Assignment 1(A):

CODE :

MyClient.java :

```
import java.net.*;
import java.io.*;

public class MyClient {
   public static void main(String[] args) throws Exception{

      //The socket object takes ip and port number of the server which client wants to connect
      Socket s = new Socket("127.0.0.1",5555);
      System.out.println("Connected to Server, Please type your message and hit Enter to send");

      //Reading input from KeyBoard
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      //OutputStream object to write to Server
      OutputStream ostream = s.getOutputStream();

      //PrintWriter object to send the data to the outputstream
      PrintWriter pw = new PrintWriter(ostream, true);

      //InputStream objects to recieve from Server
      InputStream istream = s.getInputStream();

      //Reading receieved message from Server
      BufferedReader recieve = new BufferedReader(new InputStreamReader(istream));

      //Client Message and Server Message objects
      String clientmessage = "";
      String servermessage = "";

      while(true)
      {
         //Input Message to be sent to Server
         System.out.print("Client: ");
         clientmessage = br.readLine();

         //print writer object sending the message to the socket through outputstream
         pw.println(clientmessage);

         //if the message is bye end the communication here
         if(clientmessage.equals("bye"))
         {
            break;
         }

         //Read the inputstream of the server from the socket
         servermessage = recieve.readLine();
```

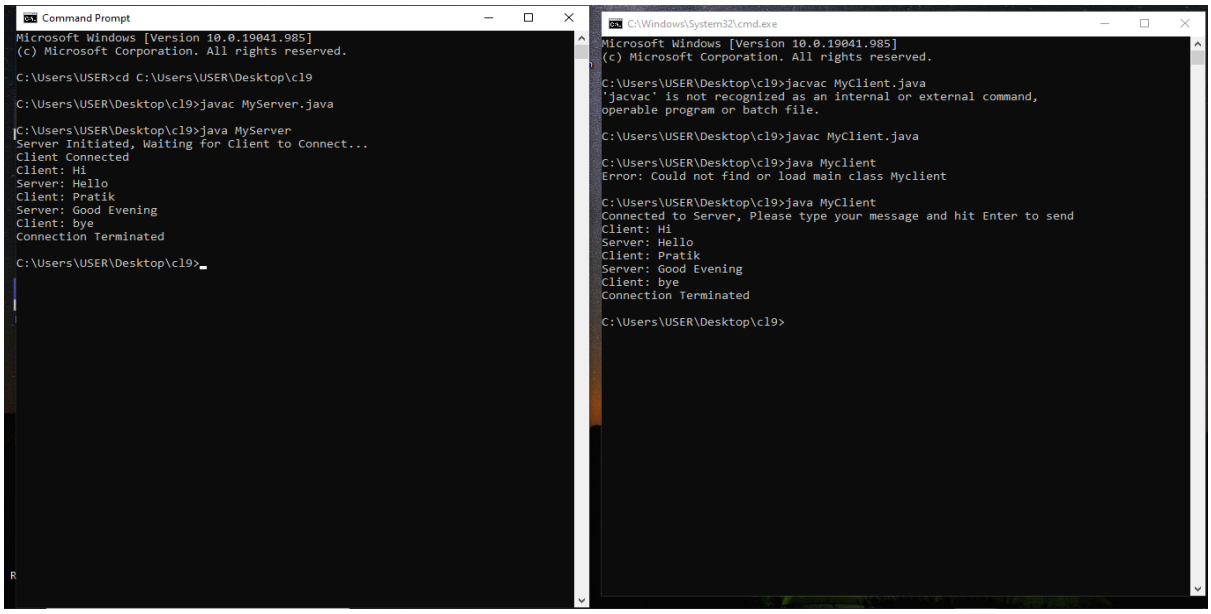
```
        System.out.println("Server: "+servermessage);

        //if the message is bye end the communication here
        if(servermessage.equals("bye"))
        {
            break;
        }
    }
    //closing all the streams and sockets
    s.close();
    istream.close();
    ostream.close();

    System.out.println("Connection Terminated");
  }
}
```

MyServer.java:

```
import java.net.*;
import java.io.*;

public class MyServer {
  public static void main(String[] args) throws Exception{

    //Creating a port for communication
    ServerSocket ss = new ServerSocket(5555);
    System.out.println("Server Initiated, Waiting for Client to Connect...");

    //Binding Client and Server on port 5555
    Socket s = ss.accept();
    System.out.println("Client Connected");

    //Reading input from KeyBoard
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    //OutputStream object to write to clients
    OutputStream ostream = s.getOutputStream();

    //PrintWriter object to send the data to the outputstream
    PrintWriter pw = new PrintWriter(ostream,true);

    //InputStream objects to recieve from Client
    InputStream istream = s.getInputStream();

    //Reading receieved message from client
    BufferedReader recieve = new BufferedReader(new InputStreamReader(istream));

    //Client Message and Server Message objects
    String servermessage = "";
    String clientmessage = "";
```

```
        while(true)
        {
            //Read the inputstream of the client from the socket
            clientmessage = recieve.readLine();
            System.out.println("Client: "+clientmessage);

            //if the message is bye end the communication here
            if(clientmessage.equals("bye"))
            {
                Break; }

            //Server writing its message
            System.out.print("Server: ");
            servermessage = br.readLine();
            //print writer object sending the message to the socket through outputstream
            pw.println(servermessage);
            if(servermessage.equals("bye"))
            {
                break;
            }
        }
        //closing all the streams and sockets
        s.close();
        ss.close();
        istream.close();
        ostream.close();

        System.out.println("Connection Terminated");
    }
}
```

Output :



Assignment 1(B):

CODE :

Clinet.java:

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

public class Client {
   public static void main(String args[]) {
      try {
         Scanner s = new Scanner(System.in);
         System.out.println("Enter the Server address : ");
         String server = s.nextLine();
         ServerInterface si = (ServerInterface) Naming.lookup("rmi://" + server + "/Server");
         System.out.println("Enter first string : ");
         String first = s.nextLine();
         System.out.println("Enter second string : ");
         String second = s.nextLine();
         System.out.println("Concatenated String : " + si.concat(first, second));
         s.close();
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

Servent.java:

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

public class Servant extends UnicastRemoteObject implements ServerInterface {
   protected Servant() throws RemoteException {
      super();
   }

