Date: 14th Jan 2022

Name: Rohan Khurana
Roll No: 1802910129

Distributed System Lab

(KCS751A)

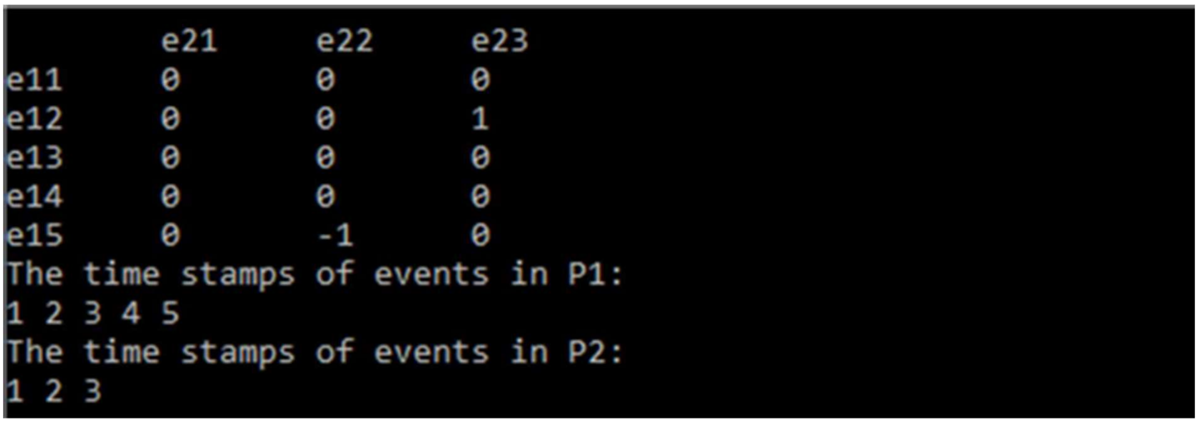
External Practical

Aim: Simulate the functioning of Lamport's logical Clock

Program:

```
#include <bits/stdc++.h>
using namespace std;
int max1(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}

void display(int e1, int e2, int p1[5], int p2[3])
{
    int i;
    cout << "\nThe time stamps of events in P1:\n";
    for (i = 0; i < e1; i++) {
        cout << p1[i] << " ";
    }
    cout << "\nThe time stamps of events in P2:\n";
    for (i = 0; i < e2; i++)
        cout << p2[i] << " ";
}
void lamportLogicalClock(int e1, int e2, int m[5][3])
{
    int i, j, k, p1[e1], p2[e2];
    for (i = 0; i < e1; i++)
        p1[i] = i + 1;
    for (i = 0; i < e2; i++)
        p2[i] = i + 1;
    for (i = 0; i < e2; i++)
        cout << "\te2" << i + 1;
    for (i = 0; i < e1; i++) {
        cout << "\ne1" << i + 1<<"\t";
        for (j = 0; j < e2; j++)
            cout << m[i][j] << "\t";
    }
    for (i = 0; i < e1; i++) {
        for (j = 0; j < e2; j++) {
            if (m[i][j] == 1) {
                p2[j] = max1(p2[j], p1[i] + 1);
                for (k = j + 1; k < e2; k++)
                    p2[k] = p2[k - 1] + 1;
            }
            if (m[i][j] == -1) {
                p1[i] = max1(p1[i], p2[j] + 1);
                for (k = i + 1; k < e1; k++)
                    p1[k] = p1[k - 1] + 1;
            }
        }
    }
    display(e1, e2, p1, p2);
}
int main()
{
    int e1 = 5, e2 = 3, m[5][3];
    m[0][0] = 0; m[0][1] = 0;
    m[0][2] = 0; m[1][0] = 0;
    m[1][1] = 0; m[1][2] = 1;
    m[2][0] = 0; m[2][1] = 0;
    m[2][2] = 0; m[3][0] = 0;
    m[3][1] = 0; m[3][2] = 0;
    m[4][0] = 0; m[4][1] = -1;
    m[4][2] = 0;
    lamportLogicalClock(e1, e2, m);
    return 0;
}
```

Output:



```
        e21     e22     e23
e11     0       0       0
e12     0       0       1
e13     0       0       0
e14     0       0       0
e15     0       -1      0
The time stamps of events in P1:
1 2 3 4 5
The time stamps of events in P2:
1 2 3
```

