Assign 1

Search.java

```java
package remotes;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Search extends Remote {
    public String query(String search) throws RemoteException;
}
```

SearchQuery.java

```java
package remotes;

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

public class SearchQuery extends UnicastRemoteObject implements Search {
    public SearchQuery() throws RemoteException {
        super();
    }

    public String query(String search) throws RemoteException {
        String result = "No results found";
        if (search.equals("p2p")) {
            result = "Found 1 result";
        }
        return result;
    }
}
```

SearchServer.java

```java
package server;

import java.rmi.*;
import java.rmi.registry.*;
```

```java
import remotes.Search;
import remotes.SearchQuery;

public class SearchServer {
    public static void main(String[] args) {
        try {
            Search search = new SearchQuery();
            Registry registry = LocateRegistry.createRegistry(1099);
            Naming.rebind("rmi://localhost:1099"+ "/REMOTE_SEARCH", search);
            System.out.println("Search Server ready");
        } catch (Exception e) {
            System.out.println("Search Server main " + e.getMessage());
        }
    }
}
```

ClientRequest.java

```java
package client;

import java.rmi.*;

import remotes.Search;

public class ClientRequest {
    public static void main(String[] args) {
        try {
            String search = (args.length < 1) ? "p2p" : args[0];
            String url = "rmi://localhost:1099/REMOTE_SEARCH";
            Search access = (Search) Naming.lookup(url);
            String result = access.query(search);
            System.out.println("Found: " + result);
        } catch (Exception e) {
            System.out.println("ClientRequest exception: " + e.getMessage());
        }
    }
}
```

Output

Server

```
PS D:\Acad\DS Assign\Assign1>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.2.13-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExc
eptionMessages' '-cp' 'C:\Users\kanch\AppData\Roaming\Code\User\workspaceStorage\fcadef60e121916cc193c7d2d4daad24\redhat.java\jdt_
ws\Assign1_272fc5c2\bin' 'server.SearchServer'
Search Server ready
```

Client

```
PS D:\Acad\DS Assign\Assign1>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.2.13-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExc
eptionMessages' '-cp' 'C:\Users\kanch\AppData\Roaming\Code\User\workspaceStorage\fcadef60e121916cc193c7d2d4daad24\redhat.java\jdt_
ws\Assign1_272fc5c2\bin' 'client.ClientRequest'
Found: Found 1 result
PS D:\Acad\DS Assign\Assign1>
```

Assign 2

CalculatorImpl.java

```java
package server;

import org.omg.CORBA.ORB;

import calculator_module.CalculatorPOA;

public class CalculatorImpl extends CalculatorPOA {

    private ORB orb;

    public void setORB(ORB orb_val) {
        orb = orb_val;
    }

    // implement add() method
    @Override
    public int add(int a, int b) {
        return a + b;
    }

    // implement subtract() method
    @Override
    public int subtract(int a, int b) {
        return a - b;
    }

    // implement multiply() method
    @Override
    public int multiply(int a, int b) {
        return a * b;
    }

    // implement divide() method
    @Override
    public int divide(int a, int b) {
        return a / b;
    }

    // implement shutdown() method
    @Override
```

```
        public void shutdown() {
            orb.shutdown(false);
        }

}
```