   @Override
   public String concat(String a, String b) throws RemoteException {
      return a + b;
   }
}
```

Server.java:

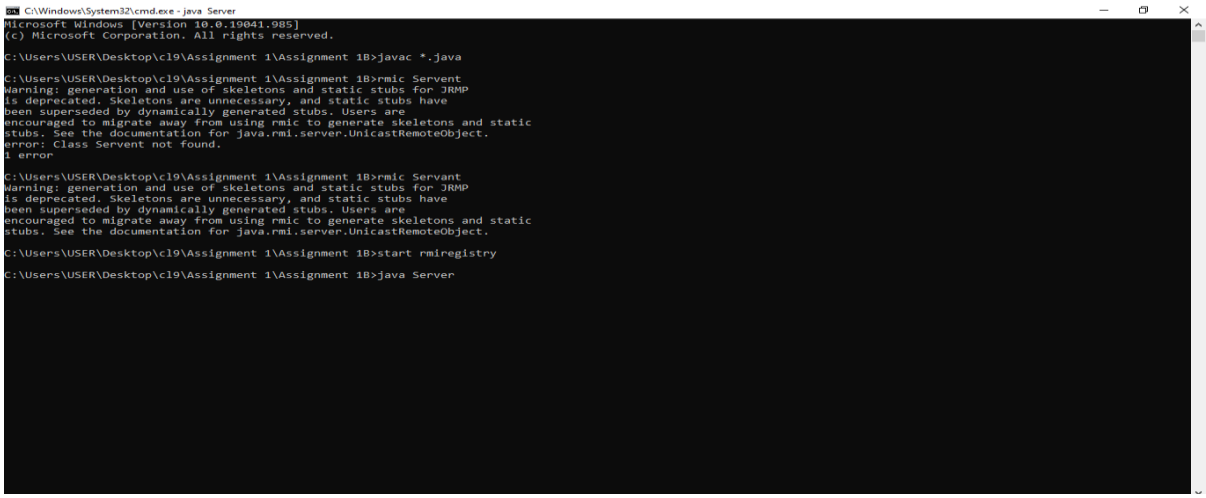
```
import java.rmi.*;
import java.net.*;
```

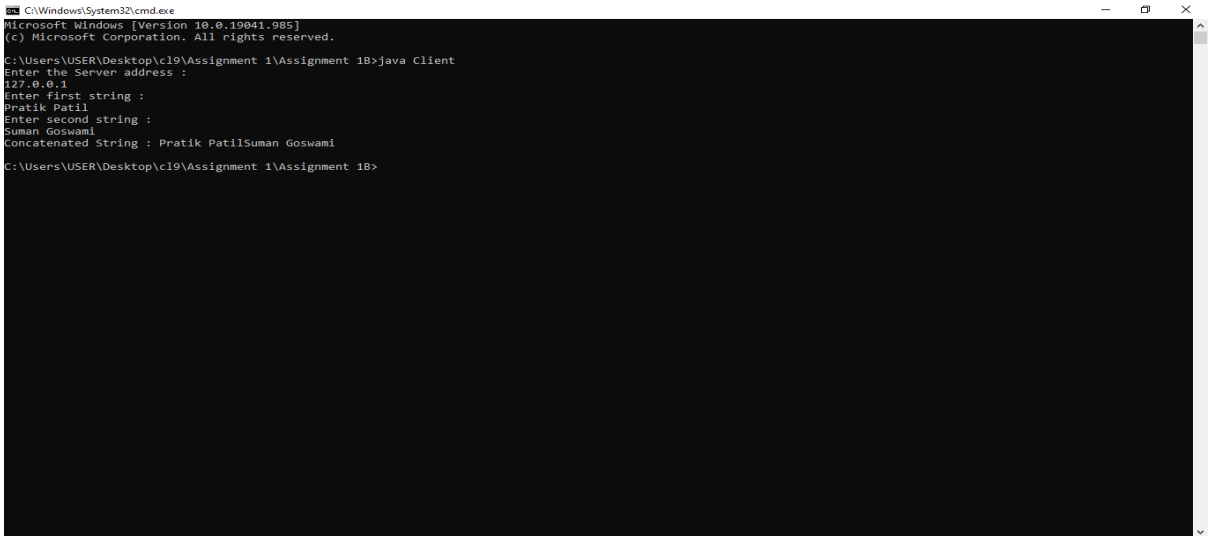
```
public class Server {
   public static void main(String[] args) {
      try {
         Servant s = new Servant();
         Naming.rebind("Server", s);
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

ServerInterface.java:

```
import java.rmi.*;

public interface ServerInterface extends Remote {

   String concat(String a, String b) throws RemoteException;

}
```

Output :





Assignment 2:

CODE :

_HelloStub.java:

```
package HelloModule;

public class _HelloStub extends org.omg.CORBA.portable.ObjectImpl implements HelloModule.Hello
{

  public String print_hello (String s)
  {
        org.omg.CORBA.portable.InputStream $in = null;
        try {
           org.omg.CORBA.portable.OutputStream $out = _request ("print_hello", true);
           $out.write_string (s);
           $in = _invoke ($out);
           String $result = $in.read_string ();
           return $result;
        } catch (org.omg.CORBA.portable.ApplicationException $ex) {
           $in = $ex.getInputStream ();
           String _id = $ex.getId ();
           throw new org.omg.CORBA.MARSHAL (_id);
        } catch (org.omg.CORBA.portable.RemarshalException $rm) {
           return print_hello (s       );
        } finally {
           _releaseReply ($in);
        }
  } // print_hello

  // Type-specific CORBA::Object operations
  private static String[] __ids = {
    "IDL:HelloModule/Hello:1.0"};

  public String[] _ids ()
  {
    return (String[])__ids.clone ();
  }

  private void readObject (java.io.ObjectInputStream s) throws java.io.IOException
  {
    String str = s.readUTF ();
    String[] args = null;
    java.util.Properties props = null;
    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init (args, props);
   try {
    org.omg.CORBA.Object obj = orb.string_to_object (str);
    org.omg.CORBA.portable.Delegate delegate = ((org.omg.CORBA.portable.ObjectImpl) obj)._get_delegate ();
    _set_delegate (delegate);
   } finally {
    orb.destroy() ;
```

```
  }
 }

 private void writeObject (java.io.ObjectOutputStream s) throws java.io.IOException
 {
   String[] args = null;
   java.util.Properties props = null;
   org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init (args, props);
  try {
   String str = orb.object_to_string (this);
   s.writeUTF (str);
  } finally {
   orb.destroy() ;
  }
 }
}
```

Hello.java:

```
package HelloModule;

public interface Hello extends HelloOperations, org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity
{
} // interface Hello
```

HelloImpl.java:

```
import HelloModule.HelloPOA;

class HelloImpl extends HelloPOA{
   HelloImpl()
   {
      super();
      System.out.println("Ready");
   }

   public String print_hello(String s)
   {
      return("Hello "+s);
   }
}
```

HelloHelper.java:

```
package HelloModule;

abstract public class HelloHelper
{
 private static String  _id = "IDL:HelloModule/Hello:1.0";

 public static void insert (org.omg.CORBA.Any a, HelloModule.Hello that)
 {
   org.omg.CORBA.portable.OutputStream out = a.create_output_stream ();
```

```
  a.type (type ());
  write (out, that);
  a.read_value (out.create_input_stream (), type ());
}

public static HelloModule.Hello extract (org.omg.CORBA.Any a)
{
  return read (a.create_input_stream ());
}

private static org.omg.CORBA.TypeCode __typeCode = null;
synchronized public static org.omg.CORBA.TypeCode type ()
{
  if (__typeCode == null)
  {
    __typeCode = org.omg.CORBA.ORB.init ().create_interface_tc (HelloModule.HelloHelper.id (), "Hello");
  }
  return __typeCode;
}

public static String id ()
{
  return _id;
}

public static HelloModule.Hello read (org.omg.CORBA.portable.InputStream istream)
{
  return narrow (istream.read_Object (_HelloStub.class));
}

public static void write (org.omg.CORBA.portable.OutputStream ostream, HelloModule.Hello value)
{
  ostream.write_Object ((org.omg.CORBA.Object) value);
}

public static HelloModule.Hello narrow (org.omg.CORBA.Object obj)
{
  if (obj == null)
    return null;
  else if (obj instanceof HelloModule.Hello)
    return (HelloModule.Hello)obj;
  else if (!obj._is_a (id ()))
    throw new org.omg.CORBA.BAD_PARAM ();
  else
  {
    org.omg.CORBA.portable.Delegate delegate = ((org.omg.CORBA.portable.ObjectImpl)obj)._get_delegate ();
    HelloModule._HelloStub stub = new HelloModule._HelloStub ();
    stub._set_delegate(delegate);
    return stub;
  }
}

public static HelloModule.Hello unchecked_narrow (org.omg.CORBA.Object obj)
```

```
  {
   if (obj == null)
     return null;
   else if (obj instanceof HelloModule.Hello)
     return (HelloModule.Hello)obj;
   else
   {
     org.omg.CORBA.portable.Delegate delegate = ((org.omg.CORBA.portable.ObjectImpl)obj)._get_delegate ();
     HelloModule._HelloStub stub = new HelloModule._HelloStub ();
     stub._set_delegate(delegate);
     return stub;
   }
  }

}
```

