ASSIGNMENT 1

Name- Om Patil
Roll No- 49
Class- BE(IT)

Code -

Server.java

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

public class Server implements Service {
    private ArrayList<String> messages = new ArrayList<>();

    public Server() throws RemoteException {
        UnicastRemoteObject.exportObject(this, 0);
    }

    public void receiveMessage(String message) throws RemoteException {
        System.out.println("Received message: " + message);
        messages.add(message);
    }

    public static void main(String[] args) {
        try {
            Server server = new Server();
            Registry registry = LocateRegistry.createRegistry(1099);
            Naming.rebind("rmi://localhost/Service", server);
```

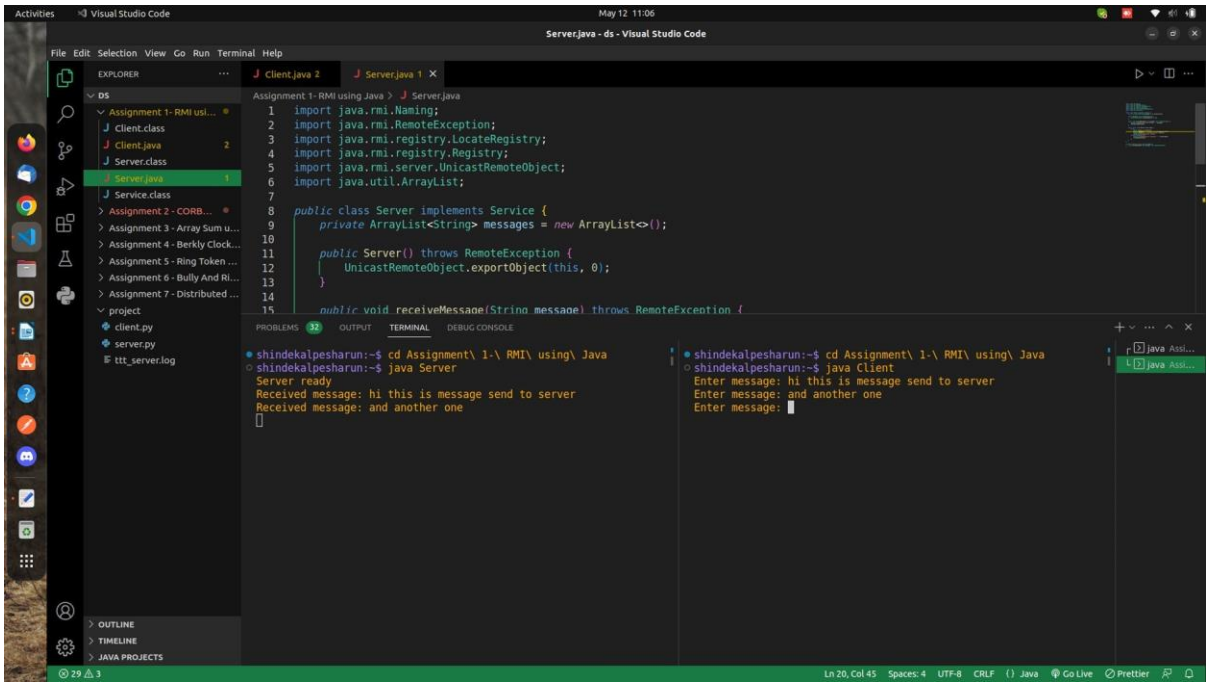
```
            System.out.println("Server ready");
        } catch (Exception e) {
            System.out.println("Server exception: " + e.toString());
            e.printStackTrace();s
        }
    }
}

interface Service extends java.rmi.Remote {
    void receiveMessage(String message) throws RemoteException;
}
```

Client.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;

public class Client implements Runnable {
    private Service service;

    public Client(Service service) {
        this.service = service;
    }

    public void run() {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.print("Enter message: ");
            String message = scanner.nextLine();
            try {
                service.receiveMessage(message);
```

```
            } catch (RemoteException e) {
                System.out.println("Client exception: " + e.toString());
                e.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        try {
            Registry registry = LocateRegistry.getRegistry(1099);
            Service service = (Service) Naming.lookup("rmi://localhost/Service");
            Client client = new Client(service);
            Thread thread = new Thread(client);
            thread.start();
        } catch (Exception e) {
            System.out.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Output -


Server.java - ds - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
Client.java 2
Server.java 1
Assignment 1- RMI using Java > Server.java
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
public class Server implements Service {
private ArrayList<String> messages = new ArrayList<>();
public Server() throws RemoteException {
UnicastRemoteObject.exportObject(this, 0);
}
PROBLEMS 32 OUTPUT TERMINAL DEBUG CONSOLE
shindekalpesharun:~$ cd Assignment\ 1-\ RMI\ using\ Java
shindekalpesharun:~$ java Server
Server ready
Received message: hi this is message send to server
Received message: and another one
shindekalpesharun:~$ cd Assignment\ 1-\ RMI\ using\ Java
shindekalpesharun:~$ java Client
Enter message: hi this is message send to server
Enter message: and another one
Enter message:

ASSIGNMENT 2

Name- Om Patil
Roll No- 49
Class- BE(IT)

Code -

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
interface
            // 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();
          }
    }
}
```

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));
    }
}
```

ReverseModule.idl

```
module ReverseModule //module ReverseModule is the name of the module
{
    interface Reverse{
        string reverse_string(in string str);
    };
};
```

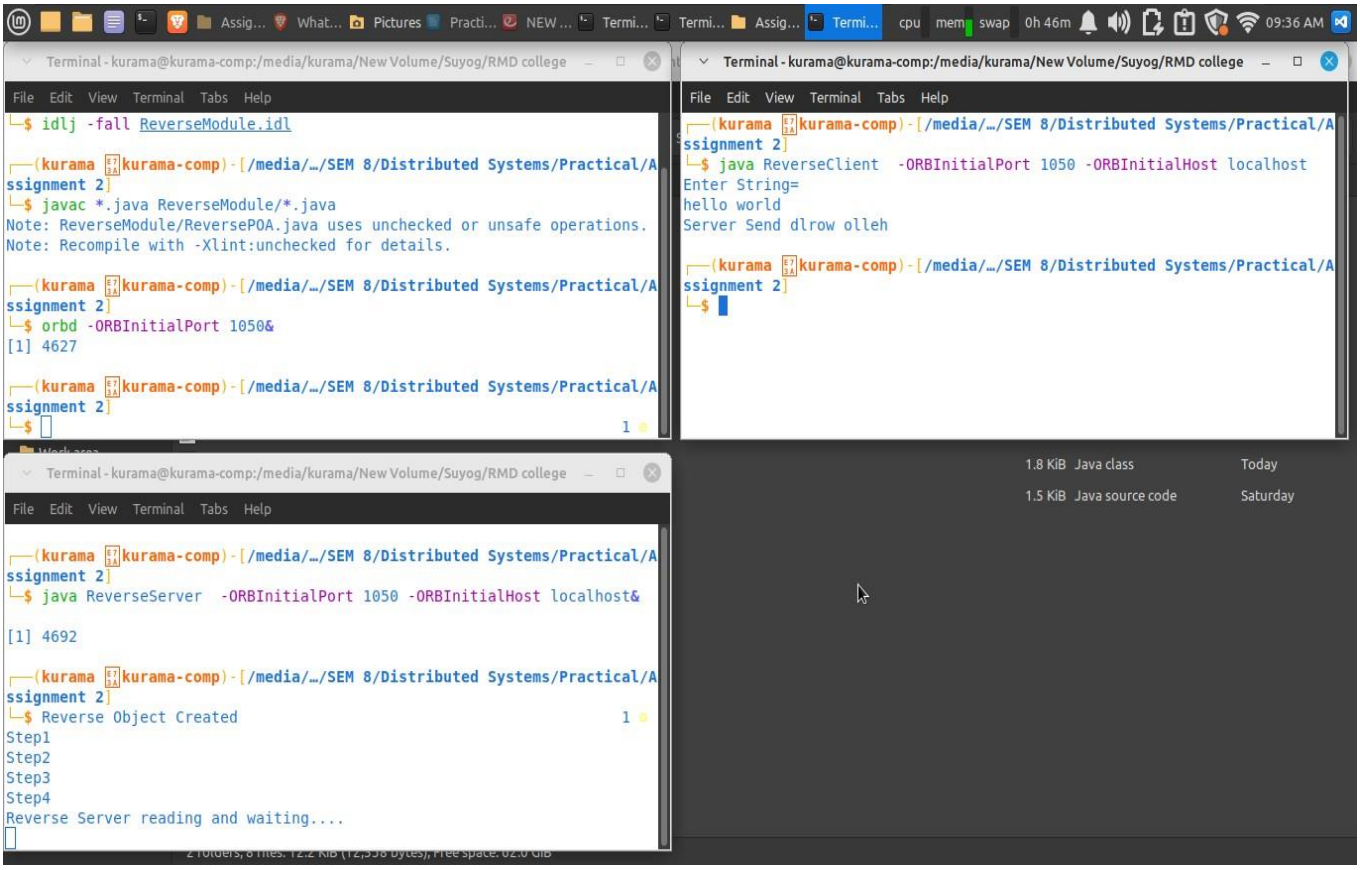
ReverseServer.java

```
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- Om Patil
Roll No- 49
Class- BE(IT)

Code -

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;
}
```

arr_sum_mpi.c

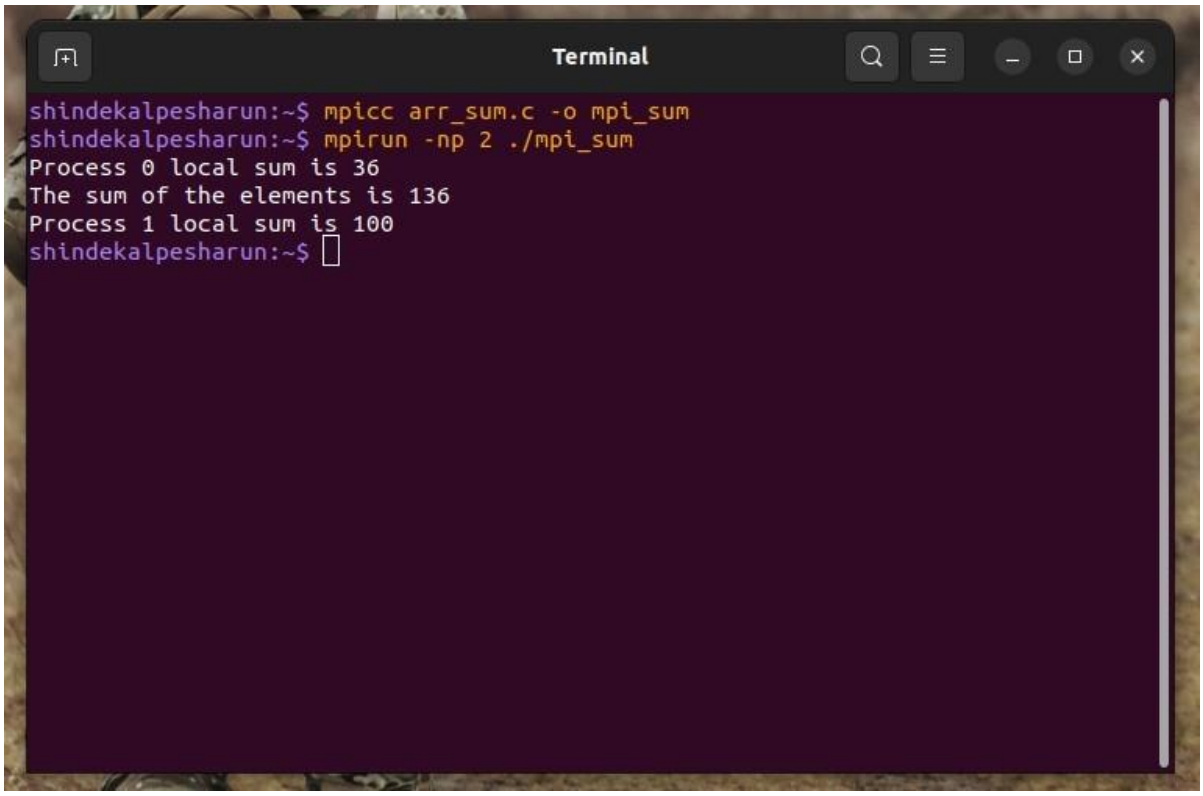
```
#include<stdio.h>
#include<mpi.h>
#define arr_size 15
int main(int argc, char *argv[]){
        int rank, size;
        MPI_Init(&argc, &argv);
```

```
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;
}
```

Output -


Terminal
shindekalpesharun:~$ mpicc arr_sum.c -o mpi_sum
shindekalpesharun:~$ mpirun -np 2 ./mpi_sum
Process 0 local sum is 36
The sum of the elements is 136
Process 1 local sum is 100
shindekalpesharun:~$

ASSIGNMENT 4

Name- Om Patil
Roll No- 49
Class- BE(IT)

Code -

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)
```

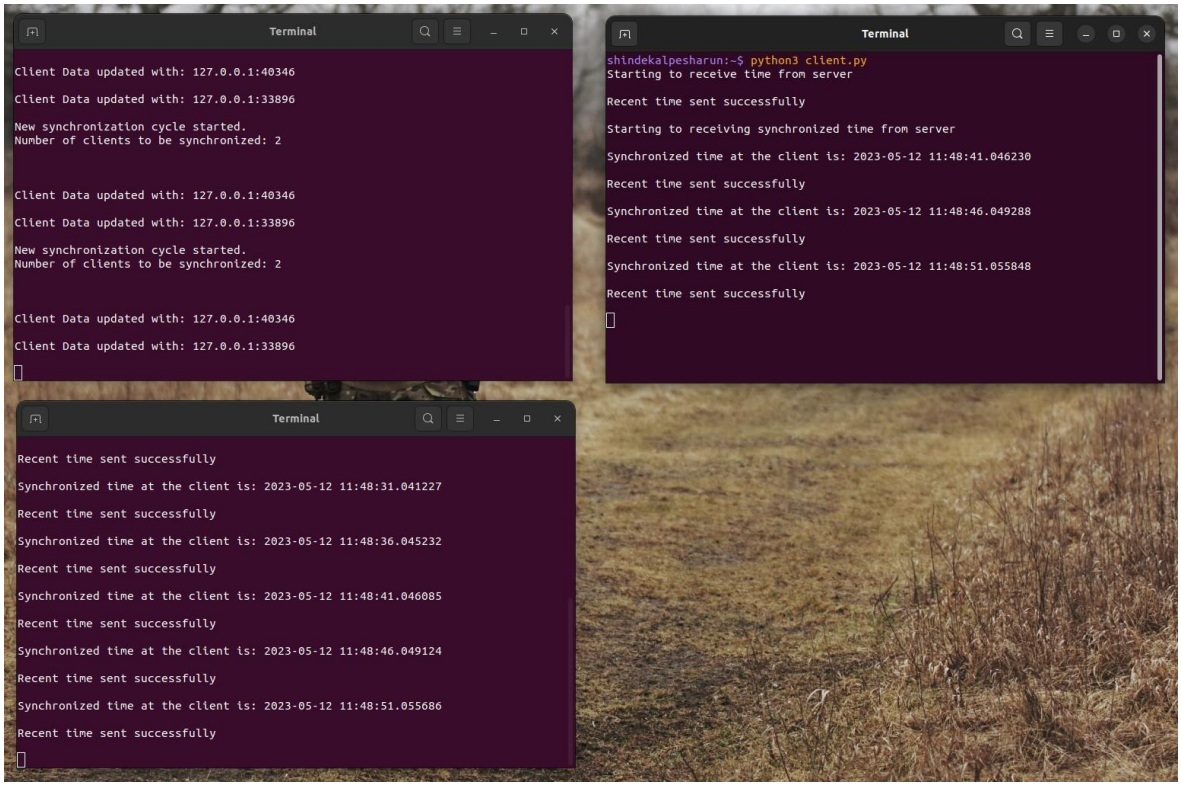
client.py

```
# 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

Name- Om Patil
Roll No- 49
Class- BE(IT)

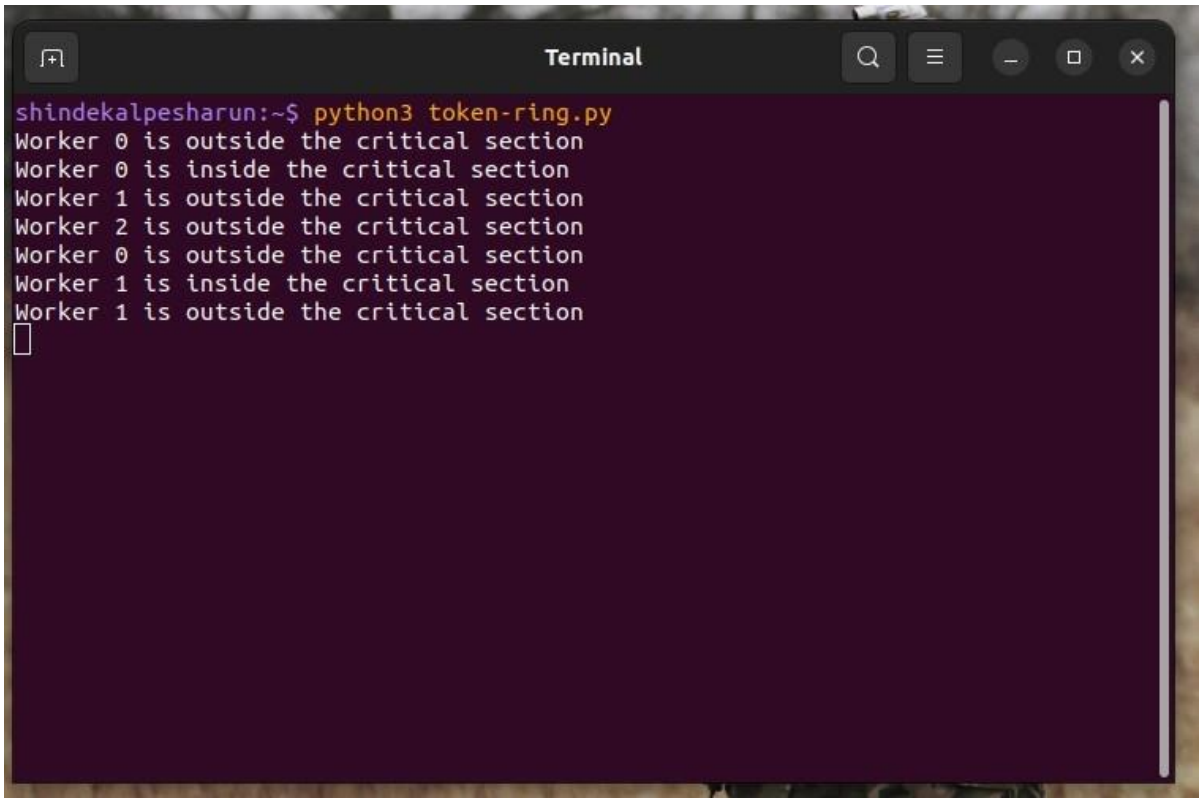
Code -

token-ring.py

```
import threading
import time

class TokenRingMutex:
    def__init_(self, n):
        self.tokens = [threading.Event() for _ in range(n)]
        self.tokens[0].set()
        self.n = n
        self.queue = []

    def request_critical_section(self):
        self.queue.append(threading.current_thread().ident)
        while True:
            token_idx = self.queue.index(threading.current_thread().ident)
            self.tokens[token_idx % self.n].wait()
            if token_idx == 0:
                return

    def release_critical_section(self):
        token_idx = self.queue.index(threading.current_thread().ident)
        self.tokens[(token_idx + 1) % self.n].set()
        self.queue.remove(threading.current_thread().ident)
```

```
def worker(mutex, id):
    while True:
        print(f"Worker {id} is outside the critical section")
        mutex.request_critical_section()
        print(f"Worker {id} is inside the critical section")
        time.sleep(1)
        mutex.release_critical_section()

if__name__== "_main_":
    mutex = TokenRingMutex(3)
    workers = []
    for i in range(3):
        worker_thread = threading.Thread(target=worker, args=(mutex, i))
        workers.append(worker_thread)
        worker_thread.start()

    for worker_thread in workers:
        worker_thread.join()
```

Output -


Terminal
shindekalpesharun:~$ python3 token-ring.py
Worker 0 is outside the critical section
Worker 0 is inside the critical section
Worker 1 is outside the critical section
Worker 2 is outside the critical section
Worker 0 is outside the critical section
Worker 1 is inside the critical section
Worker 1 is outside the critical section

ASSIGNMENT 6

Name- Om Patil
Roll No- 49
Class- BE(IT)

Code - bully_ring.py

```
# we define MAX as the maximum number of processes our program can simulate
# we declare pStatus 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 elections

MAX = 20
pStatus = [0 for _ in range(MAX)]
n = 0
coordinator = 0

# def take_input():
#     global coordinator,n
#     n = int(input("Enter number of processes: "))
#     for i in range(1, n+1):
#         print("Enter Process ",i, " is alive or not(0/1): ")
#         x = int(input())
#         pStatus[i] = x
#         if pStatus[i]:
#             coordinator = i

def bully():
    " bully election implementation"
    global coordinator
    condition = True
    while condition:
```

```
print('------------------------------------------ ')
print("1.CRASH\n2.ACTIVATE\n3.DISPLAY\n4.EXIT")
print('------------------------------------------ \n')
print("Enter your choice: ", end=")
schoice = int(input())

if schoice == 1:
    # we manually crash the process to see if our implementation
    # can elect another leader
    print("Enter process to crash: ", end=")
    crash = int(input())
    # if the process is alive then set its status to dead
    if (pStatus[crash] != 0):
        pStatus[crash] = 0
    else:
        print('Process', crash, ' is already dead!\n')
        break
    condition = True
    while condition:
        # enter another process to initiate the election
        print("Enter election generator id: ", end=")
        gid = int(input())
        if (gid == coordinator or pStatus[gid] == 0):
            print("Enter a valid generator id!")
        condition = (gid == coordinator or pStatus[gid] == 0)
    flag = 0
    # if the coordinator has crashed then we need to find another leader
    if (crash == coordinator):
        # the election generator process will send the message to all higher process
        i = gid + 1
        while i <= n:
            print("Message is sent from", gid, " to", i, end='\n')
            # if the higher process is alive then it will respond
```

```
            if (pStatus[i] != 0):
                subcoordinator = i
                print("Response is sent from", i, " to", gid, end='\n')
                flag = 1
            i += 1
        # the highest responding process is selected as the leader
        if (flag == 1):
            coordinator = subcoordinator
        # else if no higher process are alive then the election generator process
        # is selected as leader
        else:
            coordinator = gid
    display()

elif schoice == 2:
    # enter process to revive
    print("Enter Process ID to be activated: ", end='')
    activate = int(input())
    # if the entered process was dead then it is revived
    if (pStatus[activate] == 0):
        pStatus[activate] = 1
    else:
        print("Process", activate, " is already alive!", end='\n')
        break
    # if the highest process is activated then it is the leader
    if (activate == n):
        coordinator = n
        break
    flag = 0
    # else, the activated process sends message to all higher process
    i = activate + 1
    while i <= n:
        print("Message is  sent from", activate, "to", i, end='\n')
```

```
            # if higher process is active then it responds
            if (pStatus[i] != 0):
                subcoordinator = i
                print("Response is sent from", i,
                      "to", activate, end='\n')
                flag = 1
            i += 1
        # the highest responding process is made the leader
        if flag == 1:
            coordinator = subcoordinator
        # if no higher process respond then the activated process is leader
        else:
            coordinator = activate
        display()

    elif schoice == 3:
        display()

    elif schoice == 4:
        pass

    condition = (schoice != 4)

def ring():
    " ring election implementation"
    global coordinator, n
    condition = True
    while condition:
        print('------------------------------------------ ')
        print("1.CRASH\n2.ACTIVATE\n3.DISPLAY\n4.EXIT")
        print('------------------------------------------ \n')
        print("Enter your choice: ", end=")
        tchoice = int(input())
```

```
if tchoice == 1:
    print("\nEnter process to crash : ", end='')
    crash = int(input())

    if pStatus[crash]:
        pStatus[crash] = 0
    else:
        print("Process", crash, "is already dead!", end='\n')
    condition = True
    while condition:
        print("Enter election generator id: ", end='')
        gid = int(input())
        if gid == coordinator:
            print("Please, enter a valid generator id!", end='\n')
        condition = (gid == coordinator)

    if crash == coordinator:
        subcoordinator = 1
        i = 0
        while i < (n+1):
            pid = (i + gid) % (n+1)
            if pid != 0:     # since our process starts from 1 (to n)
                if pStatus[pid] and subcoordinator < pid:
                    subcoordinator = pid
                print("Election message passed from", pid, ": #Msg", subcoordinator, end='\n')
            i += 1

        coordinator = subcoordinator
    display()

elif tchoice == 2:
    print("Enter Process ID to be activated: ", end='')
    activate = int(input())
```

```
            if not pStatus[activate]:
                pStatus[activate] = 1
            else:
                print("Process", activate, "is already alive!", end='\n')
                break

            subcoordinator = activate
            i = 0
            while i < (n+1):
                pid = (i + activate) % (n+1)
                if pid != 0:    # since our process starts from 1 (to n)
                    if pStatus[pid] and subcoordinator < pid:
                        subcoordinator = pid
                    print("Election message passed from", pid,
                          ": #Msg", subcoordinator, end='\n')
                i += 1

            coordinator = subcoordinator
            display()

        elif tchoice == 3:
            display()

        condition = tchoice != 4



def choice():
    """ choice of options """
    while True:
        print('------------------------------------------ ')
        print("1.BULLY ALGORITHM\n2.RING ALGORITHM\n3.DISPLAY\n4.EXIT")
        print('------------------------------------------ \n')
        fchoice = int(input("Enter your choice: "))
```

```
        if fchoice == 1:
            bully()
        elif fchoice == 2:
            ring()
        elif fchoice == 3:
            display()
        elif fchoice == 4:
            exit(0)
        else:
            print("Please, enter valid choice!")


def display():
    """ displays the processes, their status and the coordinator """
    global coordinator
    print('------------------------------------------ ')
    print("PROCESS:", end=' ')
    for i in range(1, n+1):
        print(i, end='\t')
    print('\nALIVE:', end='    ')
    for i in range(1, n+1):
        print(pStatus[i], end='\t')
    print('\n ----------------------------------------- ')
    print('COORDINATOR IS', coordinator, end='\n')
    # print(' ------------------------------------------ ')


if__name__== '__main__':

    # take_input()

    n = int(input("Enter number of processes: "))
```

```
for i in range(1, n+1):
    print("Enter Process ", i, " is alive or not(0/1): ")
    x = int(input())
    pStatus[i] = x
    if pStatus[i]:
        coordinator = i

display()
choice()
```

Output -

```
shindekalpesharun:~$ python3 bully_ring.py
Enter number of processes: 5
Enter Process  1  is alive or not(0/1):
1
Enter Process  2  is alive or not(0/1):
1
Enter Process  3  is alive or not(0/1):
1
Enter Process  4  is alive or not(0/1):
0
Enter Process  5  is alive or not(0/1):
0
---------------------------------------------
PROCESS:  1     2       3       4       5
ALIVE:    1     1       1       0       0
---------------------------------------------
COORDINATOR IS 3
---------------------------------------------
1.BULLY ALGORITHM
2.RING ALGORITHM
3.DISPLAY
4.EXIT
---------------------------------------------

Enter your choice:
```

ASSIGNMENT 7

Name- Om Patil
Roll No- 49
Class- BE(IT)

Code -

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 -

```
<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 -

s