CalculatorSever.java

```
package server;

import org.omg.CORBA.ORB;
import org.omg.CosNaming.*;
import org.omg.PortableServer.*;

import calculator_module.Calculator;
import calculator_module.CalculatorHelper;


public class CalculatorServer {

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

                // create and initialize the ORB
                ORB orb = ORB.init(args, null);

                // get reference to rootpoa & activate the POAManager
                POA rootpoa = (POA)orb.resolve_initial_references("RootPOA");
                rootpoa.the_POAManager().activate();

                // create servant and register it with the ORB
                CalculatorImpl calculatorImpl = new CalculatorImpl();
                calculatorImpl.setORB(orb);

                // get object reference from the servant
                org.omg.CORBA.Object ref =
rootpoa.servant_to_reference(calculatorImpl);
                Calculator href = CalculatorHelper.narrow(ref);


                // get the root naming context
                // NameService invokes the transient name service
                org.omg.CORBA.Object objRef =
orb.resolve_initial_references("NameService");
                // Use NamingContextExt which is part of the Interoperable
                // Naming Service (INS) specification.
```

```
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            // bind the Object Reference in Naming
            String name = "Calculator";
            NameComponent path[] = ncRef.to_name( name );
            ncRef.rebind(path, href);

            System.out.println("CalculatorServer ready and waiting ... ");

            // wait for invocations from clients
            orb.run();

        } catch (Exception e) {
            System.err.println("ERROR: " + e);
            e.printStackTrace(System.out);

        } finally {
            System.out.println("CalculatorServer Exiting ... ");

        }
    }
}
```

CalculatorClient.java

```
package client;

import org.omg.CORBA.ORB;
import org.omg.CORBA.ORBPackage.InvalidName;
import org.omg.CosNaming.*;

import calculator_module.Calculator;
import calculator_module.CalculatorHelper;

public class CalculatorClient {

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

            // create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // get the root naming context
            // NameService invokes the transient name service
            org.omg.CORBA.Object objRef =
```

```
orb.resolve_initial_references("NameService");
            // Use NamingContextExt which is part of the Interoperable
            // Naming Service (INS) specification.
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            // resolve the Object Reference in Naming
            String name = "Calculator";
            Calculator calculator =
CalculatorHelper.narrow(ncRef.resolve_str(name));

            System.out.println("Obtained a handle on server object");
            System.out.println(calculator.add(1, 2));
            System.out.println(calculator.subtract(1, 2));
            System.out.println(calculator.multiply(1, 2));
            System.out.println(calculator.divide(1, 2));

        } catch (Exception e) {
            System.out.println("ERROR : " + e);
            e.printStackTrace(System.out);
        }
    }
}
```

ordb

```
PS D:\Acad\DS Assign\Assign 2> orbd -ORBInitialPort 1050 -ORBInitialHost localhost
```

Calculator Server

```
PS D:\Acad\DS Assign\Assign 2>  & 'C:\Program Files\Eclipse Adoptium\jdk-8.0.402.6-hotspot\bin\java.exe' '-cp' 'C:\Users\kanch\App
Data\Roaming\Code\User\workspaceStorage\390afc84350f74a738b803fcb9ccb67a\redhat.java\jdt_ws\Assign 2_bec8f0a1\bin' 'server.Calcula
torServer' -ORBInitialPort 1050 -ORBInitialHost localhost
CalculatorServer ready and waiting ...
```

Calculator Client

```
         ... 11 more
  PS D:\Acad\DS Assign\Assign 2>  & 'C:\Program Files\Eclipse Adoptium\jdk-8.0.402.6-hotspot\bin\java.exe' '-cp' 'C:\Users\kanch\App
  Data\Roaming\Code\User\workspaceStorage\390afc84350f74a738b803fcb9ccb67a\redhat.java\jdt_ws\Assign 2_bec8f0a1\bin' 'client.Calcula
  torClient' -ORBInitialPort 1050 -ORBInitialHost localhost
  Obtained a handle on server object
  3
  -1
  2
  0
  PS D:\Acad\DS Assign\Assign 2>
```