HelloHolder.java:

```
package HelloModule;

public final class HelloHolder implements org.omg.CORBA.portable.Streamable
{
  public HelloModule.Hello value = null;

  public HelloHolder ()
  {
  }

  public HelloHolder (HelloModule.Hello initialValue)
  {
   value = initialValue;
  }

  public void _read (org.omg.CORBA.portable.InputStream i)
  {
   value = HelloModule.HelloHelper.read (i);
  }

  public void _write (org.omg.CORBA.portable.OutputStream o)
  {
   HelloModule.HelloHelper.write (o, value);
  }

  public org.omg.CORBA.TypeCode _type ()
  {
   return HelloModule.HelloHelper.type ();
  }

}
```

HelloOperations.java:

```
package HelloModule;

public interface HelloOperations
{
  String print_hello (String s);
} // interface HelloOperations
```

HelloPOA.java:

```
public abstract class HelloPOA extends org.omg.PortableServer.Servant
 implements HelloModule.HelloOperations, org.omg.CORBA.portable.InvokeHandler
{
  // Constructors
  private static java.util.Hashtable _methods = new java.util.Hashtable ();
  static
  {
    _methods.put ("print_hello", new java.lang.Integer (0));
  }

  public org.omg.CORBA.portable.OutputStream _invoke (String $method,
                                org.omg.CORBA.portable.InputStream in,
                                org.omg.CORBA.portable.ResponseHandler $rh)
  {
    org.omg.CORBA.portable.OutputStream out = null;
    java.lang.Integer __method = (java.lang.Integer)_methods.get ($method);
    if (__method == null)
      throw new org.omg.CORBA.BAD_OPERATION (0, org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);

    switch (__method.intValue ())
    {
      case 0:  // HelloModule/Hello/print_hello
      {
        String s = in.read_string ();
        String $result = null;
        $result = this.print_hello (s);
        out = $rh.createReply();
        out.write_string ($result);
        break;
      }

      default:
        throw new org.omg.CORBA.BAD_OPERATION (0,
org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);
    }

    return out;
  } // _invoke

  // Type-specific CORBA::Object operations
  private static String[] __ids = {
    "IDL:HelloModule/Hello:1.0"};
```

```
  public String[] _all_interfaces (org.omg.PortableServer.POA poa, byte[] objectId)
  {
    return (String[])__ids.clone ();
  }

  public Hello _this()
  {
    return HelloHelper.narrow(
    super._this_object());
  }

  public Hello _this(org.omg.CORBA.ORB orb)
  {
    return HelloHelper.narrow(
    super._this_object(orb));
  }
}
```

Client.java:

```
import HelloModule.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        Hello HelloImpl = null;
        try {

            // create and initialize ORB
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

            //obtaining the ORB object references for initial services
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");

            //Naming ContextExt contains set of name bindings of Interoperable Naming services
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            //We have binded the name Hello from server so using same name for lookup
            String name = "Hello";

            //Getting reference of server name hello and then we are narrowing it down to Hello type
            HelloImpl = HelloHelper.narrow(ncRef.resolve_str(name));

            //Taking user Input
            System.out.println("Enter your name: ");
            Scanner sc = new Scanner(System.in);
            String userName = sc.nextLine();
```

```
        //Invoking the print_hello
        System.out.println(HelloImpl.print_hello(userName));

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

Server.java:

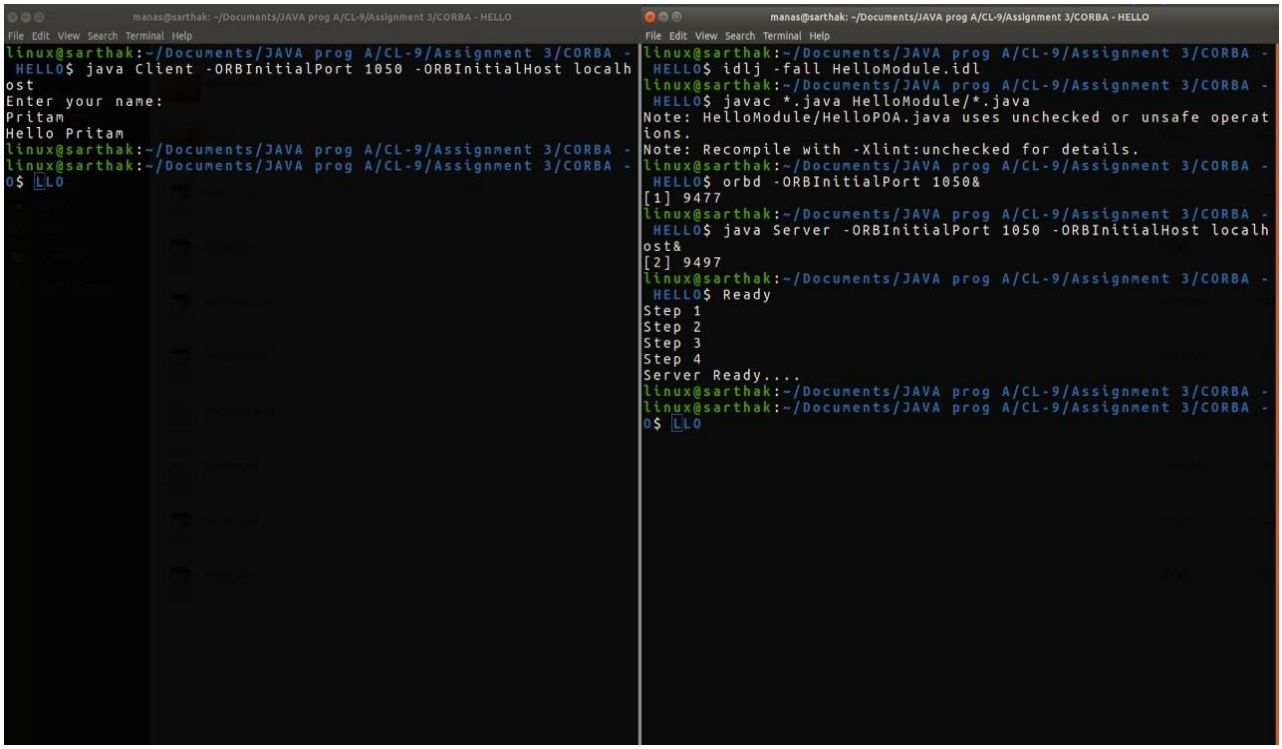
```
import HelloModule.Hello;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;