Date: 25th Sept 2021

Name: Rohan Khurana
Roll No: 1802910129

Distributed System Lab

LAB - 3

Aim: WAP to implement Vector Clock

Program:

```
#include<iostream>
#include<conio.h>
#define SIZE 10
using namespace std;

class node {
      public:
      int data[SIZE];
      node *next;
            node() {

                  for(int p=0; p<SIZE; p++) {
                        data[p] = 0;
                  }
                  next = NULL;
            }
            node(int v[], int n1) {

                  for(int s = 0; s < n1; s++) {
                        data[s] = v[s];
                  }
                  next = NULL;
            }
            friend class process;

}*start=NULL;



int main() {
      int n, events, sent, receive, sentE, recE, commLines = 0;
```

```
node *temp;
node *proc[SIZE];
cout<<"Enter no. of processes: ";
cin>>n;
int vector[n] = {0};

for(int i = 0; i < n; i++) {
      for(int v = 0; v < n; v++) {
            vector[v] = 0;
      }

      cout<<"Enter no. of events in process "<<i+1<<": ";
      cin>>events;
      for(int j = 1; j <= events; j++) {
            vector[i] = j;
            node *newnode = new node(vector,n);
            if(start == NULL) {
                  start = newnode;
                  temp = start;
            }
            else {
                  temp->next = newnode;
                  temp = temp->next;
            }
      }
      proc[i] = start;
      start = NULL;
}

cout<<"\nEnter the number of communication lines: ";
cin>>commLines;
node *tempS, *tempR;
for(int i = 0; i < commLines; i++) {
      cout<<"\nEnter the sending process: ";
      cin>>sent;
      cout<<"\nEnter the receiving process: ";
      cin>>receive;
      cout<<"\nEnter the sending event number: ";
      cin>>sentE;
      cout<<"\nEnter the receiving event number: ";
```

```
        cin>>recE;
        tempS = proc[sent - 1];
        tempR = proc[receive - 1];
        for(int j = 1; j < sentE; j++)
            tempS = tempS->next;
        for(int j = 1; j < recE; j++)
            tempR = tempR->next;
        for(int j = 0; j < n; j++) {
            tempR->data[j] = (tempR->data[j] < tempS->data[j]) ?
tempS->data[j] : tempR->data[j];
        }
    }


    cout<<"\nThe resulting vectors are:\n\n";
    for(int k = 0; k < n; k++) {
        cout<<"Process "<<k + 1<<": ";
        node *temp1 = proc[k];
        while(temp1) {
            cout<<"(";
            for(int f = 0; f < n - 1; f++)
                cout<<temp1->data[f]<<",";
            cout<<temp1->data[n-1];
            cout<<")";
            temp1 = temp1->next;
        }
        cout<<endl;
    }
    return 0;

}
```

Date: 03rd Dec 2021

Name: Rohan Khurana
Roll No: 1802910129

Distributed System Lab

LAB – 8

Aim: Implement 'Java RMI' mechanism for accessing methods of remote systems.

Programs:

1. **Remote Interface**

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

public interface Hello extends Remote {
   void printMsg() throws RemoteException;
}
```

2. **Implementation Class**

```
public class ImplExample implements Hello {

   public void printMsg() {

      System.out.println("This is an example RMI program");

   }

}
```

3. **Server Program**

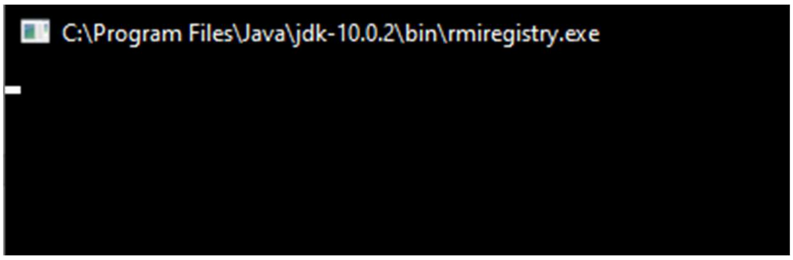
```
import java.rmi.registry.Registry;

import java.rmi.registry.LocateRegistry;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class Server extends ImplExample {

   public Server() {}

   public static void main(String args[]) {

      try {
         ImplExample obj = new ImplExample();

         Hello stub = (Hello) UnicastRemoteObject.exportObject(obj,
0);

         Registry registry = LocateRegistry.getRegistry();

         registry.bind("Hello", stub);

         System.err.println("Server ready");

      } catch (Exception e) {

         System.err.println("Server exception: " + e.toString());

         e.printStackTrace();

      }

   }

}
```

4. **Client Program**

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Client {
   private Client() {}
   public static void main(String[] args) {
      try {

         Registry registry = LocateRegistry.getRegistry(null);
         Hello stub = (Hello) registry.lookup("Hello");
         stub.printMsg();
      } catch (Exception e) {
         System.err.println("Client exception: " + e.toString());
         e.printStackTrace();
      }
   }
}
```

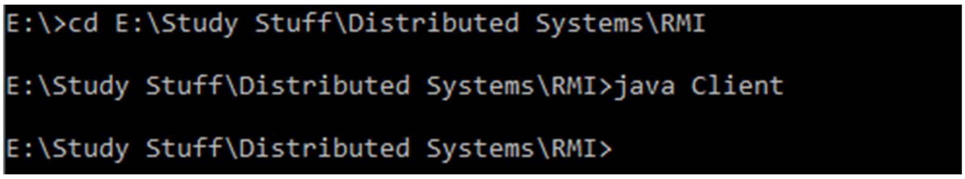
Output:

RMI Registry



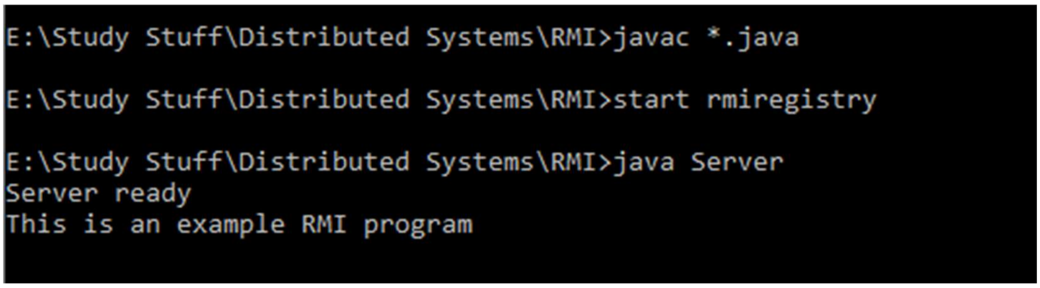
C:\Program Files\Java\jdk-10.0.2\bin\rmiregistry.exe

Client.exe



```
E:\>cd E:\Study Stuff\Distributed Systems\RMI

E:\Study Stuff\Distributed Systems\RMI>java Client

E:\Study Stuff\Distributed Systems\RMI>
```

Server.exe



```
E:\Study Stuff\Distributed Systems\RMI>javac *.java

E:\Study Stuff\Distributed Systems\RMI>start rmiregistry

E:\Study Stuff\Distributed Systems\RMI>java Server
Server ready
This is an example RMI program
```

Date: 03rd Dec 2021

Name: Rohan Khurana
Roll No: 1802910129

Distributed System Lab

LAB – 9

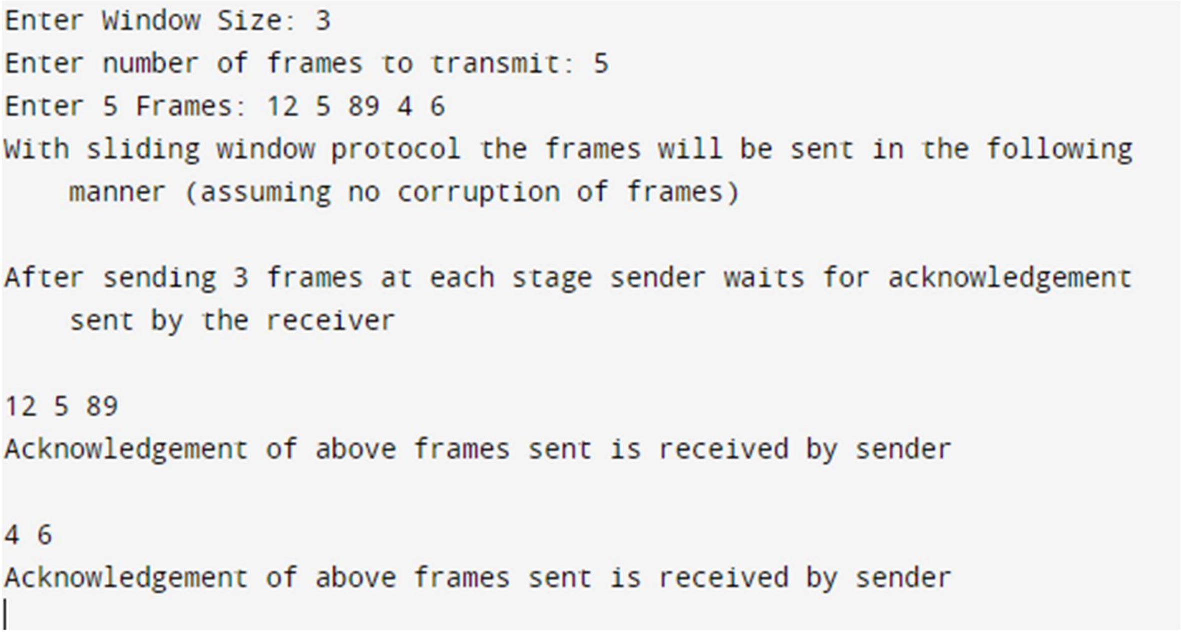
Aim: Simulate Balanced Sliding Window Protocol in ‘C’.

Program:

```
#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:



```
Enter Window Size: 3
Enter number of frames to transmit: 5
Enter 5 Frames: 12 5 89 4 6
With sliding window protocol the frames will be sent in the following
    manner (assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement
    sent by the receiver

12 5 89
Acknowledgement of above frames sent is received by sender

4 6
Acknowledgement of above frames sent is received by sender
```

Date: 10th Dec 2021

Name: Rohan Khurana
Roll No: 1802910129

Distributed System Lab

LAB – 10

Aim: Write a program to implement CORBA mechanism by using C++ program at one end and Java program on the other.

Program:

Server.cpp

```
#include <iostream>
#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;
}
```

Client.cpp

```
#include <iostream>
#include <string>
#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();
 } catch(const std::exception& std_e){
 cerr << std_e.what() << endl;
 }
 }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;
}
```