Assign 3

```c
#include <stdio.h>
#include <omp.h>

#define N 100
#define NUM_PROCESSORS 4

int main()
{
    int arr[N];
    for (int i = 0; i < N; i++)
    {
        arr[i] = sizeof(int) * i;
    }

    int sum = 0;
    int PARTIAL_SUM[NUM_PROCESSORS];

#pragma omp parallel num_threads(NUM_PROCESSORS)
    {
        int thread_id = omp_get_thread_num();
        int start = thread_id * (N / NUM_PROCESSORS);
        int end = (thread_id + 1) * (N / NUM_PROCESSORS);
        PARTIAL_SUM[thread_id] = 0;

        for (int i = start; i < end; i++)
        {
            PARTIAL_SUM[thread_id] += arr[i];
        }
    }

    for (int i = 0; i < NUM_PROCESSORS; i++)
    {
        sum += PARTIAL_SUM[i];
        printf("Partial sum of thread %d: %d\n", i, PARTIAL_SUM[i]);
    }

    printf("Sum: %d\n", sum);

    return 0;
}
```

```
PS D:\Acad\DS Assign\Assign3> gcc -fopenmp main.c -o output
PS D:\Acad\DS Assign\Assign3> .\output
Partial sum of thread 0: 1200
Partial sum of thread 1: 3700
Partial sum of thread 2: 6200
Partial sum of thread 3: 8700
Sum: 19800
PS D:\Acad\DS Assign\Assign3>
```

Assign 4

server.py

```python
import socket
import time
import random
import json


SERVER_IP = "127.0.0.1"
PORT = 5001



def get_local_time():
    return random.randint(int(time.time() - 1e5), int(time.time() + 1e5))



def main():
    ## Create server socket
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((SERVER_IP, PORT))
    server_socket.listen(1)

    ## Get local time
    server_local_time = get_local_time()

    print(f"Time server listening on {SERVER_IP}:{PORT}")
    print(f"Server time: {server_local_time}")

    is_client_enough = False

    clients = []

    while not is_client_enough:
        ## Accept client connection
        client_socket, client_address = server_socket.accept()
        print(f"Connection established with {client_address}")

        clients.append(client_socket)

        option = input("Do you want to add more clients? (y/n) ")
        if option == "n" or option == "N":
            is_client_enough = True
        else:
```

```python
            print("Waiting for more clients ... " + "\n")

    client_local_times = []

    ## Get local time from all clients
    for client_socket in clients:
        time_req_body = json.dumps({"operation": "time_req"})
        client_socket.send(time_req_body.encode())

        client_local_time_response = json.loads(client_socket.recv(1024).decode())

        client_local_times.append(float(client_local_time_response["client_time"]))

    ## Calculate adjusted time
    average_offset = sum(client_local_times) / len(client_local_times)
    adjusted_time_offset = (server_local_time + average_offset) / 2

    ## Send adjusted time to all clients
    for i, client_socket in enumerate(clients):
        print(
            f"Client {client_socket.getpeername()} LocalTime :
{client_local_times[i]}"
        )
        adjusted_time = json.dumps(
            {
                "adjusted_time": client_local_times[i] - adjusted_time_offset,
                "operation": "time_adj",
            }
        )

        client_socket.send(str(adjusted_time).encode())
        print(f"Adjusted time sent to {client_socket.getpeername()}")

    server_socket.close()


if __name__ == "__main__":
    main()
```

client.py

```python
import socket
import time
import json
import random
```

```python
SERVER_IP = "127.0.0.1"
PORT = 5001


def get_local_time():
    return random.randint(int(time.time() - 1e5), int(time.time() + 1e5))


def main():
    ## Connect to server
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((SERVER_IP, PORT))
    print(f"Connected to {SERVER_IP}:{PORT}")

    ## Get local time
    client_local_time = get_local_time()

    time_adjusted = False

    while not time_adjusted:
        server_res = json.loads(client_socket.recv(1024).decode())

        if server_res["operation"] == "time_req":
            ## Send local time to server
            print(f"Local time: {client_local_time}")
            client_socket.send(json.dumps({"client_time":
client_local_time}).encode())

        if server_res["operation"] == "time_adj":
            ## Adjust local time
            print(f"Time adjustment: {server_res['adjusted_time']}")
            client_local_time += float(server_res["adjusted_time"])

            print(f"Adjusted time: {client_local_time}")

            time_adjusted = True

    client_socket.close()


if __name__ == "__main__":
    main()
```

server

```
PS D:\Acad\DS Assign\Assign4> python server.py
Time server listening on 127.0.0.1:5001
Server time: 1713363824
Connection established with ('127.0.0.1', 50005)
Do you want to add more clients? (y/n) y
Waiting for more clients...

Connection established with ('127.0.0.1', 50019)
Do you want to add more clients? (y/n) []
```

Client

```
PS D:\Acad\DS Assign\Assign4> python client.py
Connected to 127.0.0.1:5001
[]
```

Assign 5

server.py

```python
import socket
import threading

TOKEN = "TOKEN"
PORT = 8080
BUFFER_SIZE = 1024


class TokenRingServer:
    def __init__(self):
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.clients = []
        self.client_threads = []
        self.running = False

    def start(self):
        self.server_socket.bind(("localhost", PORT))
        self.server_socket.listen()
        self.running = True
        print("Server started. Listening for connections ... ")

        try:
            while self.running:
                ## Accept new connections
                client_socket, client_address = self.server_socket.accept()
                print(f"New client connected: {client_address}")
                self.clients.append(client_socket)

                ## If this is the first client, send the token
                if len(self.clients) == 1:
                    # Send the token to the first client
                    client_socket.send(TOKEN.encode())

                ## Start a new thread to handle the client
                thread = threading.Thread(
                    target=self.handle_client, args=(client_socket,)
                )
                thread.start()

                self.client_threads.append(thread)
```

```python
            except KeyboardInterrupt:
                self.stop()


    def handle_client(self, client_socket):
        while self.running:
            ## Receive data from the client
            data = client_socket.recv(BUFFER_SIZE).decode()

            ## select the next client to send the token to
            next_client = self.clients[
                (self.clients.index(client_socket) + 1) % len(self.clients)
            ]

            ## If the client sends CLOSE, remove it from the list of clients and
close the connection
            if data == "CLOSE":
                print(f"Client disconnected: {client_socket.getpeername()}")
                self.clients.remove(client_socket)
                client_socket.close()
                data = TOKEN
                break

            ## If the client sends TOKEN, send it to the next client
            if data == TOKEN:
                print("Received token")
                if len(self.clients) ≥ 1:
                    if self.running:
                        print("Sending token to next client")
                        next_client.send(TOKEN.encode())

                    else:
                        print("Server stopped. Not sending token to next client")
                        break

    def stop(self):
        self.running = False

        print("Closing server..")

        ## Send close signal to all clients
        for client in self.clients:
            print(f"Sending close signal to {client.getpeername()}")
            client.send("CLOSE".encode())
            client.close()
```

```python
        ## Wait for all threads to finish
        for thread in self.client_threads:
            thread.join()

        self.server_socket.close()


if __name__ == "__main__":
    server = TokenRingServer()
    server.start()
```

Client.py

```python
import socket

SERVER_ADDRESS = ("localhost", 8080)
BUFFER_SIZE = 1024


class TokenRingClient:
    def __init__(self):
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    def connect(self):
        self.client_socket.connect(SERVER_ADDRESS)
        print("Connected to server")

    def start(self):
        try:
            while True:
                data = self.client_socket.recv(BUFFER_SIZE).decode()
                if data == "TOKEN":
                    print("Token received. Accessing resource.")
                    # Perform operations on the resource

                    # Simulating work on the resource
                    print("Working on the resource ... ")
                    # Simulating work by sleeping for 5 seconds
                    import time

                    time.sleep(5)

                    print("Resource access complete. Releasing token.")
                    self.client_socket.send("TOKEN".encode())
```

```python
            if data == "CLOSE":
                print("Closing client..")
                self.stop()
                break

        except KeyboardInterrupt:
            print("Closing client..")
            self.client_socket.send("CLOSE".encode())
            self.stop()

    def stop(self):
        self.client_socket.close()


if __name__ == "__main__":
    client = TokenRingClient()
    client.connect()
    client.start()
```

Server

```
PS D:\Acad\DS Assign\Assign5> python server.py
Server started. Listening for connections...
New client connected: ('127.0.0.1', 50291)
Received token
Sending token to next client
Received token
Sending token to next client
Received token
Sending token to next client
Received token
Sending token to next client
New client connected: ('127.0.0.1', 50296)
Received token
Sending token to next client
Received token
Sending token to next client
New client connected: ('127.0.0.1', 50297)
Received token
Sending token to next client
Received token
```

Clients

Client 1

```
PS D:\Acad\DS Assign\Assign5> python client.py
Connected to server
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
```

Client 2

```
PS D:\Acad\DS Assign\Assign5> python client.py
Connected to server
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
```

Client 3

```
PS D:\Acad\DS Assign\Assign5> python client.py
Connected to server
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
Token received. Accessing resource.
Working on the resource...
Resource access complete. Releasing token.
```

Assign 6

bully.py

```python
class Bully:
    def __init__(self, num_process=5):
        # Initialize the Bully object with the number of processes and their states
        self.num_process = num_process
        self.state = [True for _ in range(num_process)]
        self.leader = num_process

    def election(self, process_id):
        # Perform the election algorithm to elect a coordinator
        print(f"Process {process_id} is sending election messages to higher
processes")
        cod = process_id
        for i in range(process_id + 1, self.num_process + 1):
            if self.state[i - 1]:
                print(
                    f"Process {process_id} is sending election message to process
{i}"
                )
                cod = i

        print(f"Process {cod} is sending coordinator message to all")

        # Update the leader to the elected coordinator
        self.leader = cod
        print(f"Process {self.leader} is now coordinator.")

    def up(self, process_id):
        # Bring up a process and trigger an election if necessary
        if self.state[process_id - 1]:
            print(f"Process {process_id} is already up")
            return
        else:
            self.state[process_id - 1] = True
            print(f"Process {process_id} is up")
            self.election(process_id)

    def down(self, process_id):
        # Bring down a process and initiate a new election if the leader is down
        if not self.state[process_id - 1]:
            print(f"Process {process_id} is already down.")
```

```python
        else:
            self.state[process_id - 1] = False
            print(f"Process {process_id} is now down")

            if self.leader == process_id:
                # If the leader is down, randomly select a new active process and
trigger an election
                active = [i for i, _ in enumerate(self.state) if i]
                import random

                index = random.randint(0, len(active) - 1)
                self.election(active[index])

    def message(self, process_id):
        # Send a message and check if the coordinator is active
        if self.state[process_id - 1]:
            if self.state[self.leader - 1]:
                print("OK")
            else:
                # If the coordinator is down, initiate a new election
                self.election(process_id)
        else:
            print(f"Process {process_id} is down.")


if __name__ == "__main__":
    # Create a Bully object
    bully = Bully()

    print("5 Active processes are:")
    print("Processes up = p1 p2 p3 p4 p5")
    print(f"Process {bully.leader} is the coordinator")

    choice = 5

    while choice != 4:
        print("——————————————————————————————")
        print("1) Up a process")
        print("2) Down a Process")
        print("3) Send a Message")
        print("4) Exit")

        choice = int(input("Enter choice: "))

        if choice == 1:
            process_id = int(input("Enter process id: "))
```

```
            bully.up(process_id)

        elif choice == 2:
            process_id = int(input("Enter process id: "))
            bully.down(process_id)

        elif choice == 3:
            process_id = int(input("Enter process id: "))
            bully.message(process_id)

        else:
            break
```

ring.py

```
class Ring:
    def __init__(self, num_process=5):
        self.num_process = num_process
        self.coordinator = 5
        self.active_processes = set(range(1, num_process + 1))

    def election(self, process_id):
        if self.coordinator is None:
            # Only one process in the system
            self.coordinator = process_id
            print(f"Process {process_id} is the coordinator.")
            return

        if process_id not in self.active_processes:
            print(f"Process {process_id} is not active.")
            return

        highest_id = process_id
        next_process = (process_id % self.num_process) + 1

        while next_process ≠ process_id:
            if next_process in self.active_processes:
                print(
                    f"Process {process_id} is passing election message to process
{next_process}."
                )
                if next_process > highest_id:
                    highest_id = next_process
            else:
```

```python
                print(
                    f"Process {next_process} is down and cannot receive the
election message."
                )
            next_process = (next_process % self.num_process) + 1

        self.coordinator = highest_id
        print(f"Process {self.coordinator} is the coordinator.")

    def start_election(self, process_id):
        if process_id not in self.active_processes:
            print(f"Process {process_id} is not active.")
            return

        print(f"Process {process_id} starts the election process.")
        self.election(process_id)

    def bring_up_process(self, process_id):
        if process_id in self.active_processes:
            print(f"Process {process_id} is already up.")
            return

        self.active_processes.add(process_id)
        print(f"Process {process_id} is up.")

    def bring_down_process(self, process_id):
        if process_id not in self.active_processes:
            print(f"Process {process_id} is already down.")
            return

        self.active_processes.remove(process_id)
        print(f"Process {process_id} is now down.")

        if self.coordinator == process_id:
            self.start_election(process_id)

    def print_active_processes(self):
        print("Active processes:")
        for process_id in self.active_processes:
            print(f"Process {process_id}")

    def print_coordinator(self):
        if self.coordinator is None:
            print("Coordinator: None")
        else:
            print(f"Coordinator: Process {self.coordinator}")
```

```python
if __name__ == "__main__":
    ring = Ring()

    while True:
        print("——————————————————————————————")
        print("1) Start Election")
        print("2) Bring Up Process")
        print("3) Bring Down Process")
        print("4) Print Active Processes")
        print("5) Print Coordinator")
        print("6) Exit")

        choice = int(input("Enter choice: "))

        if choice == 1:
            process_id = int(input("Enter process id to start the election: "))
            ring.start_election(process_id)

        elif choice == 2:
            process_id = int(input("Enter process id to bring up: "))
            ring.bring_up_process(process_id)

        elif choice == 3:
            process_id = int(input("Enter process id to bring down: "))
            ring.bring_down_process(process_id)

        elif choice == 4:
            ring.print_active_processes()

        elif choice == 5:
            ring.print_coordinator()

        else:
            break
```

bully

```
PS D:\Acad\DS Assign\Assign6> python bully.py
5 Active processes are:
Processes up = p1 p2 p3 p4 p5
Process 5 is the coordinator
-------------------------------------
1) Up a process
2) Down a Process
3) Send a Message
4) Exit
Enter choice: 3
Enter process id: 2
OK
-------------------------------------
1) Up a process
2) Down a Process
3) Send a Message
4) Exit
Enter choice: █
```

ring

```
PS D:\Acad\DS Assign\Assign6> python ring.py
-------------------------------------
1) Start Election
2) Bring Up Process
3) Bring Down Process
4) Print Active Processes
5) Print Coordinator
6) Exit
Enter choice: 4
Active processes:
Process 1
Process 2
Process 3
Process 4
Process 5
-------------------------------------
1) Start Election
2) Bring Up Process
3) Bring Down Process
4) Print Active Processes
5) Print Coordinator
6) Exit
Enter choice: 1
Enter process id to start the election: 2
Process 2 starts the election process.
Process 2 is passing election message to process 3.
Process 2 is passing election message to process 4.
Process 2 is passing election message to process 5.
Process 2 is passing election message to process 1.
Process 5 is the coordinator.
-------------------------------------
1) Start Election
2) Bring Up Process
3) Bring Down Process
4) Print Active Processes
```

Assign 7

api.py

```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/add", methods=["POST"])
def add():
    data = request.get_json()
    num1 = data["num1"]
    num2 = data["num2"]
    result = num1 + num2
    return str(result)


@app.route("/subtract", methods=["POST"])
def subtract():
    data = request.get_json()
    num1 = data["num1"]
    num2 = data["num2"]
    result = num1 - num2
    return str(result)


@app.route("/multiply", methods=["POST"])
def multiply():
    data = request.get_json()
    num1 = data["num1"]
    num2 = data["num2"]
    result = num1 * num2
    return str(result)


@app.route("/divide", methods=["POST"])
def divide():
    data = request.get_json()
    num1 = data["num1"]
    num2 = data["num2"]
    result = num1 / num2
    return str(result)
```

```
if __name__ == "__main__":
    app.run(debug=True)
```

app.py

```python
from flask import Flask, render_template, request
import requests
import json

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/calculate", methods=["POST"])
def calculate():
    num1 = int(request.form["num1"])
    num2 = int(request.form["num2"])
    operation = request.form["operation"]

    payload = {"num1": num1, "num2": num2}

    if operation == "add":
        url = "http://localhost:5000/add"
    elif operation == "subtract":
        url = "http://localhost:5000/subtract"
    elif operation == "multiply":
        url = "http://localhost:5000/multiply"
    elif operation == "divide":
        url = "http://localhost:5000/divide"

    response = requests.post(url, json=payload)
    result = json.loads(response.text)

    return render_template("result.html", result=result)

if __name__ == "__main__":
    app.run(debug=True, port=3000)
```

index.html

```html
<!DOCTYPE html>
<html>
    <head>
```

```html
        <title>Calculator Web App</title>
    </head>
    <body>
        <h1>Calculator Web App</h1>
        <form action="/calculate" method="POST">
            <label for="num1">Number 1:</label>
            <input type="number" id="num1" name="num1" required /><br />

            <label for="num2">Number 2:</label>
            <input type="number" id="num2" name="num2" required /><br />

            <label for="operation">Operation:</label>
            <select id="operation" name="operation" required>
                <option value="add">Addition</option>
                <option value="subtract">Subtraction</option>
                <option value="multiply">Multiplication</option>
                <option value="divide">Division</option></select
            ><br />

            <input type="submit" value="Calculate" />
        </form>
    </body>
</html>
```

result.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Calculator Web App – Result</title>
    </head>
    <body>
        <h1>Calculator Web App – Result</h1>
        <p>The result is: {{ result }}</p>
        <a href="/">Go Back</a>
    </body>
</html>
```
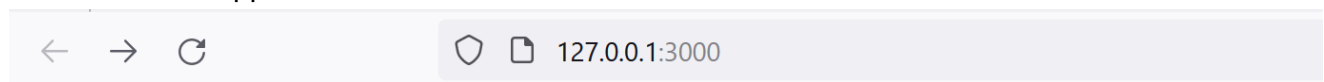
api

```
PS D:\Acad\DS Assign\Assign7> python api.py
 * Serving Flask app 'api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 917-801-138
127.0.0.1 - - [16/Apr/2024 18:15:16] "POST /multiply HTTP/1.1" 200 -
```

app

```
PS D:\Acad\DS Assign\Assign7> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 917-801-138
127.0.0.1 - - [16/Apr/2024 18:15:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 18:15:01] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [16/Apr/2024 18:15:16] "POST /calculate HTTP/1.1" 200 -
```
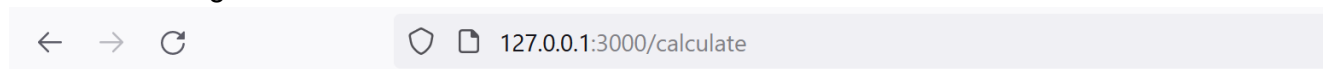
Calculator web app

```
←  →  C                    🛡  📄  127.0.0.1:3000
```

Calculator Web App

Number 1: `33`
Number 2: `42`
Operation: `Multiplication ⌄`
`Calculate`

Result Web Page

```
←  →  C                    🛡  📄  127.0.0.1:3000/calculate
```

Calculator Web App - Result

The result is: 1386

[Go Back](#)