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

        // create and initialize ORB
        org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

        //Getting reference of ROOTPOA
        POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

        //Activating ROOTPOA
        rootPOA.the_POAManager().activate();

        //Create Object of Interface implementation which will act as servant
        HelloImpl helloImpl = new HelloImpl();

        //Registering the servant object reference in the rootPOA
        org.omg.CORBA.Object ref = rootPOA.servant_to_reference(helloImpl);

        //narrowing the ROOTPOA reference object to propertype which in this case is of type Hello
        System.out.println("Step 1");
        Hello h_ref = HelloModule.HelloHelper.narrow(ref);

        //obtaining the ORB object references for initial services
        System.out.println("Step 2");
        org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");

        //Afain narrowing the ORB object reference to NamingContext type to bin it with server
        System.out.println("Step 3");
        NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

        //passing path and the servant object to the naming service, binding the servant object to the "Hello"
        System.out.println("Step 4");
        String name = "Hello";
        NameComponent path[] = ncRef.to_name(name);
        ncRef.rebind(path,h_ref);
```

```
        //Enbaling ORB to run on main thread and waiting till invocation comes for ORB. Since it is in main
method after invocation it will wait again
        System.out.println("Server Ready....");
        orb.run();

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

HelloModule.idl:

```
module HelloModule{
   interface Hello{
      string print_hello(in string s);
   };
};
```

Output:



Assignment 3:

Code for Adding N(N=4) numbers in array using 4 cores:

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char* argv[])
{
 int rank, size;
 int num[20]; //N=20, n=4

 MPI_Init(&argc, &argv);
 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
 MPI_Comm_size(MPI_COMM_WORLD, &size);
 for(int i=0;i<20;i++){
   num[i]=i+1;
 }

 if(rank == 0){
   int s[4];
   printf("Distribution at rank %d \n", rank);
   for(int i=1;i<4;i++){
   MPI_Send(&num[i*5], 5, MPI_INT, i, 1, MPI_COMM_WORLD); //N/n i.e. 20/4=5
   int sum=0, local_sum=0;
   for(int i=0;i<5;i++){
      local_sum=local_sum+num[i];
   }
   for(int i=1;i<4;i++){
      MPI_Recv(&s[i], 1, MPI_INT, i, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
   }
   printf("local sum at rank %d is %d\n", rank,local_sum);
   sum=local_sum;

   for(int i=1;i<4;i++){
      sum=sum+s[i];
      printf("final sum = %d\n\n",sum);
   }
 }
 else{
   int k[5];
   MPI_Recv(k, 5, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
   int local_sum=0;
   for(int i=0;i<5;i++){
      local_sum=local_sum+k[i];
   }
   printf("local sum at rank %d is %d\n", rank, local_sum);
   MPI_Send(&local_sum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
```

```
 }
 MPI_Finalize();
 return 0;
}
```

Output :

```
Distribution at rank 0
local sum at rank 1 is 40
local sum at rank 2 is 65
local sum at rank 3 is 90
local sum at rank 0 is 15
final sum = 210
```

Assignment 4:

Client.cpp:

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#include <iostream>
#include <stdlib.h>     /* srand, rand */
#include <cstdlib>
#include <ctime>
#include <vector>

#define PORT 8080

using namespace std;

// function for string delimiter
vector<string> split(string s, string delimiter) {
   size_t pos_start = 0, pos_end, delim_len = delimiter.length();
   string token;
   vector<string> res;

   while ((pos_end = s.find (delimiter, pos_start)) != string::npos) {
      token = s.substr (pos_start, pos_end - pos_start);
      pos_start = pos_end + delim_len;
      res.push_back (token);
   }
   res.push_back (s.substr (pos_start));
   return res;
}

int main(int argc, char const *argv[])
{

   srand((unsigned int)time(NULL)); // avoid always same output of rand()
   float client_local_clock = rand() % 10; // range from 0 to 9
   printf("Client starts. Client pid is %d \n", getpid());
   printf("Client local clock is %f \n\n", client_local_clock);

   int client_socket_fd, valread;
   char client_read_buffer[1024] = {0};
```

```
struct sockaddr_in server_addr;
server_addr.sin_family = AF_INET;
// server_addr.sin_addr.s_addr = inet_addr(argv[1]); // hardcode to 127.0.0.1
server_addr.sin_port = htons(PORT);


// Creating socket file descriptor (IPv4, TCP, IP)
if ((client_socket_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
   printf("\n Client: Socket creation error \n");
   return -1;
}
if(inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr)<=0)
{
   printf("\nClient: Invalid address/ Address not supported \n");
   return -1;
}

// Connecting server, return 0 with success, return -1 with error
if (connect(client_socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0)
{
   printf("\nClient: Connection Failed \n");
   return -1;
}

char server_ip[INET_ADDRSTRLEN]="";
inet_ntop(AF_INET, &server_addr.sin_addr, server_ip, INET_ADDRSTRLEN);
printf("Client: connected server(%s:%d). \n", server_ip, ntohs(server_addr.sin_port));
printf("\n\n");

// first round communicattion
// receiving form server
valread = read( client_socket_fd , client_read_buffer, 1024);
printf("Client: read: '%s'\n",client_read_buffer );

// convert char array to string
string recv_msg = string(client_read_buffer);

// reply according to what client receive
if (strcmp(client_read_buffer, "Hello from server, please tell me your local clock value.") == 0) {
   // prepare msg
   string msg_str = "Hello from client, my local clock value is " + to_string(client_local_clock);
   char msg_char_array[msg_str.length() + 1];
   strcpy(msg_char_array, msg_str.c_str());
```

```
  // sending a message to server
      send(client_socket_fd , &msg_char_array , strlen(msg_char_array) , 0 );
      printf("Client: sent message: '%s'\n", msg_char_array);
   }

   //
   // second round communicattion
   //

   // receiving form server
   valread = read( client_socket_fd , client_read_buffer, 1024);
   printf("Client: read: '%s'\n",client_read_buffer );

   // convert char array to string
   recv_msg = string(client_read_buffer);

   if (recv_msg.find("From server, your clock adjustment offset is") != string::npos){
      string substr_after_lastbutone_space;
      string substr_after_last_space;
      vector<string> split_str = split(recv_msg, " ");
      substr_after_lastbutone_space = split_str[ split_str.size() - 2 ];
      substr_after_last_space = split_str[ split_str.size() - 1 ];

      cout << "Client: received local clock adjustment offset (string) is " <<
substr_after_lastbutone_space << " " << substr_after_last_space << endl;
      float substr_after_last_space_f = stof(substr_after_last_space);
      cout << "Client: received local clock adjustment offset (float) is " <<
substr_after_lastbutone_space << " " << substr_after_last_space_f << endl;

      char oper_char_array[substr_after_lastbutone_space.length() + 1];
      strcpy(oper_char_array, substr_after_lastbutone_space.c_str());
      if (strcmp(oper_char_array, "add") == 0 ){
         client_local_clock += substr_after_last_space_f;
      }else if (strcmp(oper_char_array, "minus") == 0 ){
         client_local_clock -= substr_after_last_space_f;
      }

      printf("Client local clock is %f \n\n", client_local_clock);
   }
      close(client_socket_fd);
   return 0;
}
```

Server.cpp:

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
#include <vector>
#include <cstdlib>
#include <ctime>

#define PORT 8080

using namespace std;

// function for string delimiter
vector<string> split(string s, string delimiter) {
   size_t pos_start = 0, pos_end, delim_len = delimiter.length();
   string token;
   vector<string> res;

   while ((pos_end = s.find (delimiter, pos_start)) != string::npos) {
      token = s.substr (pos_start, pos_end - pos_start);
      pos_start = pos_end + delim_len;
      res.push_back (token);
   }

   res.push_back (s.substr (pos_start));
   return res;
}


int main(int argc, char *argv[])
{

   srand((unsigned int)time(NULL)); // avoid always same output of rand()
   float server_local_clock = rand() % 10; // range from 0 to 9
   vector<float> clients_local_clocks;
   printf("Sever starts. Server pid is %d \n", getpid());
   printf("Server local clock is %f \n\n", server_local_clock);
```

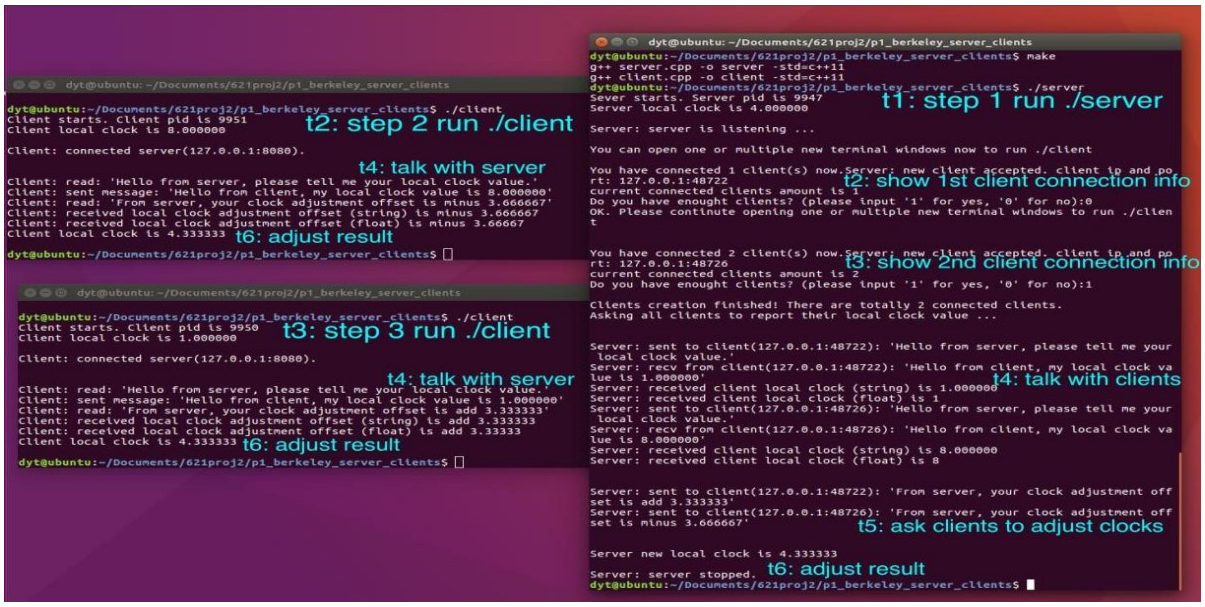
```
// Socket Cite: https://www.geeksforgeeks.org/socket-programming-cc/?ref=lbp
int server_socket_fd, new_socket, valread;
vector<int> client_sockets;
vector<string> client_ips;
vector<int> client_ports;
struct sockaddr_in server_address;
server_address.sin_family = AF_INET;    // IPv4
server_address.sin_addr.s_addr = INADDR_ANY; // localhost
server_address.sin_port = htons( PORT ); // 8080
int opt = 1; // for setsockopt

// Creating socket file descriptor (IPv4, TCP, IP)
if ((server_socket_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
{
    perror("Server: socket failed");
    exit(EXIT_FAILURE);
}

// Optional: it helps in reuse of address and port. Prevents error such as: “address already in use”.
if (setsockopt(server_socket_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                                &opt, sizeof(opt)))
{
    perror("Server: setsockopt");
    exit(EXIT_FAILURE);
}

// Forcefully attaching socket to the port 8080
if (bind(server_socket_fd, (struct sockaddr *)&server_address,
                        sizeof(server_address))<0)
{
    perror("Server: bind failed");
    exit(EXIT_FAILURE);
}
if (listen(server_socket_fd, 7) < 0)
{
    perror("Server: listen");
    exit(EXIT_FAILURE);
}
printf("Server: server is listening ...\n\nYou can open one or multiple new terminal windows now
to run ./client\n");
int clients_ctr = 0;
```

```
   // Setting up buffer for receiving msg
   char recv_buf[65536];
   memset(recv_buf, '\0', sizeof(recv_buf));

   int in_client_enough = 0;
   while ( in_client_enough == 0) { // block on accept() until positive fd or error
      struct sockaddr_in client_addr;
      socklen_t length = sizeof(client_addr);
      // Extracting the first connection request on the queue of pending connections for the listening
socket (server_socket_fd)
      // Creates a new connected socket, and returns a new file descriptor referring to that socket
      if ((new_socket = accept(server_socket_fd, (struct sockaddr *)&client_addr,
                  (socklen_t*)&length))<0)
      {
         perror("Server: accept");
         exit(EXIT_FAILURE);
      }

      clients_ctr ++;
      printf("\nYou have connected %d client(s) now.", clients_ctr);

      // converting the network address structure src in the af address family into a character string.
      char client_ip[INET_ADDRSTRLEN] = "";
      inet_ntop(AF_INET, &client_addr.sin_addr, client_ip, INET_ADDRSTRLEN);
      printf("Server: new client accepted. client ip and port: %s:%d\n", client_ip,
ntohs(client_addr.sin_port));


      // store new client connection into array
      client_sockets.push_back(new_socket);
      client_ips.push_back(client_ip);
      client_ports.push_back(ntohs(client_addr.sin_port));

      printf("current connected clients amount is %d \n", int(client_sockets.size()) );

      cout << "Do you have enought clients? (please input '1' for yes, '0' for no):" ;
      cin >> in_client_enough;
      if (in_client_enough == 0){
         cout << "OK. Please continute opening one or multiple new terminal windows to run
./client\n" << endl;
      }else if (in_client_enough != 1){
         cout << "Unrecognized input has been considered as 0. You can create one more client.\n" <<
endl;
         in_client_enough = 0;
```

```
        }
    }


    printf("\nClients creation finished! There are totally %d connected clients.\n",
int(client_sockets.size()) );
    printf("Asking all clients to report their local clock value ... \n\n\n");


    for (int i = 0; i < client_sockets.size(); i++){
        // sending a message to client
        const char *msg = "Hello from server, please tell me your local clock value.";
        send(client_sockets[i] , msg , strlen(msg) , 0 );
        printf("Server: sent to client(%s:%d): '%s'\n", client_ips[i].c_str(), client_ports[i], msg);


        // receiving
        while(recv(client_sockets[i], recv_buf, sizeof(recv_buf), 0) > 0 ){
            printf("Server: recv from client(%s:%d): '%s' \n", client_ips[i].c_str(), client_ports[i], recv_buf);

            // convert char array to string
            string recv_msg = string(recv_buf);

            if (recv_msg.find("Hello from client, my local clock value is") != string::npos){
                string substr_after_last_space;
                vector<string> split_str = split(recv_msg, " ");
                substr_after_last_space = split_str[ split_str.size() - 1 ];

                cout << "Server: received client local clock (string) is " << substr_after_last_space << endl;
                float substr_after_last_space_f = stof(substr_after_last_space);
                cout << "Server: received client local clock (float) is " << substr_after_last_space_f << endl;

                clients_local_clocks.push_back(substr_after_last_space_f);
            }

            memset(recv_buf, '\0', strlen(recv_buf));
            break;
        }
    }

    printf("\n\n");

    // average clock values
    float all_clock_sum = server_local_clock;
    for (int i = 0; i < clients_local_clocks.size(); i++){
```

```
        all_clock_sum += clients_local_clocks[i];
    }
    float avg_clock = all_clock_sum / (client_sockets.size() + 1);

    // tell clients how to adjust
    for (int i = 0; i < client_sockets.size(); i++){
        // prepare msg
        float offset = clients_local_clocks[i] - avg_clock;
        string operation;
        if (offset >= 0){
            operation = "minus";
        }else{
            operation = "add";
            offset = 0 - offset;
        }
        string msg_str = "From server, your clock adjustment offset is " + operation + " " +
to_string(offset);
        char msg_char_array[msg_str.length() + 1];
        strcpy(msg_char_array, msg_str.c_str());
        // sending a message to client
        send(client_sockets[i] , &msg_char_array , strlen(msg_char_array) , 0 );
        printf("Server: sent to client(%s:%d): '%s'\n", client_ips[i].c_str(), client_ports[i],
msg_char_array);

    }

    // adjust self
    server_local_clock += avg_clock - server_local_clock;
    printf("\n\nServer new local clock is %f \n\n", server_local_clock);


    printf("Server: server stopped. \n");
    close(server_socket_fd);
    return 0;
}
```

Output:


dyt@ubuntu: ~/Documents/621proj2/p1_berkeley_server_clients
dyt@ubuntu:~/Documents/621proj2/p1_berkeley_server_clients$ make
g++ server.cpp -o server -std=c++11
g++ client.cpp -o client -std=c++11
dyt@ubuntu:~/Documents/621proj2/p1_berkeley_server_clients$ ./server
Sever starts. Server pid is 9947
Server local clock is 4.000000
t1: step 1 run ./server
Server: server is listening ...
You can open one or multiple new terminal windows now to run ./client
You have connected 1 client(s) now.Server: new client accepted. client ip and port: 127.0.0.1:48722
current connected clients amount is 1
Do you have enought clients? (please input '1' for yes, '0' for no):0
OK. Please continute opening one or multiple new terminal windows to run ./client
t2: show 1st client connection info
You have connected 2 client(s) now.Server: new client accepted. client ip and port: 127.0.0.1:48726
current connected clients amount is 2
Do you have enought clients? (please input '1' for yes, '0' for no):1
t3: show 2nd client connection info
Clients creation finished! There are totally 2 connected clients.
Asking all clients to report their local clock value ...
Server: sent to client(127.0.0.1:48722): 'Hello from server, please tell me your local clock value.'
Server: recv from client(127.0.0.1:48722): 'Hello from client, my local clock value is 1.000000'
Server: received client local clock (string) is 1.000000
Server: received client local clock (float) is 1
Server: sent to client(127.0.0.1:48726): 'Hello from server, please tell me your local clock value.'
Server: recv from client(127.0.0.1:48726): 'Hello from client, my local clock value is 8.000000'
Server: received client local clock (string) is 8.000000
Server: received client local clock (float) is 8
t4: talk with clients
Server: sent to client(127.0.0.1:48722): 'From server, your clock adjustment offset is add 3.333333'
Server: sent to client(127.0.0.1:48726): 'From server, your clock adjustment offset is minus 3.666667'
t5: ask clients to adjust clocks
Server new local clock is 4.333333
Server: server stopped.
t6: adjust result
dyt@ubuntu:~/Documents/621proj2/p1_berkeley_server_clients$
dyt@ubuntu: ~/Documents/621proj2/p1_berkeley_server_clients
dyt@ubuntu:~/Documents/621proj2/p1_berkeley_server_clients$ ./client
Client starts. Client pid is 9951
Client local clock is 8.000000
t2: step 2 run ./client
Client: connected server(127.0.0.1:8080).
t4: talk with server
Client: read: 'Hello from server, please tell me your local clock value.'
Client: sent message: 'Hello from client, my local clock value is 8.000000'
Client: read: 'From server, your clock adjustment offset is minus 3.666667'
Client: received local clock adjustment offset (string) is minus 3.666667
Client: received local clock adjustment offset (float) is minus 3.66667
Client local clock is 4.333333
t6: adjust result
dyt@ubuntu:~/Documents/621proj2/p1_berkeley_server_clients$
dyt@ubuntu: ~/Documents/621proj2/p1_berkeley_server_clients
dyt@ubuntu:~/Documents/621proj2/p1_berkeley_server_clients$ ./client
Client starts. Client pid is 9950
Client local clock is 1.000000
t3: step 3 run ./client
Client: connected server(127.0.0.1:8080).
t4: talk with server
Client: read: 'Hello from server, please tell me your local clock value.'
Client: sent message: 'Hello from client, my local clock value is 1.000000'
Client: read: 'From server, your clock adjustment offset is add 3.333333'
Client: received local clock adjustment offset (string) is add 3.333333
Client: received local clock adjustment offset (float) is add 3.33333
Client local clock is 4.333333
t6: adjust result
dyt@ubuntu:~/Documents/621proj2/p1_berkeley_server_clients$

Assignment 5:

CODE:

TokenServer.java :

```
import java.io.*;
import java.net.*;
public class TokenServer
{
    public static void main(String agrs[])throws Exception
        {

            while(true)
            {
            Server sr=new Server();
            sr.recPort(8000);
            sr.recData();
            }
        }
}
class Server
{

    boolean hasToken=false;
    boolean sendData=false;
    int recport;

    void recPort(int recport)
    {
        this.recport=recport;
    }
    void recData()throws Exception
    {
        byte buff[]=new byte[256];
        DatagramSocket ds;
        DatagramPacket dp;
        String str;
```

```
        ds=new DatagramSocket(recport);
        dp=new DatagramPacket(buff,buff.length);
        ds.receive(dp);
        ds.close();

        str=new String(dp.getData(),0,dp.getLength());
        System.out.println("The message is "+str);
    }
}
```

TokenClient1.java

```
import java.io.*;
import java.net.*;
public class TokenClient1
{
    public static void main(String arg[]) throws Exception
        {
            InetAddress lclhost;
            BufferedReader br;
            String str="";
            TokenClient12 tkcl,tkser;
            boolean hasToken;
            boolean setSendData;

            while(true)
            {
            lclhost=InetAddress.getLocalHost();
            tkcl = new TokenClient12(lclhost);
            tkser = new TokenClient12(lclhost);
            //tkcl.setSendPort(9001);
            tkcl.setSendPort(9004);
            tkcl.setRecPort(8002);
            lclhost=InetAddress.getLocalHost();
            tkser.setSendPort(9000);
```

```
            if(tkcl.hasToken == true)
            {

System.out.println("Do you want to enter the Data –> YES/NO");
                br=new BufferedReader(new InputStreamReader(System.in));
                str=br.readLine();
                if(str.equalsIgnoreCase("yes"))
                {
                    System.out.println("ready to send");
                    tkser.setSendData = true;
                    tkser.sendData();
                    tkser.setSendData = false;
                }
                else if(str.equalsIgnoreCase("no"))
                {
                    System.out.println("i m in else");
                    //tkcl.hasToken=false;
                    tkcl.sendData();
                    tkcl.recData();
                System.out.println("i m leaving else");
                }
            }
            else
            {
            System.out.println("ENTERING RECEIVING MODE…");
                tkcl.recData();
            }
    }
}
}

class TokenClient12
{
    InetAddress lclhost;
    int sendport,recport;
    boolean hasToken = true;
    boolean setSendData = false;
```

```
    TokenClient12 tkcl,tkser;
    TokenClient12(InetAddress lclhost)
    {

        this.lclhost = lclhost;
    }

    void setSendPort(int sendport)
    {
        this.sendport = sendport;
    }
    void setRecPort(int recport)
    {
        this.recport = recport;
    }

void sendData() throws Exception
        {
        BufferedReader br;
        String str=”Token”;
        DatagramSocket ds;
        DatagramPacket dp;



        if(setSendData == true)
        {
            System.out.println(“sending “);
            System.out.println(“Enter the Data”);
            br=new BufferedReader(new InputStreamReader(System.in));
            str = “ClientOne…..” + br.readLine();
            System.out.println(“now sending”);

        }
            ds = new DatagramSocket(sendport);
            dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);
            ds.send(dp);
            ds.close();
```

```
        setSendData = false;
        hasToken = false;
    }

    void recData()throws Exception
    {
        String msgstr;
        byte buffer[] = new byte[256];
        DatagramSocket ds;
        DatagramPacket dp;
        ds = new DatagramSocket(recport);
        dp = new DatagramPacket(buffer,buffer.length);
        ds.receive(dp);
        ds.close();
        msgstr = new String(dp.getData(),0,dp.getLength());
        System.out.println("The data is "+msgstr);

        if(msgstr.equals("Token"))
        {
            hasToken = true;
        }
    }

}
```

TokenClient2.java

```
import java.io.*;
import java.net.*;
public class TokenClient2
{
    static boolean setSendData ;
    static boolean hasToken ;
    public static void main(String arg[]) throws Exception
        {
        InetAddress lclhost;
```

```
BufferedReader br;
String str1;
TokenClient21 tkcl;
TokenClient21 ser;
while(true)
{
lclhost=InetAddress.getLocalHost();
tkcl = new TokenClient21(lclhost);
tkcl.setRecPort(8004);
tkcl.setSendPort(9002);
lclhost=InetAddress.getLocalHost();
ser = new TokenClient21(lclhost);
ser.setSendPort(9000);
System.out.println("entering if");
if(hasToken == true)
{

System.out.println("Do you want to enter the Data –> YES/NO");
    br=new BufferedReader(new InputStreamReader(System.in));
    str1=br.readLine();
    if(str1.equalsIgnoreCase("yes"))
    {
        System.out.println("ignorecase");
        ser.setSendData = true;
        ser.sendData();
        }
    else if(str1.equalsIgnoreCase("no"))
    {
        tkcl.sendData();
        hasToken=false;
    }
}
else
{
System.out.println("entering recieving mode");
    tkcl.recData();
    hasToken=true;
```

```
            }
        }
    }
}
class TokenClient21
{
    InetAddress lclhost;
    int sendport,recport;
    boolean setSendData = false;
    boolean hasToken = false;
    TokenClient21 tkcl;
    TokenClient21 ser;

    TokenClient21(InetAddress lclhost)
    {

        this.lclhost = lclhost;
    }

    void setSendPort(int sendport)
    {
        this.sendport = sendport;
    }
    void setRecPort(int recport)
    {
        this.recport = recport;
    }
    void sendData() throws Exception
    {
        System.out.println("case");
        BufferedReader br;
        String str="Token";
        DatagramSocket ds;
        DatagramPacket dp;

        if(setSendData == true)
        {
```

```
            System.out.println("Enter the Data");
            br=new BufferedReader(new InputStreamReader(System.in));
            str = "ClientTwo….." + br.readLine();
        }
            ds = new DatagramSocket(sendport);
            dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);
            ds.send(dp);
            ds.close();
            System.out.println("Data Sent");
            setSendData = false;
            hasToken = false;

    }



void recData()throws Exception
    {
        String msgstr;
        byte buffer[] = new byte[256];
        DatagramSocket ds;
        DatagramPacket dp;
ds = new DatagramSocket(recport);
        //ds = new DatagramSocket(4000);
        dp = new DatagramPacket(buffer,buffer.length);
        ds.receive(dp);
        ds.close();
msgstr = new String(dp.getData(),0,dp.getLength());
        System.out.println("The data is "+msgstr);
        if(msgstr.equals("Token"))
            {
                hasToken = true;
            }
    }

}
```

OUTPUT :

TokerServer.java

```
>java TokenServer
The message is ClientOne…..hello
The message is ClientTwo…..hii
```

TokelClient1.java

```
>java TokenC
Client1
Do you want to enter the Data –> YES/NO
yes
ready to send
sending
Enter the Data
hello
now sending
Do you want to enter the Data –> YES/NO
no
i m in else
```

TokenClient2.java

```
>java TokenClient2
entering if
entering recieving mode
The data is Token
entering if
Do you want to enter the Data –> YES/NO
yes
ignorecase
case
Enter the Data
hii
Data Sent
entering if
Do you want to enter the Data –> YES/NO
```

Assignment 6:

CODE:

Bully.java:

```
import java.io.InputStream;
import java.io.PrintStream;
import java.util.Scanner;

public class Bully {
   static boolean[] state = new boolean[5];
   int coordinator;

   public static void up(int up) {
      if (state[up - 1]) {
         System.out.println("Process " + up + " is already up");
      } else {
         int i;
         Bully.state[up - 1] = true;
         System.out.println("Process " + up + " held election");
         for (i = up; i < 5; ++i) {
            System.out.println("Election message sent from process " + up + " to process " + (i + 1));
         }
         for (i = up + 1; i <= 5; ++i) {
            if (!state[i - 1]) continue;
            System.out.println("Alive message send from process " + i + " to process " + up);
            break;
         }
      }
   }

   public static void down(int down) {
      if (!state[down - 1]) {
         System.out.println("Process " + down + " is already dowm.");
      } else {
         Bully.state[down - 1] = false;
      }
   }

   public static void mess(int mess) {
      if (state[mess - 1]) {
         if (state[4]) {
            System.out.println("0K");
         } else if (!state[4]) {
            int i;
            System.out.println("Process " + mess + " election");
            for (i = mess; i < 5; ++i) {
```

```
            System.out.println("Election send from process " + mess + " to process " + (i + 1));
        }
        for (i = 5; i >= mess; --i) {
            if (!state[i - 1]) continue;
            System.out.println("Coordinator message send from process " + i + " to all");
            break;
        }
    }
  } else {
    System.out.println("Process " + mess + " is down");
  }
}

public static void main(String[] args) {
  int choice;
  Scanner sc = new Scanner(System.in);
  for (int i = 0; i < 5; ++i) {
    Bully.state[i] = true;
  }
  System.out.println("5 active process are:");
  System.out.println("Process up  = p1 p2 p3 p4 p5");
  System.out.println("Process 5 is coordinator");
  do {
    System.out.println(".........");
    System.out.println("1) Up a process.");
    System.out.println("2) Down a process");
    System.out.println("3) Send a message");
    System.out.println("4) Exit");
    choice = sc.nextInt();
    switch (choice) {
        case 1: {
            System.out.println("Bring proces up");
            int up = sc.nextInt();
            if (up == 5) {
                System.out.println("Process 5 is co-ordinator");
                Bully.state[4] = true;
                break;
            }
            Bully.up(up);
            break;
        }
        case 2: {
            System.out.println("Bring down any process.");
            int down = sc.nextInt();
            Bully.down(down);
            break;
        }
        case 3: {
```

```
                System.out.println("Which process will send message");
                int mess = sc.nextInt();
                Bully.mess(mess);
            }
        }
    } while (choice != 4);
    sc.close();

  }
}
```

Output :

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\cl9\Assignment 4>javac Bully.java

C:\Users\USER\Desktop\cl9\Assignment 4>java Bully
5 active process are:
Process up  = p1 p2 p3 p4 p5
Process 5 is coordinator
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
2
Bring down any process.
3
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
2
Bring down any process.
5
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
1
Bring proces up
4
Process 4 is already up
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
1
Bring proces up
5
Process 5 is co-ordinator
```

```
C:\Windows\System32\cmd.exe
3) Send a message
4) Exit
1
Bring proces up
1
Process 1 is already up
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
3
Which process will send message
3
Process 3 is down
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
1
Bring proces up
5
Process 5 is co-ordinator
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
3
Which process will send message
5
OK
........
1) Up a process.
2) Down a process
3) Send a message
4) Exit
4

C:\Users\USER\Desktop\cl9\Assignment 4>Microsoft Windows [Version 10.0.19041.985]
'Microsoft' is not recognized as an internal or external command,
operable program or batch file.
```

Ring.java:

```
import java.util.Scanner;

public class Ring {

	public static void main(String[] args) {

		// TODO Auto-generated method stub

		int temp, i, j;
		char str[] = new char[10];
		Rr proc[] = new Rr[10];

		for (i = 0; i < proc.length; i++){
			proc[i] = new Rr();
		}


		Scanner in = new Scanner(System.in);
		System.out.println("Enter the number of process : ");
		int num = in.nextInt();

		for (i = 0; i < num; i++) {
			proc[i].index = i;
			System.out.println("Enter the id of process : ");
			proc[i].id = in.nextInt();
			proc[i].state = "active";
			proc[i].f = 0;
		}


		for (i = 0; i < num - 1; i++) {
			for (j = 0; j < num - 1; j++) {
				if (proc[j].id > proc[j + 1].id) {
					temp = proc[j].id;
					proc[j].id = proc[j + 1].id;
					proc[j + 1].id = temp;
				}
			}
		}


		for (i = 0; i < num; i++) {
			System.out.print("  [" + i + "]" + " " + proc[i].id);
		}
```

```
int init;
int ch;
int temp1;
int temp2;
int ch1;
int arr[] = new int[10];

proc[num - 1].state = "inactive";

System.out.println("\n process " + proc[num - 1].id + "select as co-ordinator");

while (true) {
	System.out.println("\n 1.election 2.quit ");
	ch = in.nextInt();

	for (i = 0; i < num; i++) {
		proc[i].f = 0;
	}

	switch (ch) {
	case 1:
		System.out.println("\n Enter the Process number who initialsied
election : ");
		init = in.nextInt();
		temp2 = init;
		temp1 = init + 1;

		i = 0;

		while (temp2 != temp1) {
			if ("active".equals(proc[temp1].state) && proc[temp1].f ==
0) {

				System.out.println("\nProcess " + proc[init].id + "
send message to " + proc[temp1].id);
				proc[temp1].f = 1;
				init = temp1;
				arr[i] = proc[temp1].id;
				i++;
			}
			if (temp1 == num) {
				temp1 = 0;
			} else {
				temp1++;}
		}

		System.out.println("\nProcess " + proc[init].id + " send message to "
+ proc[temp1].id);
```

```
                                        arr[i] = proc[temp1].id;
                                        i++;
                                        int max = -1;


                                        for (j = 0; j < i; j++) {
                                                if (max < arr[j]) {
                                                        max = arr[j];
                                                }
                                        }

                                        System.out.println("\n process " + max + "select as co-ordinator");


                                        for (i = 0; i < num; i++) {

                                                if (proc[i].id == max) {
                                                        proc[i].state = "inactive";
                                                }
                                        }
                                        break;
                                case 2:
        System.out.println("Program terminated ...");
        return ;
                                default:
                                        System.out.println("\n invalid response \n");
                                        break;
                                }


                        }
                }

}

class Rr {

        public int index;   // to store the index of process
        public int id;      // to store id/name of process
        public int f;
        String state;       // indiactes whether active or inactive state of node
}
```

Output :

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER\Desktop\cl9\Assignment 4>javac Ring.java

C:\Users\USER\Desktop\cl9\Assignment 4>java Ring
Enter the number of process :
5
Enter the id of process :
1
Enter the id of process :
2
Enter the id of process :
3
Enter the id of process :
4
Enter the id of process :
5
  [0] 1  [1] 2  [2] 3  [3] 4  [4] 5
 process 5select as co-ordinator

 1.election 2.quit
1

 Enter the Process number who initialsied election :
3

Process 4 send message to 1

Process 1 send message to 2

Process 2 send message to 3

Process 3 send message to 4

 process 4select as co-ordinator

 1.election 2.quit
1

 Enter the Process number who initialsied election :
3

Process 4 send message to 1
```

```
C:\Windows\System32\cmd.exe
3

Process 4 send message to 1

Process 1 send message to 2

Process 2 send message to 3

Process 3 send message to 4

 process 4select as co-ordinator

 1.election 2.quit
1

 Enter the Process number who initialsied election :
1

Process 2 send message to 3

Process 3 send message to 1

Process 1 send message to 2

 process 3select as co-ordinator

 1.election 2.quit
2
Program terminated ...

C:\Users\USER\Desktop\cl9\Assignment 4>
```

Assignment 7: