter your choice: ";

cin >> schoice; switch (schoice){ case 1:

// we manually crash the process to see if our implementation

// can elect another coordinator cout << "Enter process to crash: "; cin >> crash;

// if the process is alive then set its status to dead if (pStatus[crash])

pStatus[crash] = 0; else

cout << "Process " << crash << " is already dead!" << endl; do{cout << "Enter election generator id: ";

cin >> gid;

if (gid == coordinator || pStatus[gid] == 0)

cout << "Please, enter a valid generator id.." << endl;

} while (gid == coordinator || pStatus[gid] == 0); flag = 0;

// if the coordinator has crashed then we need to find another coordinator if (crash == coordinator)

{// the election generator process will send the message to all higher process for (i = gid + 1; i <= n; i++)

{cout << "Message is sent from " << gid << " to " << i << endl;

// if the higher process is alive then it will respond if (pStatus[i])

{subcoordinator = i;

cout << "Response is sent from " << i << " to " << gid << endl; flag = 1;}}

// the highest responding process is selected as the coordinator if (flag == 1)

coordinator = subcoordinator;

// else if no higher process are alive then the election generator process

// is selected as coordinator else

coordinator = gid;} display();

break; case 2:

// enter process to revive

cout << "Enter Process ID to be activated: "; cin >> activate;

// if the entered process was dead then it is revived if (!pStatus[activate])

{pStatus[activate] = 1;} else

{cout << "Process " << activate << " is already alive!" << endl; break;}

if (activate == n)

{coordinator = n; break; }

flag = 0;

// else, the activated process sends message to all higher process for (i = activate + 1; i <= n; i++)

{cout << "Message is sent from " << activate << " to " << i << endl;

// if higher process is active then it responds if (pStatus[i])

{subcoordinator = i;

cout << "Response is sent from " << i << " to " << activate << endl; flag = 1;}}

// the highest responding process is made the coordinator if (flag == 1)

coordinator = subcoordinator;

// if no higher process respond then the activated process is coordinator else

coordinator = activate; display();

break; case 3:

display(); break;

case 4:

break;}

} while (schoice != 4);}

// ring algorithm implementation void ring()

{int tchoice, crash, activate, gid, subcoordinator, i; do

{cout << "-- -";

cout << "\n1.CRASH\n2.ACTIVATE\n3.DISPLAY\n4.EXIT\n";

cout << "--- \n\n"; cout << "Enter your choice: ";

cin >> tchoice; switch (tchoice)

{case 1:

cout << "\nEnter Process ID to crash : "; cin >> crash;

if (pStatus[crash]) pStatus[crash] = 0;

else

cout << "Process " << crash << " is already dead!" << endl; do

{cout << "Enter election generator id: ";

cin >> gid;

if (gid == coordinator)

cout << "Please, enter a valid generator id.." << endl;

} while (gid == coordinator); if (crash == coordinator)

{subcoordinator = 1;

for (i = 0; i < (n + 1); i++)

{int pid = (i + gid) % (n + 1);

if (pid != 0) // since process id starts from 1 (to n)

{if (pStatus[pid] && subcoordinator < pid)

{subcoordinator = pid; }

cout << "Election message sent from " << pid << ": #Msg" << subcoordinator << endl; }}} coordinator = subcoordinator;}

display(); break;

case 2:

cout << "Enter Process ID to be activated: "; cin >> activate;

if (!pStatus[activate]) pStatus[activate] = 1;

else{cout << "Process " << activate << " is already alive!" << endl; break;}

subcoordinator = activate; for (i = 0; i < n + 1; i++)

{int pid = (i + activate) % (n + 1); if (pid != 0)

{if (pStatus[pid] && subcoordinator < pid)

{subcoordinator = pid; }

cout << "Election message passed from " << pid << ": #Msg" << subcoordinator << endl;}} coordinator = subcoordinator;

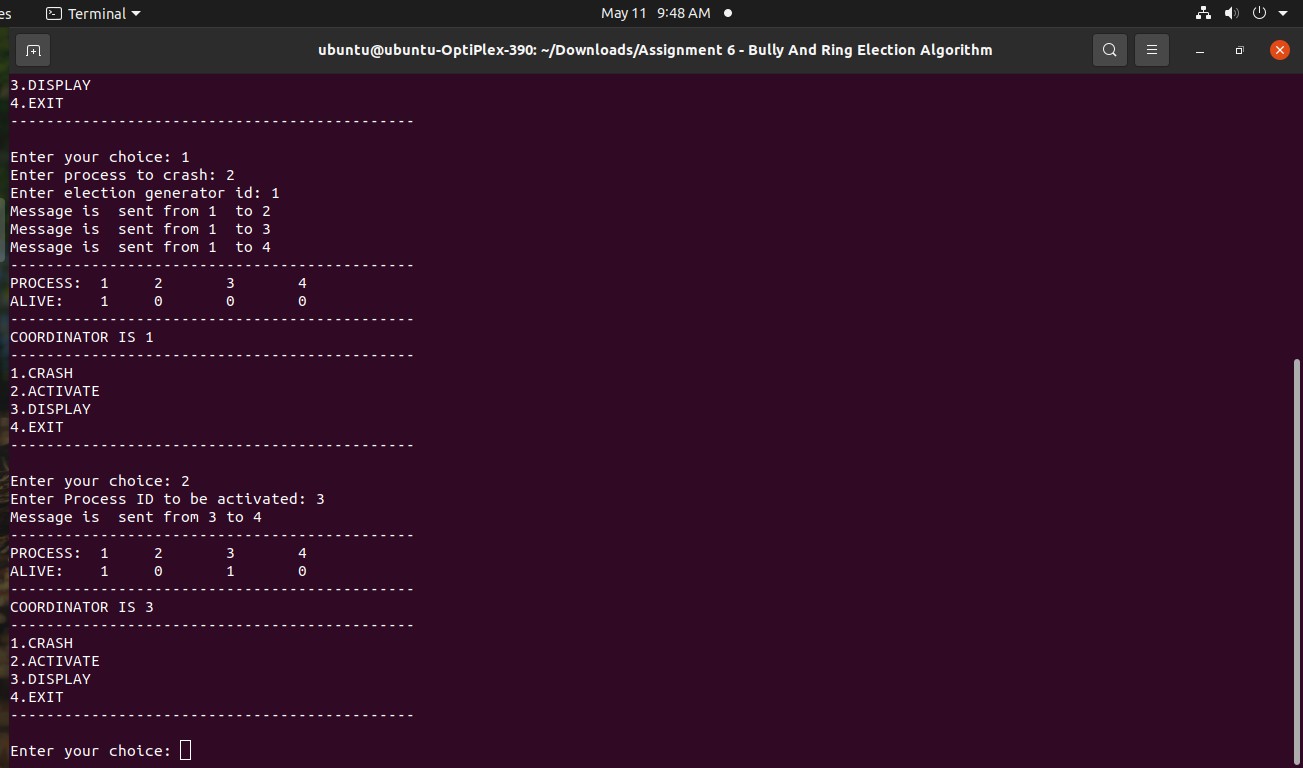
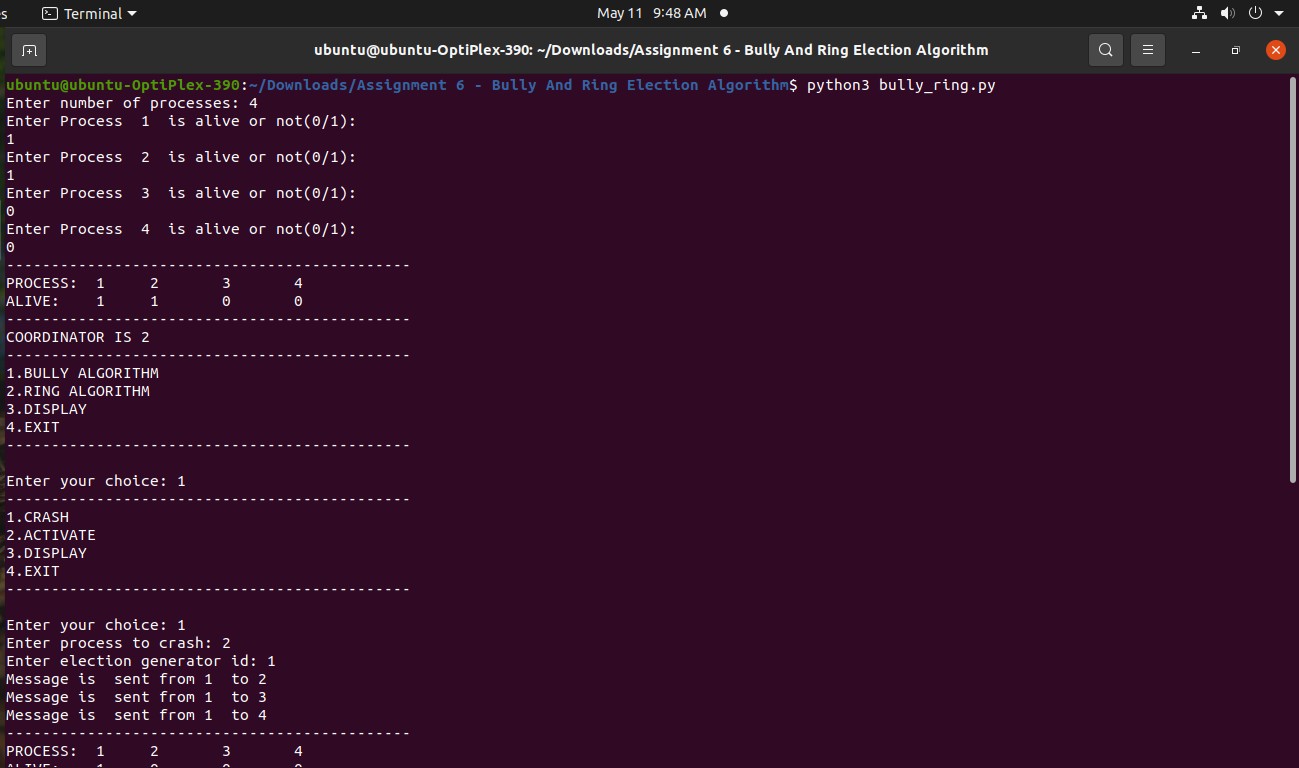
case 3:

display(); break;

default:

break; } while (tchoice != 4);

# Output-



ASSIGNMENT 7

# Name- ------

# Roll no:------

Class- BE(IT)

**SOAP:**

**CalcServlet.java file:**

protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8"); try (PrintWriter out = response.getWriter()) {

/\* TODO output your page here. You may use following sample code. \*/ out.println("<!DOCTYPE html>");

out.println("<html>"); out.println("<head>");

out.println("<title>Servlet CalcServlet</title>"); out.println("</head>");

out.println("<body>");

out.println("<h1>Servlet CalcServlet at " + request.getContextPath() + "</h1>"); out.println("</body>");

out.println("</html>");

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

* Handles the HTTP <code>GET</code> method.

\*

* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//processRequest(request, response); PrintWriter out = response.getWriter(); int x,y;

String str = "";

x = Integer.parseInt(request.getParameter("txtfno")); y = Integer.parseInt(request.getParameter("txtsno")); str = request.getParameter("operation"); if(str.equals("add"))

{

out.println("<h1>Result of Addition is:" + (x+y) +"</h1>");

}

else if(str.equals("sub"))

{

out.println("<h1>Result of Subtraction is:" + (x-y) +"</h1>");

}

else if(str.equals("mult"))

{

out.println("<h1>Result of Multiplication is:" + (x\*y) +"</h1>");

}

else if(str.equals("add"))

{

out.println("<h1>Result of Division is:" + (x/y) +"</h1>");

}

else

{

out.println("<h1>Result of Modulus is:" + (x%y) +"</h1>");

}

}

/\*\*

* Handles the HTTP <code>POST</code> method.

\*

* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

* Returns a short description of the servlet.

\*

* @return a String containing servlet description

\*/

@Override

public String getServletInfo() { return "Short description";

}// </editor-fold>

}

**Index.html file-**

<html>

<head>

<title>TODO supply a title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<div>TODO write content</div>

<form method="get" action="CalcServlet">

<h1> Calculator </h1> First Number:

<input type="text" name="txtfno"/><br/> Second Number:

<input type="text" name="txtsno"/><br/> Select the operation:<br/>

<input type="radio" name="operation" value="add">Addition

<input type="radio" name="operation" value="sub">Subtraction

<input type="radio" name="operation" value="mult">Multiplication

<input type="radio" name="operation" value="divi">Division

<input type="radio" name="operation" value="modu">Modulus <br/>

<input type="submit" value="Calculate"/>

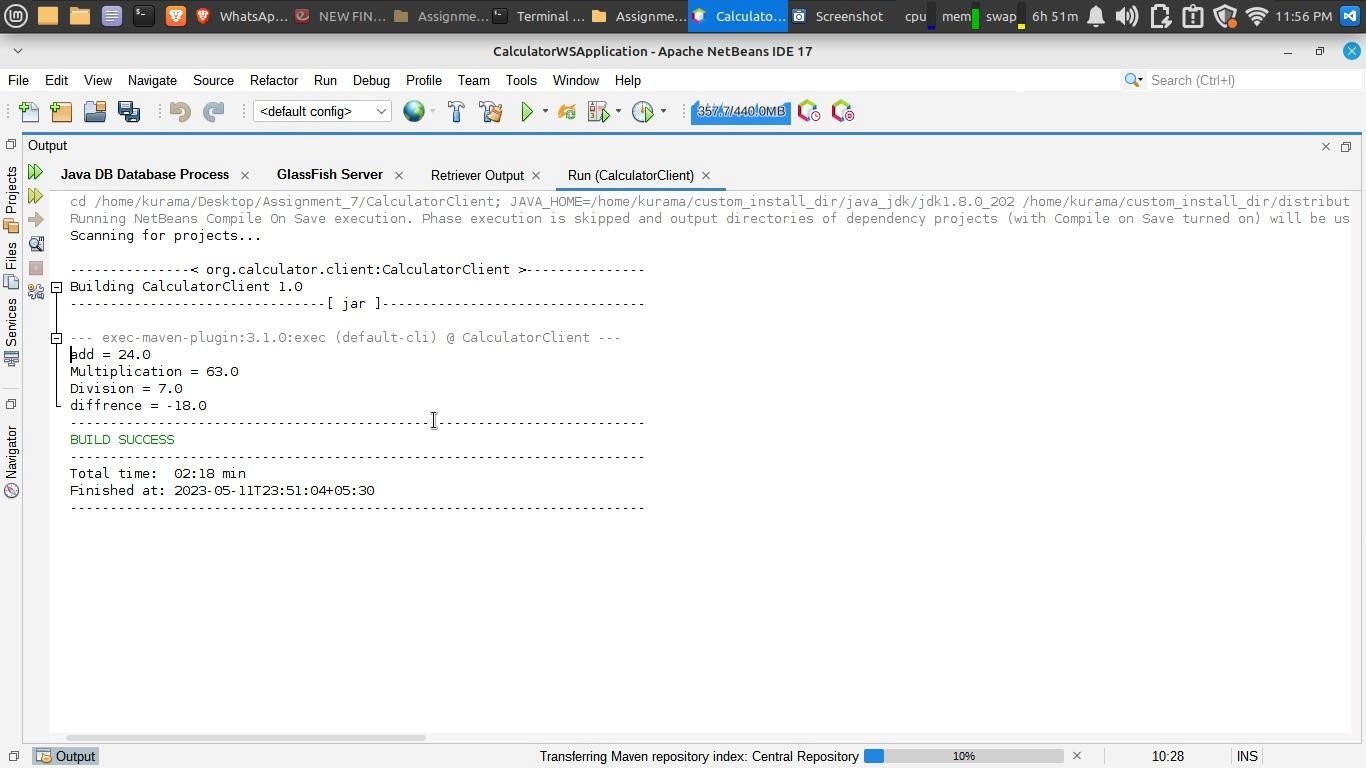
<input type="reset" value="Reset"/>

</form>

</body>

</html>

# Output:



Output:

