

**ABES Institute of Technology,**

**Ghaziabad**

**COLLEGE CODE – 290**

# Lab File

|  |  |
| --- | --- |
| **NAME** | SANDEEP KUMAR SHUKLA |
| **BRANCH** | CSE |
| **UNIVERSITY ROLL NO.** | 1729010140 |
| **SESSION** | 2020-21 |
| **NAME OF LAB** | Distributed System Lab (RCS 751) |

**Aim :-**  *Write a program to simulate the functioning of*

*Lamport’s Logical clock in ‘C’.*

**Code :-**

#include <stdio.h>

int max1(int a, int b)

{

if (a > b)

return a;

else

return b;

}

int main()

{

int i, j, k, p1[20], p2[20], e1, e2, dep[20][20];

printf("Enter the events : ");

scanf("%d %d", &e1, &e2);

for (i = 0; i < e1; i++)

p1[i] = i + 1;

for (i = 0; i < e2; i++)

p2[i] = i + 1;

printf("Enter the dependency matrix:\n");

printf("\t Enter 1 if e1->e2 \n\t enter -1, if e2->e1 \n\t else enter 0 \n\n");

for (i = 0; i < e2; i++)

printf("\te2%d", i + 1);

for (i = 0; i < e1; i++)

{

printf("\n e1%d \t", i + 1);

for (j = 0; j < e2; j++)

scanf("%d", &dep[i][j]);

}

for (i = 0; i < e1; i++)

{

for (j = 0; j < e2; j++)

{

if (dep[i][j] == 1)

{

p2[j] = max1(p2[j], p1[i] + 1);

for (k = j; k < e2; k++)

p2[k + 1] = p2[k] + 1;

}

if (dep[i][j] == -1)

{

p1[i] = max1(p1[i], p2[j] + 1);

for (k = i; k < e1; k++)

p2[k + 1] = p1[k] + 1;

}

}

}

printf("P1 : ");

for (i = 0; i < e1; i++)

{

printf("%d", p1[i]);

}

printf("\n P2 : ");

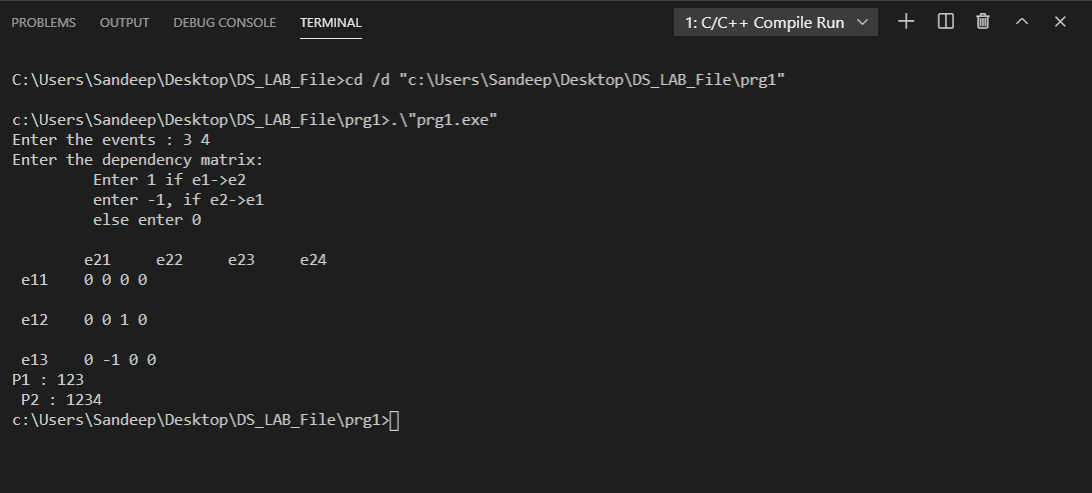
for (j = 0; j < e2; j++)

printf("%d", p2[j]);

return 0;

}

**Output :-**

****

**Aim :-** *Write a program to simulate the Distributed Mutual Exclusion in ‘C’.*

**Code :-**

#include <stdio.h>

#include <conio.h>

#include <dos.h>

#include <time.h>

void main()

{

int cs = 0, pro = 0;

double run = 5;

char key = 'a';

time\_t t1, t2;

printf("Press a key(except q) to enter a process into critical section.");

printf(" \nPress q at any time to exit.");

t1 = time(NULL) - 5;

while (key != 'q')

{

while (!kbhit())

if (cs != 0)

{

t2 = time(NULL);

if (t2 - t1 > run)

{

printf("Process%d ", pro - 1);

printf(" exits critical section.\n");

cs = 0;

}

}

key = getch();

if (key != 'q')

{

if (cs != 0)

{

printf("Error: Another process is currently executing critical section Please wait till its execution is over.\n");

}

else

{

printf("Process %d ", pro);

printf(" entered critical section\n");

cs = 1;

pro++;

t1 = time(NULL);

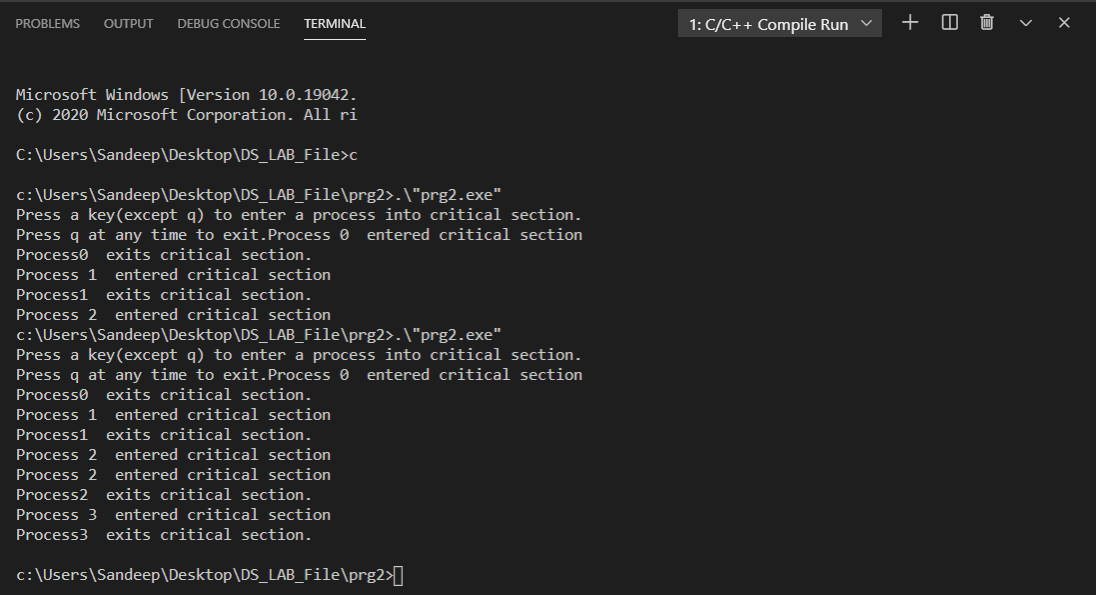
}

}

}

}

**Output :-**

****

**Aim :-** *Write a Write a program to implement a distributed Chat Server using*

*TCP sockets in ‘C’.*

**Code :-**

**TCP SERVER**

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

// Function designed for chat between client and server.

void func(int sockfd)

{

char buff[MAX];

int n;

// infinite loop for chat

for (;;)

{

bzero(buff, MAX);

// read the message from client and copy it in buffer

read(sockfd, buff, sizeof(buff));

// print buffer which contains the client contents

printf("From client: %s\t To client : ", buff);

bzero(buff, MAX);

n = 0;

// copy server message in the buffer

while ((buff[n++] = getchar()) != '\n')

;

// and send that buffer to client

write(sockfd, buff, sizeof(buff));

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0)

{

printf("Server Exit...\n");

break;

}

}

}

// Driver function

int main()

{

int sockfd, connfd, len;

struct sockaddr\_in servaddr, cli;

// socket create and verification

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1)

{

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(sockfd, (SA \*)&servaddr, sizeof(servaddr))) != 0)

{

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

// Now server is ready to listen and verification

if ((listen(sockfd, 5)) != 0)

{

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

// Accept the data packet from client and verification

connfd = accept(sockfd, (SA \*)&cli, &len);

if (connfd < 0)

{

printf("server acccept failed...\n");

exit(0);

}

else

printf("server acccept the client...\n");

// Function for chatting between client and server

func(connfd);

// After chatting close the socket

close(sockfd);

} sfdffdf

**TCP CLIENT**

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int n;

for (;;)

{

bzero(buff, sizeof(buff));

printf("Enter the string : ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

write(sockfd, buff, sizeof(buff));

bzero(buff, sizeof(buff));

read(sockfd, buff, sizeof(buff));

printf("From Server : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0)

{

printf("Client Exit...\n");

break;

}

}

}

int main()

{

int sockfd, connfd;

struct sockaddr\_in servaddr, cli;

// socket create and varification

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1)

{

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

servaddr.sin\_port = htons(PORT);

// connect the client socket to server socket

if (connect(sockfd, (SA \*)&servaddr, sizeof(servaddr)) != 0)

{

printf("connection with the server failed...\n");

exit(0);

}

else

printf("connected to the server..\n");

// function for chat

func(sockfd);

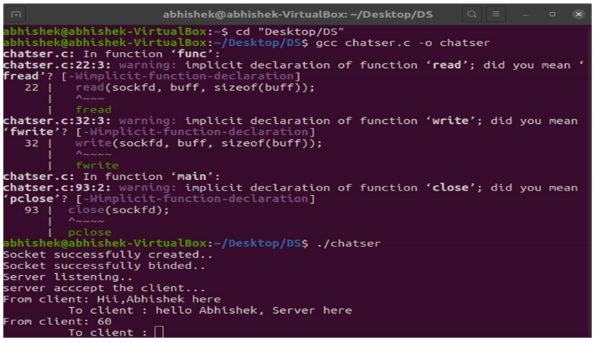
// close the socket

close(sockfd);

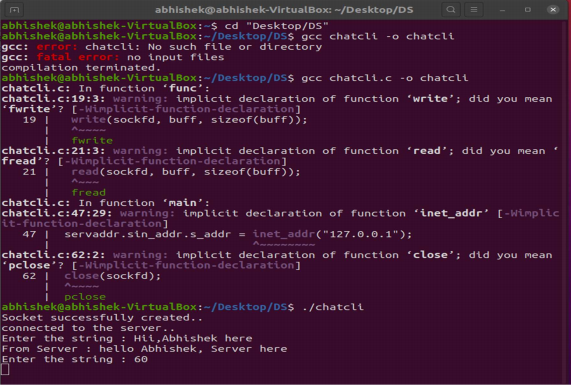
}

**Output :-**

**TCP SERVER**

****

**TCP CLIENT**

****

**Aim :-** *Write a program on RPC mechanism for a file transfer across a*

*network in ‘C’.*

**Code :-**

**CLIENT SIDE**

#include "transfer.h"

#include <time.h>

void

transfer\_1(char \*host, char \*filetotransf)

{

CLIENT \*clnt;

int \*result\_1;

file transf\_1\_arg;

FILE \*ofile;

long long int total = 0;

clnt = clnt\_create(host, TRANSFER, TRANSFER\_1, "tcp");

if (clnt == NULL)

{

clnt\_pcreateerror(host);

exit(1);

}

ofile = fopen(filetotransf, "rb");

if (ofile == NULL)

{

printf("File not found.\n");

exit(1);

}

printf("Sending file %s.\n", filetotransf);

strcpy(transf\_1\_arg.name, filetotransf);

clock\_t begin = clock();

while (1)

{

transf\_1\_arg.nbytes = fread(transf\_1\_arg.data, 1, MAXLEN, ofile);

total += transf\_1\_arg.nbytes;

//printf("\r%lld bytes of %s sent to server.", total, transf\_1\_arg.name);

result\_1 = transf\_1(&transf\_1\_arg, clnt);

if (result\_1 == (int \*)NULL)

{

clnt\_perror(clnt, "call failed");

}

if (transf\_1\_arg.nbytes < MAXLEN)

{

printf("\nUpload finished.\n");

break;

}

}

clock\_t end = clock();

double upload\_time = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("Upload time: %lf\n", upload\_time);

clnt\_destroy(clnt);

fclose(ofile);

}

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

{

char \*host;

char \*filetotransf;

if (argc < 3)

{

printf("usage: %s <server\_host> <file>\n", argv[0]);

exit(1);

}

host = argv[1];

filetotransf = argv[2];

transfer\_1(host, filetotransf);

exit(0);

}

**SERVER SIDE**

#include "transfer.h"

char opened\_file[MAXLEN];

FILE \*ofile;

long long int total = 0;

int \*transf\_1\_svc(file \*argp, struct svc\_req \*rqstp)

{

static int result;

static char tempName[MAXLEN];

strcpy(tempName, "uploaded\_");

strcat(tempName, argp->name);

strcpy(argp->name, tempName);

total += argp->nbytes;

if (strcmp(opened\_file, "") == 0 && ofile == NULL)

{

printf("Receiving new file %s.\n", argp->name);

strcpy(opened\_file, argp->name);

ofile = fopen(argp->name, "ab+");

}

if (strcmp(opened\_file, argp->name) == 0)

{

//printf("\r%lld bytes of file %s were received.", total, argp->name);

fflush(stdout);

fwrite(argp->data, 1, argp->nbytes, ofile);

if (argp->nbytes < MAXLEN)

{

printf("\nFinished receiving %s.\n", argp->name);

total = 0;

fclose(ofile);

ofile = NULL;

strcpy(opened\_file, "");

}

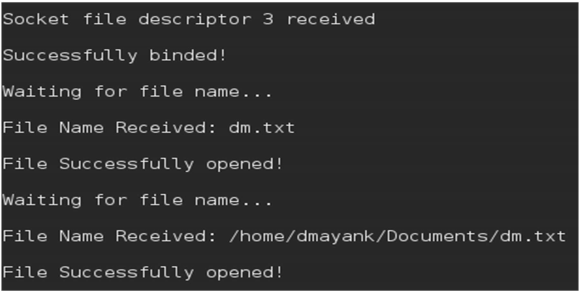
}

return &result;

}

**Output :-**

**SERVER SIDE**

****

**CLIENT SIDE**

****

**Aim :-** *Write a program to implement Java RMI mechanism for accessing*

*methods of remote systems.*

**Code :-**

**AddClient.java**

import java.rmi.\*;

public class AddClient

{

public static void main(String args[])

{

try

{

String addServerURL="rmi://"+ args[0] + "/AddServer";

AddServerIntf addServerIntf =

(AddServerIntf)Naming.lookup(addServerURL);

System.out.println("the first no is:" + args[1]);

double d1=Double.valueOf(args[1]).doubleValue();

System.out.println("the second no is:" + args[2]);

double d2=Double.valueOf(args[2]).doubleValue();

System.out.println("Sum = " + addServerIntf.add(d1,d2));

}

catch(Exception e)

{

System.out.println("Exception:" +e);

}

}

}

**AddServer.java**

import java.net.\*;

import java.rmi.\*;

public class AddServer

{

public static void main(String args[]){

try

{

AddServerImpl addServerImpl = new AddServerImpl();

Naming.rebind("AddServer", addServerImpl);

}

catch(Exception e){

System.out.println("Exception:" +e);

}

}

}AddServerImpl.java

import java.rmi.\*;

import java.rmi.server.\*;

public class AddServerImpl extends UnicastRemoteObject implements

AddServerIntf

{

public AddServerImpl() throws RemoteException

{

}

public double add(double d1,double d2) throws RemoteException

{

return d1+d2;

}

}AddServerIntf.java

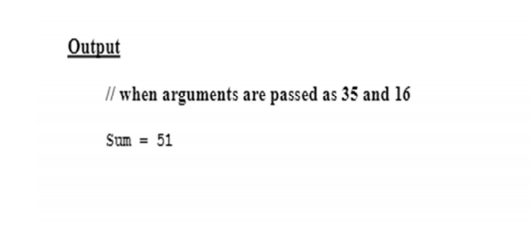
import java.rmi.\*;

public interface AddServerIntf extends Remote {

double add(double d1, double d2) throws RemoteException;

}

**Output :-**

****

**Aim :-** *Write a Write a program to simulate Balanced Window Protocol in ‘C’.*

**Code :-**

#include <stdio.h>

int main()

{

int w, i, f, frames[50];

printf("Enter window size: ");

scanf("%d", &w);

printf("\nEnter number of frames to transmit: ");

scanf("%d", &f);

printf("\nEnter %d frames: ", f);

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

scanf("%d", &frames[i]);

printf("\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n");

printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n", w);

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

{

if (i % w == 0)

{

printf("%d\n", frames[i]);

printf("Acknowledgement of above frames sent is received by sender\n\n");

}

else

printf("%d ", frames[i]);

}

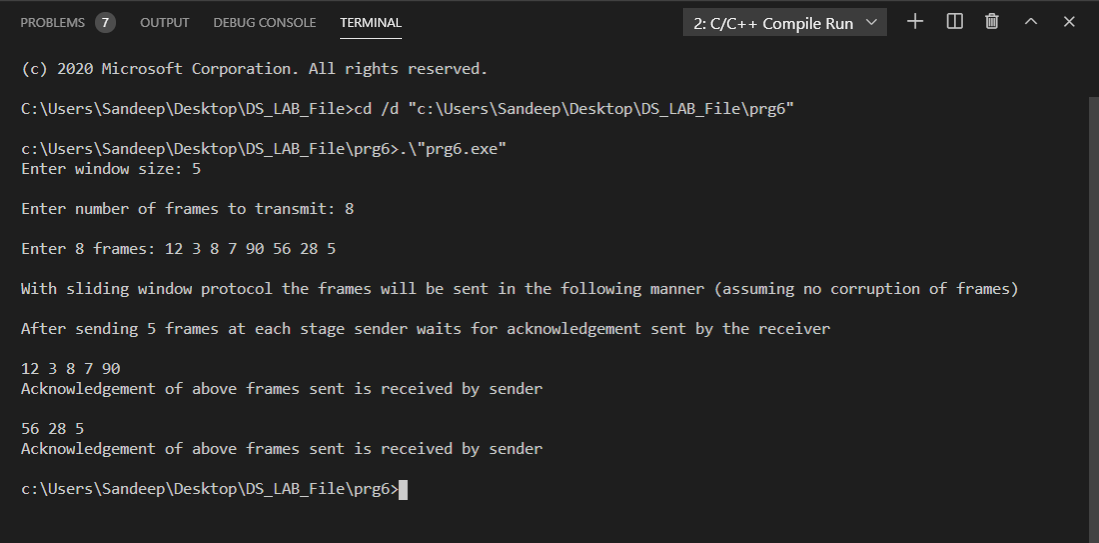
if (f % w != 0)

printf("\nAcknowledgement of above frames sent is received by sender\n");

return 0;

}

**Output :-**

****

**Aim :-** *Write a Write a program to implement COBRA mechanism using ‘C++’*

*program at one end and Java program on the other.*

**Code :-**

**Creating the Server**

#include <iostream.h>

#include "OB/CORBA.h"

#include <OB/Cosnaming.h>

#include "crypt.h"

#include "cryptimpl.h"

using namespace std;

int main(int argc, char \*\*argv)

{

// Declare ORB and servant object

CORBA::ORB\_var orb;

CryptographicImpl \*CrypImpl = NULL;

try

{

// Initialize the ORB.

orb = CORBA::ORB\_init(argc, argv);

// Get a reference to the root POA

CORBA::Object\_var rootPOAObj = orb -

> resolve\_initial\_references("RootPOA");

// Narrow it to the correct type

PortableServer::POA\_var rootPOA =

PortableServer::POA::\_narrow(rootPOAObj.in());

// Create POA policies

CORBA::PolicyList policies;

policies.length(1);

policies[0] = rootPOA->create\_thread\_policy(PortableServer::SINGLE\_THREAD\_MODEL);

// Get the POA manager object

PortableServer::POAManager\_var manager = rootPOA->the\_POAManager();

// Create a new POA with specified policies

PortableServer::POA\_var myPOA = rootPOA->create\_POA("myPOA",

manager, policies);

// Free policies

CORBA::ULong len = policies.length();

for (CORBA::ULong i = 0; i < len; i++)

policies[i]->destroy();

// Get a reference to the Naming Service root\_context

CORBA::Object\_var rootContextObj = orb -

> resolve\_initial\_references("NameService");

// Narrow to the correct type

CosNaming::NamingContext\_var nc =

CosNaming::NamingContext::\_narrow(rootContextObj.in());

// Create a reference to the servant

CrypImpl = new CryptographicImpl(orb);

// Activate object

PortableServer::ObjectId\_var myObjID = myPOA->activate\_object(CrypImpl);

// Get a CORBA reference with the POA through the servant

CORBA::Object\_var o = myPOA->servant\_to\_reference(CrypImpl);

// The reference is converted to a character string

CORBA::String\_var s = orb->object\_to\_string(o);

cout << "The IOR of the object is: " << s.in() << endl;

CosNaming::Name name;

name.length(1);

name[0].id = (const char \*)"CryptographicService";

name[0].kind = (const char \*)"";

// Bind the object into the name service

nc->rebind(name, o);

// Activate the POA

manager->activate();

cout << "The server is ready. Awaiting for incoming requests..." << endl;

// Start the ORB

orb->run();

}

catch (const CORBA::Exception &e)

{

// Handles CORBA exceptions

cerr << e << endl;

}

// Decrement reference count

if (CrypImpl)

CrypImpl->\_remove\_ref();

// End CORBA

if (!CORBA::is\_nil(orb))

{

try

{

orb->destroy();

cout << "Ending CORBA..." << endl;

}

catch (const CORBA::Exception &e)

{

cout << "orb->destroy() failed:" << e << endl;

return 1;

}

}

return 0;

}

**Implementing the Client**

#include <iostream.h>

#include <string.h>

#include "OB/CORBA.h"

#include "OB/Cosnaming.h"

#include "crypt.h"

using namespace std;

int main(int argc, char \*\*argv)

{

// Declare ORB

CORBA::ORB\_var orb;

try

{

// Initialize the ORB

orb = CORBA::ORB\_init(argc, argv);

// Get a reference to the Naming Service

CORBA::Object\_var rootContextObj = orb -

> resolve\_initial\_references("NameService");

CosNaming::NamingContext\_var nc =

CosNaming::NamingContext::\_narrow(rootContextObj.in());

CosNaming::Name name;

name.length(1);

name[0].id = (const char \*)"CryptographicService";

name[0].kind = (const char \*)"";

// Invoke the root context to retrieve the object reference

CORBA::Object\_var managerObj = nc->resolve(name);

// Narrow the previous object to obtain the correct type

::CaesarAlgorithm\_var manager =

::CaesarAlgorithm::\_narrow(managerObj.in());

string info\_in, exit, dummy;

CORBA::String\_var info\_out;

::CaesarAlgorithm::charsequence\_var inseq;

unsigned long key, shift;

try

{

do

{

cout << "\nCryptographic service client" << endl;

cout << "----------------------------" << endl;

do

{ // Get the cryptographic key

if (cin.fail())

{

cin.clear();

cin >> dummy;

}

cout << "Enter encryption key: ";

cin >> key;

} while (cin.fail());

do

{ // Get the shift

if (cin.fail())

{

cin.clear();

cin >> dummy;

}

cout << "Enter a shift: ";

cin >> shift;

} while (cin.fail());

// Used for debug pourposes

//key = 9876453;

//shift = 938372;

getline(cin, dummy); // Get the text to encrypt

cout << "Enter a plain text to encrypt: ";

getline(cin, info\_in);

// Invoke first remote method

inseq = manager->encrypt(info\_in.c\_str(), key, shift);

cout << "------------------------------------------" << endl;

cout << "Encrypted text is: " << inseq->get\_buffer() << endl;

// Invoke second remote method

info\_out = manager->decrypt(inseq.in(), key, shift);

cout << "Decrypted text is: " << info\_out.in() << endl;

cout << "-------------------------------------------" << endl;

cout << "Exit? (y/n): ";

cin >> exit;

} while (exit != "y");

// Shutdown server message

manager->shutdown();

} // end of tyr2

catch (const std::exception &std\_e)

{

cerr << std\_e.what() << endl;

}

} //end of try1

catch (const CORBA::Exception &e)

{

// Handles CORBA exceptions

cerr << e << endl;

} // End CORBA

if (!CORBA::is\_nil(orb))

{

try

{

orb->destroy();

cout << "Ending CORBA..." << endl;

}

catch (const CORBA::Exception &e)

{

cout << "orb->destroy failed:" << e << endl;

return 1;

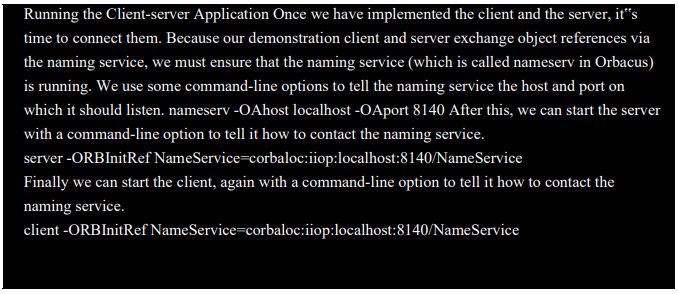
}

}

return 0;

} //end of main

**Output :-**

****