

DISTRIBUTED COMPUTING

(CACSC15)



PRACTICAL FILE

Submitted By:

Name: Ritesh Kumar

Roll No: 2019UCS2035

Branch: CSAI (Semester V)

INDEX

S.NO	TOPIC
1.	Program to implement Lamport's Logical Clock.
2.	Program to implement non-token based algorithm for Mutual Exclusion.
3.	Program to implement edge chasing distributed deadlock detection algorithm.
4.	Program to implement locking algorithm.
5.	Program to implement Remote Method Invocation.
6.	Program to implement Remote Procedure Call.
7.	Program to implement Chat Server.
8.	Program to implement termination detection.
9.	Program to implement RSA Algorithm.
10.	Program to implement Diffie Hellman Key Exchange Algorithm

Practical-1

Aim: Write a Program to implement Lamport's Logical Clock.

Code:

```
#include <bits/stdc++.h>
using namespace std;

int d = 1;

class process
{
    int n;
    vector<int> timestamps;

public:
    process(int n)
    {
        this->n = n;
        for (int i = 0; i < n; i += d)
            timestamps.push_back(i + 1);
    }

    int get_events()
    {
        return n;
    }

    int get_timestamp(int event_no)
    {
        return timestamps[event_no];
    }

    void adjust_timestamp(int Ci, int recieve_event)
    {
        int &Cj = timestamps[recieve_event];

        Cj = max(Cj, Ci + d);

        for (int i = recieve_event + 1; i < timestamps.size(); i++)
        {
            if (timestamps[i - 1] < timestamps[i])
                break;

            timestamps[i] = timestamps[i - 1] + d;
        }
    }

    void print_timestamps()
    {
        for (int i = 0; i < n; i++)
```

```

        cout << timestamps[i] << " ";
    cout << endl;
}
};

int main()
{
    int n;
    cout << "Enter the no. of processes: " << endl;
    cin >> n;

    vector<process> processes;
    for (int i = 0; i < n; i++)
    {
        int events;
        cout << "Enter the no. of events in process " << i + 1 << endl;
        cin >> events;

        process p(events);
        processes.push_back(p);
    }

    int m;
    cout << "Enter the no. of messages sent: " << endl;
    cin >> m;

    for (int i = 0; i < m; i++)
    {
        int send_process, send_event, recieve_process, recieve_event;
        cout << "Enter the process no. and event no. of sender for message " << i + 1 << endl;
        cin >> send_process >> send_event;

        cout << "Enter the process no. and event no. of reciever for message " << i + 1 << endl;
        cin >> recieve_process >> recieve_event;

        process sender = processes[send_process - 1];
        process &reciever = processes[recieve_process - 1];
        reciever.adjust_timestamp(sender.get_timestamp(send_event - 1), recieve_event - 1);
    }

    for (int i = 0; i < n; i++)
    {
        cout << "The timestamps of events in Process " << i + 1 << " are :" << endl;
        processes[i].print_timestamps();
    }

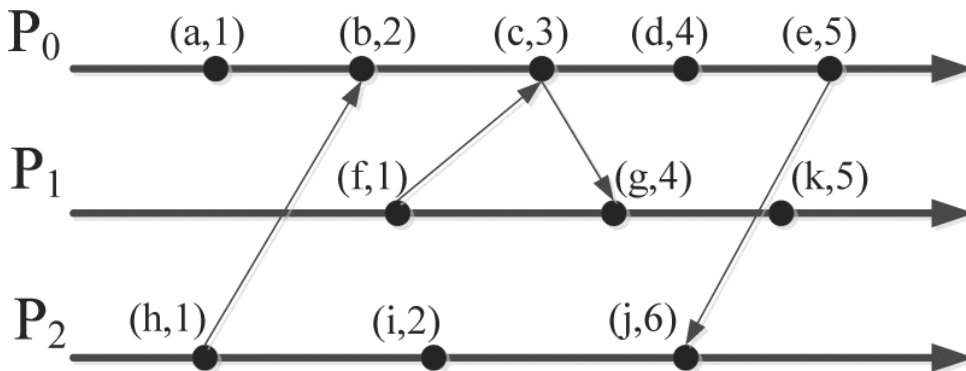
    return 0;
}

```

Output:

```
Enter the no. of events in process 2
4
Enter the no. of events in process 3
3
Enter the no. of events in process 4
3
Enter the no. of messages sent:
3
Enter the process no. and event no. of sender for message 1
1 2
Enter the process no. and event no. of reciever for message 1
2 3
Enter the process no. and event no. of sender for message 2
3 1
Enter the process no. and event no. of reciever for message 2
4 2
Enter the process no. and event no. of sender for message 3
2 1
Enter the process no. and event no. of reciever for message 3
4 1
The timestamps of events in Process 1 are :
1 2
The timestamps of events in Process 2 are :
1 2 3 4
The timestamps of events in Process 3 are :
1 2 3
The timestamps of events in Process 4 are :
2 3 4
```

This example is shown here:



Practical-2

Aim: Write a Program to implement non-token based algorithm for Mutual Exclusion.

Code:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    int process_id;
    int count = 0;
    bool flag = true;
    cout << "Lamport's Distributed Mutual Exclusion Algorithm: " << endl;
    cout << "Enter no of processes:" << endl;
    cin >> n;
    vector<int> waiting_queue;
    do
    {
        int choice;
        cout << "Enter the process which wants to execute Critical Section:" << endl;
        cin >> process_id;
        waiting_queue.push_back(process_id);
        cout << "Any other process wants to execute Critical Section?:" << endl;
        cout << "1. Yes" << endl;
        cout << "2. No" << endl;
        cin >> choice;
        if (choice == 2)
            flag = false;

    } while (flag);

    for (int j = 0; j < waiting_queue.size(); j++)
    {
        printf("\nCritical Section executing for the Process %d .....", waiting_queue[j]);
        printf("\nCritical Section is finished for the Process %d", waiting_queue[j]);
        printf("\nRelease Message has been sent by the Process %d\n", waiting_queue[j]);
    }

    return 0;
}
```

Output:

```

Lamport's Distributed Mutual Exclusion Algorithm:
Enter no of processes:
2
Enter the process which wants to execute Critical Section:
2
Any other process wants to execute Critical Section?:
1. Yes
2. No
1
Enter the process which wants to execute Critical Section:
1
Any other process wants to execute Critical Section?:
1. Yes
2. No
2

Critical Section executing for the Process 2 .....
Critical Section is finished for the Process 2
Release Message has been sent by the Process 2

Critical Section executing for the Process 1 .....
Critical Section is finished for the Process 1
Release Message has been sent by the Process 1

```

Practical-3

Aim: Write a Program to implement edge chasing distributed deadlock detection algorithm.

Code:

```

#include <bits/stdc++.h>
using namespace std;

bool deadlock(int start, int current, vector<vector<bool>> &depends,
              vector<bool> &visited, vector<int> &site_of_event)
{
    if (visited[current])
        return true;

    visited[current] = true;
    for (int i = 0; i < depends[current].size(); i++)
    {
        if (!depends[current][i])
            continue;
        if (i == current)
            return true;
        if (site_of_event[current] != site_of_event[i])
            cout << "Probe is sent: (" << start + 1 << ", " << current + 1 << ", "
                  << i + 1 << ")" << endl;
        return deadlock(start, i, depends, visited, site_of_event);
    }
    return false;
}

```

```

int main()
{
    int sites;
    vector<int> site_of_event;

    cout << "Enter number of sites: " << endl;
    cin >> sites;

    int total_no_of_events = 0;
    for (int i = 0; i < sites; i++)
    {
        int events;
        cout << "Enter number of events in site " << i + 1 << ": " << endl;
        cin >> events;

        for (int j = 0; j < events; j++)
            site_of_event.push_back(i);

        total_no_of_events += events;
    }

    cout << "So, we have " << sites << " sites and " << total_no_of_events << " events numbered " << endl;
    for (int i = 1; i <= total_no_of_events; i++)
        cout << i << " ";
    cout << endl;

    vector<vector<bool>> depends(total_no_of_events, vector<bool>(total_no_of_events, false));

    int m;
    cout << "Enter the no. of dependencies: " << endl;
    cin >> m;

    for (int i = 0; i < m; i++)
    {
        int a, b;
        cout << "Enter the Dependencies (If event 1 depends on event 2, enter 1 2):" << endl;
        cin >> a >> b;
        depends[a - 1][b - 1] = true;
    }

    cout << "Enter the Node to Start Probe: " << endl;
    int start;
    cin >> start;
    start--;

    vector<bool> visited(total_no_of_events, false);

    if (deadlock(start, start, depends, visited, site_of_event))
        cout << "A Deadlock exists" << endl;
    else
        cout << "No Deadlock doesn't exist" << endl;
    return 0;
}

```


Output:

```
Enter number of sites:
3
Enter number of events in site 1:
2
Enter number of events in site 2:
3
Enter number of events in site 3:
4
So, we have 3 sites and 9 events numbered
1 2 3 4 5 6 7 8 9
Enter the no. of dependencies:
2
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
3 1
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
4 5
Enter the Node to Start Probe:
3
Probe is sent: (3,3,1)
No Deadlock doesn't exist
```

Practical-4

Aim: Write a Program to implement locking algorithm.

Code:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int lock = 0;
    bool locked_by_T1 = false, locked_by_T2 = false;

    while (1)
    {
        int choice;

        if (!locked_by_T1)
        {
            cout << "Transaction T1 wants to Lock Data Object ?" << endl;
            cout << "1. Yes" << endl;
            cout << "2. No" << endl;
            cin >> choice;

            if (!lock && choice == 1)
            {
                lock = 1;
            }
        }
    }
}
```

```

        locked_by_T1 = true;
        cout << "T1 has been given the Lock for the Data Object" << endl;
    }
    else if (lock)
        cout << "\nData Object is Already Locked by " << (locked_by_T1 ? "T1" : "T2") << endl
            << endl;
}

if (!locked_by_T2)
{
    cout << "Transaction T2 wants to Lock Data Object ?" << endl;
    cout << "1. Yes" << endl;
    cout << "2. No" << endl;
    cin >> choice;

    if (!lock && choice == 1)
    {
        lock = 1;
        locked_by_T2 = true;
        cout << "T2 has been given the Lock for the Data Object" << endl;
    }
    else if (lock)
        cout << "\nData Object is Already Locked by " << (locked_by_T1 ? "T1" : "T2") << endl
            << endl;
}

if (lock)
{
    if (locked_by_T1)
    {
        cout << "Transaction T1 wants to Release the Lock on Data Object?" << endl;
        cout << "1. Yes" << endl;
        cout << "2. No" << endl;
        cin >> choice;
        if (choice == 1)
        {
            lock = 0, locked_by_T1 = false;
            cout << "The Lock on Data Object has been released by T1 !" << endl;
        }
    }
    else
    {
        cout << "Transaction T2 wants to Release the Lock on Data Object?" << endl;
        cout << "1. Yes" << endl;
        cout << "2. No" << endl;
        cin >> choice;
        if (choice == 1)
        {
            lock = 0, locked_by_T2 = false;
            cout << "The Lock on Data Object has been released by T2 !" << endl;
        }
    }
}
}
}

```

```
    return 0;
}
```

Output:

```
Transaction T1 wants to Lock Data Object ?
1. Yes
2. No
1
T1 has been given the Lock for the Data Object
Transaction T2 wants to Lock Data Object ?
1. Yes
2. No
1

Data Object is Already Locked by T1

Transaction T1 wants to Release the Lock on Data Object?
1. Yes
2. No
2
Transaction T2 wants to Lock Data Object ?
1. Yes
2. No
1

Data Object is Already Locked by T1

Transaction T1 wants to Release the Lock on Data Object?
1. Yes
2. No
2
Transaction T2 wants to Lock Data Object ?
1. Yes
2. No
2
```

Practical-5

Aim: Write a Program to implement Remote Method Invocation.

Code: We have implemented a Calculator using RMI. The implementation of this program consists of four Java files, namely CalculatorImpl.java, CalculatorServer.java, Calculator.java and CalculatorClient.java.

Calculator.java:

```
import java.rmi.*;

public interface Calculator extends Remote {
    public long add(long a, long b) throws RemoteException;

    public long sub(long a, long b) throws RemoteException;

    public long mul(long a, long b) throws RemoteException;
```

```
    public long div(long a, long b) throws RemoteException;
}
```

CalculatorImpl.java

```
public class CalculatorImpl extends java.rmi.server.UnicastRemoteObject implements Calculator {

    public CalculatorImpl() throws java.rmi.RemoteException {
        super();
    }

    public long add(long a, long b) {
        return a + b;
    }

    public long sub(long a, long b) {
        return a - b;
    }

    public long mul(long a, long b) {
        return a * b;
    }

    public long div(long a, long b) {
        return a / b;
    }

}
```

CalculatorServer.java

```
import java.rmi.Naming;

public class CalculatorServer {
    public CalculatorServer() {
        try {
            Calculator c = new CalculatorImpl();
            Naming.rebind("rmi://localhost:1099/CalculatorService", c);
        } catch (Exception e) {
            System.out.println("Trouble: " + e);
        }
    }

    public static void main(String args[]) {
        new CalculatorServer();
    }
}
```

CalculatorClient.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
```

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

public class CalculatorClient {

    public static void main(String[] args) {
        try {
            Calculator c = (Calculator) Naming.lookup("rmi://localhost/CalculatorService");
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter the first number, a = ");
            int a = sc.nextInt();
            System.out.print("Enter the second number, b = ");
            int b = sc.nextInt();
            sc.close();

            System.out.println();
            System.out.print("The sum of a and b = ");
            System.out.println(c.add(a, b));

            System.out.print("The difference of a and b = ");
            System.out.println(c.sub(a, b));

            System.out.print("The product of a and b = ");
            System.out.println(c.mul(a, b));

            System.out.print("The division of a and b = ");
            System.out.println(c.div(a, b));
        }

        catch (MalformedURLException murle) {
            System.out.println();
            System.out.println("MalformedURLException");
            System.out.println(murle);
        }

        catch (RemoteException re) {
            System.out.println();
            System.out.println("Remote Exception");
            System.out.println(re);
        }

        catch (NotBoundException nbe) {
            System.out.println();
            System.out.println("NotBoundException");
            System.out.println(nbe);
        }

        catch (java.lang.ArithmeticException ae) {
            System.out.println();
            System.out.println("java.lang.ArithmeticException");
            System.out.println(ae);
        }
    }
}
```

Output:

```
Enter the first number, a = 12
Enter the second number, b = 3

The sum of a and b = 15
The difference of a and b = 9
The product of a and b = 36
The division of a and b = 4
```

Practical-6

Aim: Write a Program to implement Remote Procedure Call.

Code:

Client –

```
#include "IDL.h"
#include <stdio.h>
float compute_6(char *host, float a, float b, char op)
{
    CLIENT *client_object;
    float *result;
    values oper_6_arg;

    oper_6_arg.num1 = a;
    oper_6_arg.num2 = b;
    oper_6_arg.operation = op;
    client_object = client_object_create(host, COMPUTE, COMPUTE_VERS,
                                         "udp");

    if (client_object == NULL)
    {
        client_object_pcreateerror(host);
        exit(1);
    }

    if (op == '+')
        result = add_6(&oper_6_arg, client_object);

    else if (op == '-')
        result = sub_6(&oper_6_arg, client_object);

    else if (op == '*')
        result = mul_6(&oper_6_arg, client_object);

    else if (op == '/')
    {
        if (b == 0)
        {
            printf("Division by Zero is Invalid !!!\n");
            exit(1);
        }
    }
}
```

```

        result = div_6(&oper_6_arg, client_object);
    }

    if (result == (float *)NULL)
    {
        client_object_perror(client_object, "call failed");
    }
    client_object_destroy(client_object);
    return (*result);
}

int main(int argc, char *argv[])
{
    char *host;
    float number1, number2;
    char oper;
    printf("Enter the first number:\n");
    scanf("%f", &number1);

    printf("Enter the operator (+, -, *, /):\n");
    scanf("%s", &oper);

    printf("Enter the second number:\n");
    scanf("%f", &number2);

    host = argv[1];
    printf("%f %s %f = %f\n", number1, oper, number2, compute_6(host, number1, number2, oper));
    exit(0);
}

```

Server –

```

#include "IDL.h"
#include <stdio.h>
float *
add_6_svc(values *argp, struct svc_req *rqstp)
{
    static float result;
    result = argp->num1 + argp->num2;
    return &result;
}
float *
sub_6_svc(values *argp, struct svc_req *rqstp)
{
    static float result;
    result = argp->num1 - argp->num2;
    return &result;
}
float *
mul_6_svc(values *argp, struct svc_req *rqstp)
{
    static float result;
    result = argp->num1 * argp->num2;
    return &result;
}

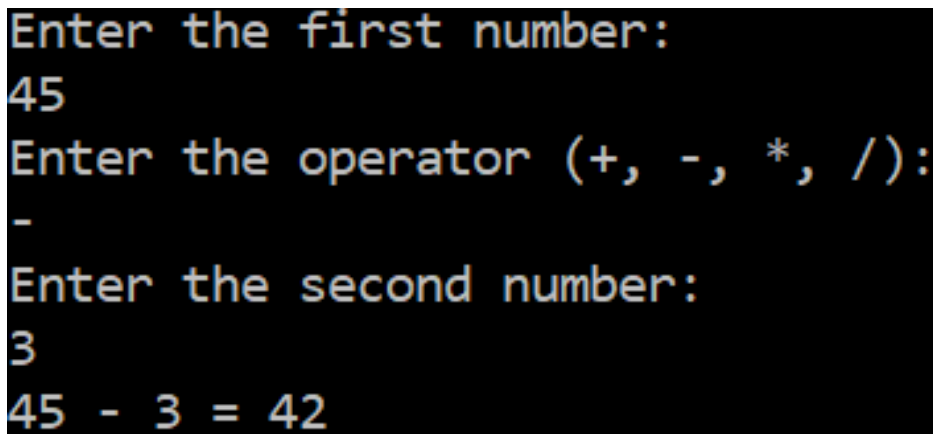
```

```

}
float *
div_6_svc(values *argp, struct svc_req *rqstp)
{
    static float result;
    result = argp->num1 / argp->num2;
    return &result;
}

```

Output:



```

Enter the first number:
45
Enter the operator (+, -, *, /):
-
Enter the second number:
3
45 - 3 = 42

```

Practical-7

Aim: Write a Program to implement chat server.

Code:

Client:

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char *msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

```



```

char buffer[255];
if (argc < 3)
{
    fprintf(stderr, "Too few arguments to run the command\n");
    exit(1);
}
portno = atoi(argv[2]);
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
{
    error("Error opening socket");
}
server = gethostbyname(argv[1]);
if (server == NULL)
{
    fprintf(stderr, "Error no such host");
    exit(1);
}
bzero((char *)&serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
serv_addr.sin_port = htons(portno);
if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    error("Connection failed");
while (1)
{
    bzero(buffer, 255);
    fgets(buffer, 255, stdin);
    n = write(sockfd, buffer, strlen(buffer));
    if (n < 0)
    {
        error("Error on writing");
    }
    bzero(buffer, 255);
    n = read(sockfd, buffer, 255);
    if (n < 0)
    {
        error("Error on reading");
    }
    printf("Server :%s\n", buffer);
    int i = strncmp("Bye", buffer, 3);
    if (i == 0)
        break;
}
close(sockfd);
return 0;
}

```

Server:

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

```

#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void error(const char *msg)
{
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
    if (argc < 2)
    {
        fprintf(stderr, "Port is not provided/n");
        exit(1);
    }
    int sockfd, newsockfd, portno, n;
    char buffer[255];
    struct sockaddr_in serv_addr, cli_addr;
    socklen_t clilen;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        error("Error opening socket\n");
    }
    bzero((char *)&serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
        error("Binding failed");
    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
    if (newsockfd < 0)
    {
        error("Error on accept");
    }
    while (1)
    {
        bzero(buffer, 255);
        n = read(newsockfd, buffer, 255);
        if (n < 0)
        {
            error("Error on reading");
        }
        printf("Client: %s\n", buffer);
        bzero(buffer, 255);
        fgets(buffer, 255, stdin);
        n = write(newsockfd, buffer, strlen(buffer));
        if (n < 0)

```

```

    {
        error("Error on writing");
    }
    int i = strncmp("Bye", buffer, 3);
    if (i == 0)
        break;
}
close(newsockfd);
close(sockfd);
return 0;
}

```

Output:

```

hi! i am client
Server :hi! i am server

```

Practical-8

Aim: Write a Program to implement termination detection.

Code:

```

#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    cout << "Enter the no. of processes: " << endl;
    cin >> n;

    int controlling_process;
    cout << "Enter the controlling process: " << endl;
    cin >> controlling_process;

    unordered_map<int, float> weight;
    for (int i = 1; i <= n; i++)
    {
        if (i == controlling_process)
            weight[i] = 1;

        else
            weight[i] = 0;
    }

    while (1)
    {

```

```

int sender, reciever;
float w;
cout << "For a message, Enter the sender process, reciever process and weight associated: " << endl;
cin >> sender >> reciever >> w;

if (weight[sender] == 0)
    cout << "Process " << sender << " is inactive ! Cannot send message !!!" << endl;
else
{
    weight[sender] = (weight[sender] <= w ? 0 : weight[sender] - w);
    weight[reciever] += w;

    if (weight[controlling_process] == 1)
    {
        cout << "Termination Detected !" << endl;
        break;
    }

    else
        cout << "Process not terminated yet ..." << endl;
}
}
return 0;
}

```

Output:

```

Enter the no. of processes:
3
Enter the controlling process:
2
For a message, Enter the sender process, reciever process and weight associated:
1 3 2
Process 1 is inactive ! Cannot send message !!!
For a message, Enter the sender process, reciever process and weight associated:
2 1 3
Process not terminated yet ...
For a message, Enter the sender process, reciever process and weight associated:
3 2 1
Process 3 is inactive ! Cannot send message !!!
For a message, Enter the sender process, reciever process and weight associated:
1 2 3
Process not terminated yet ...

```

Practical-9

Aim: Write a Program to implement RSA Algorithm.

Code:

```

#include <bits/stdc++.h>
using namespace std;

long long inverse(long long e, long long phi)
{

```

```
long long r1 = e, r2 = phi;  
long long s1 = 1, s2 = 0;
```

```
while (r2 > 0)  
{  
    long long q = r1 / r2;  
    long long r = r1 - q * r2;  
    r1 = r2;  
    r2 = r;  
  
    long long s = s1 - q * s2;  
    s1 = s2;  
    s2 = s;  
}
```

```
if (s1 < 0)  
{  
    long long q = (-1 * s1) / phi;  
    s1 += (q + 1) * phi;  
}
```

```
return s1;  
}
```

```
long long power(long long a, long long b, long long n)
```

```
{  
    if (a == 0)  
        return 0;  
  
    if (b == 0)  
        return 1;  
  
    long long val = power(a, b / 2, n);  
    val = (val * val) % n;  
  
    if (b & 1)  
        val = (val * a) % n;  
  
    return val;  
}
```

```
int main()
```

```
{  
    long long p, q, n, e, phi, d, m, c;  
    cout << "Enter the value of prime numbers p and q: " << endl;  
    cin >> p >> q;  
    n = p * q;  
    phi = (p - 1) * (q - 1);  
    cout << "Enter the value of the public key (e, where  $0 < e < \phi$  and e is coprime to  $\phi$ ): " << endl;  
    cin >> e;  
    d = inverse(e, phi);  
  
    while (1)  
    {
```

```

int choice;
cout << "MENU" << endl;
cout << "1. Encryption" << endl;
cout << "2. Decryption" << endl;
cout << "3. Quit" << endl;
cout << "Enter your choice: " << endl;
cin >> choice;
switch (choice)
{
case 1:
    cout << "Enter the value of plaintext (less than " << n << "):" << endl;
    cin >> m;

    c = power(m, e, n);
    cout << "The ciphertext is: " << c << endl;
    break;

case 2:
    cout << "Enter the value of ciphertext (less than " << n << "):" << endl;
    cin >> c;

    m = power(c, d, n);
    cout << "The plaintext is: " << m << endl;
    break;

case 3:
    exit(0);
    break;

default:
    cout << "Wrong input !!!" << endl;
    break;
}
}

return 0;
}

```

Output:

```

Enter the value of prime numbers p and q:
17 23
Enter the value of the public key (e, where  $0 < e < 352$  and e is coprime to 352):
156
MENU
1. Encryption
2. Decryption
3. Quit
Enter your choice:
1
Enter the value of plaintext (less than 391) :
342
The ciphertext is: 101
MENU
1. Encryption
2. Decryption
3. Quit
Enter your choice:
2
Enter the value of ciphertext (less than 391) :
101
The plaintext is: 288
MENU
1. Encryption
2. Decryption
3. Quit

```

Practical-10

Aim: Write a Program to implement Diffie Hellman Key Exchange Algorithm.

Code:

```

#include <bits/stdc++.h>
using namespace std;

long long power(long long a, long long b, long long n)
{
    if (a == 0)
        return 0;

    if (b == 0)
        return 1;

    long long val = power(a, b / 2, n);
    val = (val * val) % n;

    if (b & 1)
        val = (val * a) % n;

    return val;
}

```

```

int main()
{
    long long p, q, Xa, Xb, Ya, Yb;
    cout << "Enter the value of prime number p: " << endl;
    cin >> p;
    cout << "Enter the value of q where q < p and q is a primitive root of p: " << endl;
    cin >> q;
    cout << "Enter the value of the private key of person A : " << endl;
    cin >> Xa;
    Ya = power(q, Xa, p);

    cout << "Enter the value of the private key of person B : " << endl;
    cin >> Xb;
    Yb = power(q, Xb, p);

    cout << "The public key of A is: " << Ya << endl;
    cout << "The public key of B is: " << Yb << endl;

    long long k1 = power(Yb, Xa, p), k2 = power(Ya, Xb, p);

    cout << "The key calculated by A is: " << k1 << endl;
    cout << "The key calculated by B is: " << k2 << endl;

    if (k1 == k2)
        cout << "The key has been exchanged successfully !!!" << endl;

    else
        cout << "Key exchange failed !!!" << endl;

    return 0;
}

```

Output:

```

Enter the value of prime number p:
23
Enter the value of q where q < p and q is a primitive root of p:
4 16
Enter the value of the private key of person A :
Enter the value of the private key of person B :
34
The public key of A is: 12
The public key of B is: 4
The key calculated by A is: 12
The key calculated by B is: 12
The key has been exchanged successfully !!!

```