DISTRIBUTED COMPUTING

(CACSC15)

Name: Ayush Kumar

Roll No: 2019UCS2045

Branch: CSAI

Section: 1

INDEX

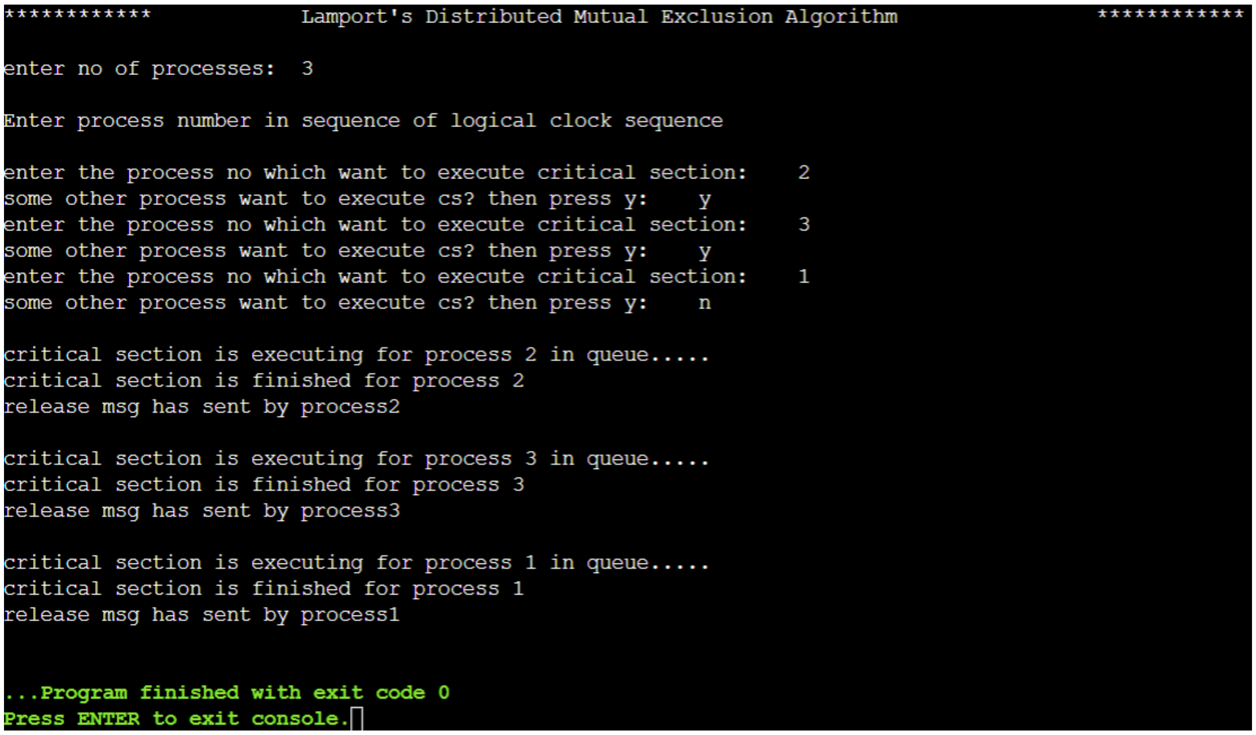
S.NO.	Program	Page No.
1.	Program to implement non token based algorithm for Mutual Exclusion	**3**
2.	Program to implement Lamport’s Logical Clock	**5**
3.	Program to implement edge chasing distributed deadlock detection algorithm	**8**
4.	Program to implement locking algorithm.	**10**
5.	Program to implement Remote Method Invocation.	**15**
6.	Program to implement Remote Procedure Call.	**17**
7.	Program to implement Chat Server.	**18**
8.	Program to implement termination detection.	**21**
9.	Program to implement RSA Algorithm.	**23**
10.	Program to implement the Diff Hellman Key Exchange Algorithm.	**27**

Q- Program to implement a non token based algorithm for Mutual Exclusion.

Source Code:

```
#include<stdio.h>

int main(){
   int number_of_processes;
   int wanting_entry[20];
   int process_id;
   int count = 0;
   char flag='y';
   printf("************\t\tLamport's Distributed Mutual Exclusion 
Algorithm\t\t************\n\n");
   printf("enter no of processes:\t");
   scanf("%d",&number_of_processes);
   printf("\nEnter process number in sequence of logical clock sequence\n\n");
   do{
      printf("enter the process no which want to execute critical section:\t");
      scanf("%d",&process_id);
      wanting_entry[count]=process_id;
      count++;
      printf("some other process want to execute cs? then press y:\t");
      scanf(" %c",&flag);
   }while(flag =='y');

   for(int j=0;j<count;j++) {
      printf("\ncritical section is executing for process %d in queue..... 
",wanting_entry[j]);
      printf("\ncritical section is finished for process %d",wanting_entry[j]);
      printf("\nrelease msg has sent by process%d\n",wanting_entry[j]);
   }

   return 0;
}
```

OUTPUT:



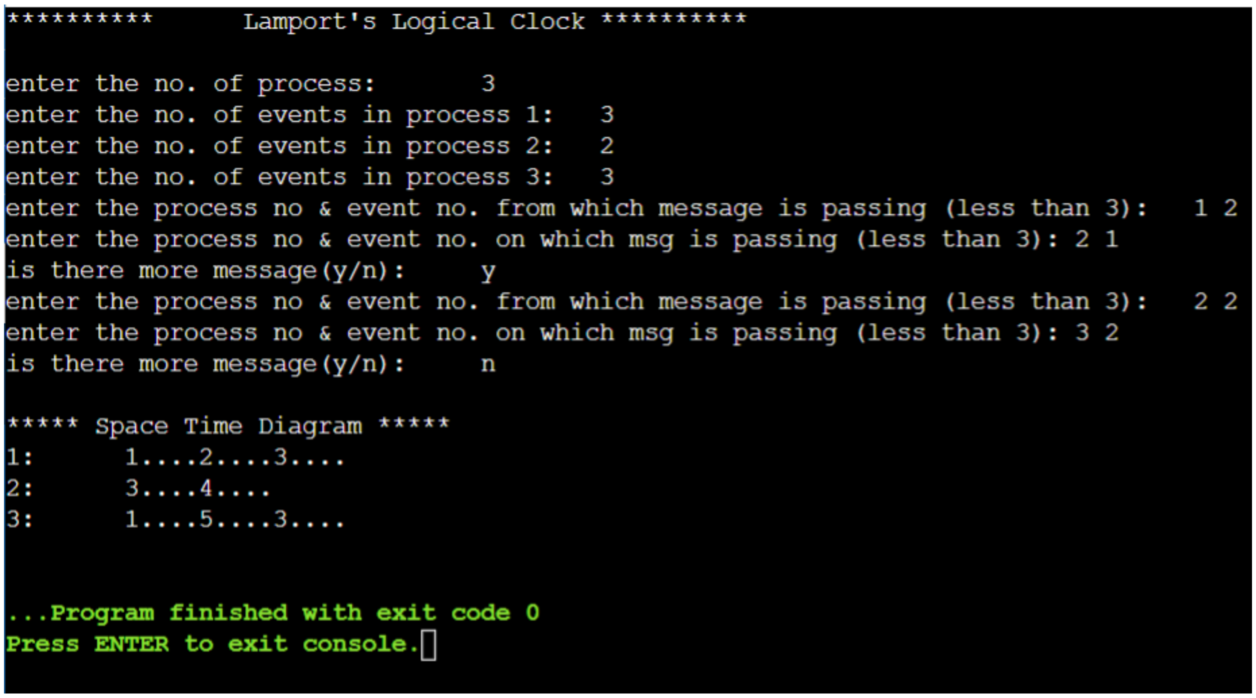
```
************            Lamport's Distributed Mutual Exclusion Algorithm                ************

enter no of processes:  3

Enter process number in sequence of logical clock sequence

enter the process no which want to execute critical section:    2
some other process want to execute cs? then press y:    y
enter the process no which want to execute critical section:    3
some other process want to execute cs? then press y:    y
enter the process no which want to execute critical section:    1
some other process want to execute cs? then press y:    n

critical section is executing for process 2 in queue.....
critical section is finished for process 2
release msg has sent by process2

critical section is executing for process 3 in queue.....
critical section is finished for process 3
release msg has sent by process3

critical section is executing for process 1 in queue.....
critical section is finished for process 1
release msg has sent by process1


...Program finished with exit code 0
Press ENTER to exit console.
```

Q- Program to implement Lamport's Logical Clock.

Source Code:

```
#include<stdio.h>

struct process{
   int event_count;
   int time_stamps[10];
}process_array[10];

int main(){
   int number_of_processes;
   int sending_process_id;
   int sending_event_id;
   int receiving_process_id;
   int receiving_event_id;
   char flag = 'y';
   printf("**********\tLamport's Logical Clock\t**********\n\n");
   printf("enter the no. of process:\t");
   scanf("%d",&number_of_processes);
   for(int i=0;i<number_of_processes;i++){
      printf("enter the no. of events in process %d:\t",i+1);
      scanf("%d",&process_array[i].event_count);
      for(int j=0;j<process_array[i].event_count;j++) {
         process_array[i].time_stamps[j]=j+1;
      }
   }
   // for(i=0;i<n;i++) {
   //    for(j=0;j<p[i].e;j++)
   //       printf("%d ",p[i].ts[j]);
   //    printf("\n");
   // }
   do{
        printf("enter the process no & event no. from which message is passing (less
than %d):\t",number_of_processes);
        scanf("%d %d",&sending_process_id,&sending_event_id);
        printf("enter the process no & event no. on which msg is passing (less than
%d):\t",number_of_processes);
        scanf("%d %d",&receiving_process_id,&receiving_event_id);

if((process_array[sending_process_id-1].time_stamps[sending_event_id-1]+1)>pro
cess_array[receiving_process_id-1].time_stamps[receiving_event_id-1])
        {

process_array[receiving_process_id-1].time_stamps[receiving_event_id-1]=proces
s_array[sending_process_id-1].time_stamps[sending_event_id-1]+1;
            for(int
i=receiving_event_id;i<process_array[receiving_process_id].event_count;i++)
            {

process_array[receiving_process_id-1].time_stamps[i]=process_array[receiving_pr
ocess_id-1].time_stamps[i-1]+1;
            }
        }
        printf("is there more message(y/n):\t");
        scanf("\t%c",&flag);
    }while(flag == 'y' || flag=='Y');
    printf("\n***** Space Time Diagram *****\n");
    for(int i=0;i<number_of_processes;i++) {
        printf("%d:\t",i+1);
        for(int j=0;j<process_array[i].event_count;j++)
            printf("%d....",process_array[i].time_stamps[j]);
        printf("\n");
    }
    return 0;
}
```

OUTPUT:



```
**********      Lamport's Logical Clock **********

enter the no. of process:       3
enter the no. of events in process 1:   3
enter the no. of events in process 2:   2
enter the no. of events in process 3:   3
enter the process no & event no. from which message is passing (less than 3):   1 2
enter the process no & event no. on which msg is passing (less than 3): 2 1
is there more message(y/n):     y
enter the process no & event no. from which message is passing (less than 3):   2 2
enter the process no & event no. on which msg is passing (less than 3): 3 2
is there more message(y/n):     n

***** Space Time Diagram *****
1:      1....2....3....
2:      3....4....
3:      1....5....3....


...Program finished with exit code 0
Press ENTER to exit console.
```

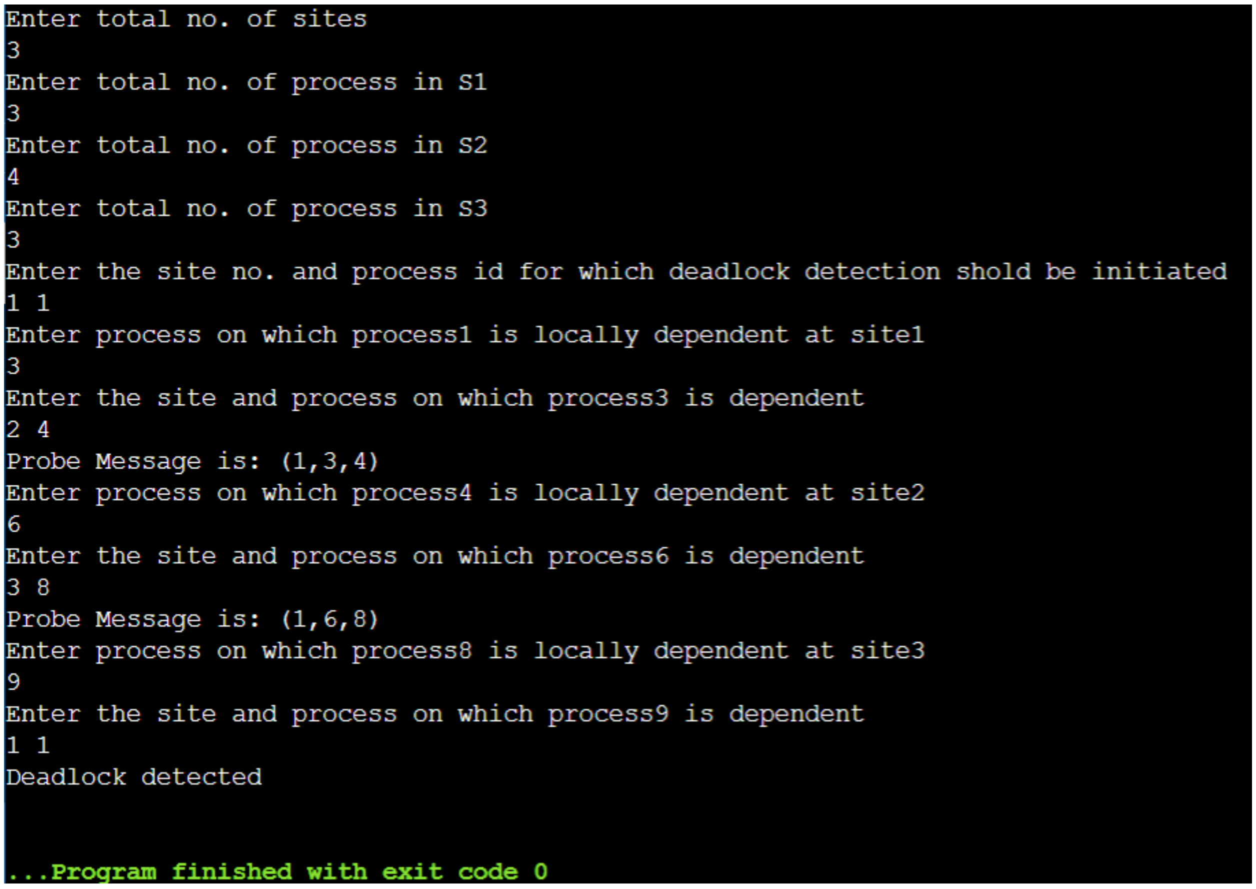
Q- Program to implement edge chasing distributed deadlock detection algorithms.

Source Code:

```
#include<stdio.h>
int main(){
   int p[10],n,i,p1,s1,s2,sp1,sp2;
   printf("Enter total no. of sites\n");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("Enter total no. of process in S%d\n",i+1);
      scanf("%d",&p[i]);
   }

   printf("Enter the site no. and process id for which deadlock detection shold be
initiated\n");
   scanf("%d %d",&s1,&p1);
   sp2 = p1;
   s2 = s1;
   while(1){
      printf("Enter process on which process%d is locally dependent at
site%d\n",sp2,s2);
      scanf("%d",&sp1);
      if(p1 == sp1){
         printf("Deadlock detected\n");
         break;
      }
      printf("Enter the site and process on which process%d is dependent\n",sp1);
      scanf("%d %d",&s2,&sp2);
      if(p1==sp2)
      {
         printf("Deadlock detected\n");
         break;
```

```
        }
        printf("Probe Message is: (%d,%d,%d)\n",p1,sp1,sp2);
    }
    return 0;
}
```

OUTPUT:



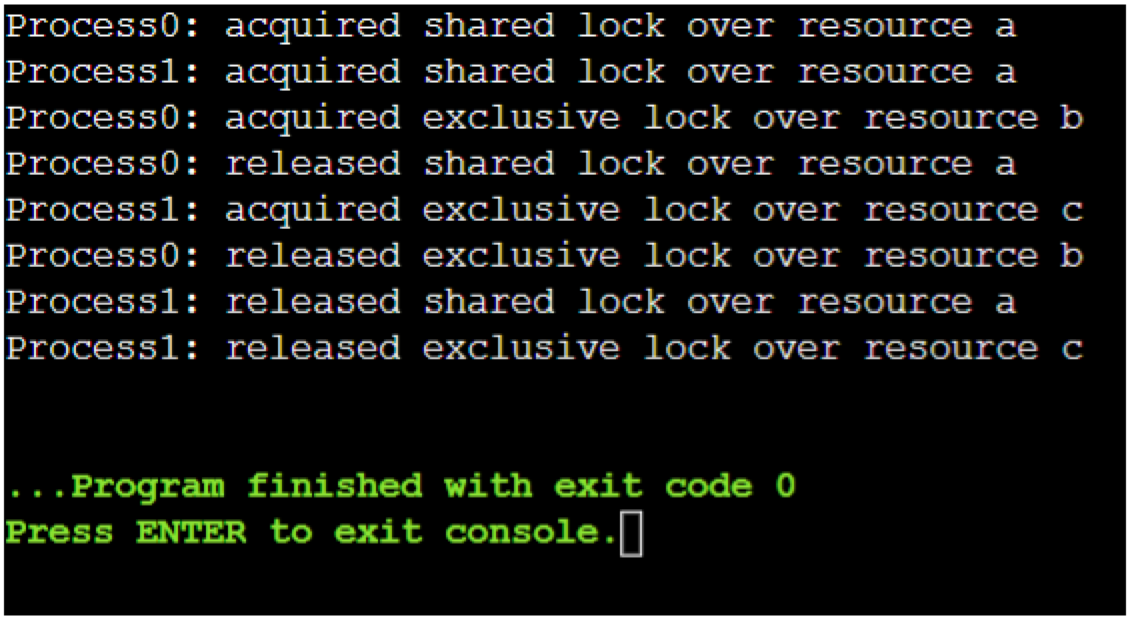
```
Enter total no. of sites
3
Enter total no. of process in S1
3
Enter total no. of process in S2
4
Enter total no. of process in S3
3
Enter the site no. and process id for which deadlock detection shold be initiated
1 1
Enter process on which process1 is locally dependent at site1
3
Enter the site and process on which process3 is dependent
2 4
Probe Message is: (1,3,4)
Enter process on which process4 is locally dependent at site2
6
Enter the site and process on which process6 is dependent
3 8
Probe Message is: (1,6,8)
Enter process on which process8 is locally dependent at site3
9
Enter the site and process on which process9 is dependent
1 1
Deadlock detected


...Program finished with exit code 0
```

Q- Program to implement locking algorithms.

Source Code:

```
#include<bits/stdc++.h>

using namespace std;

class Resource{
    char name;
    set<int> sharedLock;
    int exclusiveLock;

    public:

    Resource(char name){
        this->name = name;
        exclusiveLock = -1;
    }

    bool getsharedLock(int processId){
        if(exclusiveLock == -1){
            sharedLock.insert(processId);
            return true;
        }else{
            return false;
        }
    }

    bool getexclusiveLock(int processId){
        if(exclusiveLock == -1 && sharedLock.size() == 0){
            exclusiveLock = processId;
            return true;
        }
        return false;
    }

    void removesharedLock(int processId){
        sharedLock.erase(processId);
    }

    void removeexclusiveLock(int processId){
        exclusiveLock = -1;
    }
    char getName(){
        return name;
    }
};
Resource *resourceList[3];
vector<pair<int,pair<int,char>>> resourceRequest;
void initialise(){
    for(int i=0;i<3;i++){
        resourceList[i] = new Resource((char)('a' + i));
    }
    resourceRequest.push_back({0,{0,'a'}});
    resourceRequest.push_back({1,{0,'a'}});
    resourceRequest.push_back({0,{1,'b'}});
    resourceRequest.push_back({0,{2,'a'}});
    resourceRequest.push_back({1,{1,'c'}});
    resourceRequest.push_back({0,{3,'b'}});
    resourceRequest.push_back({1,{2,'a'}});
    resourceRequest.push_back({1,{3,'c'}});
}

int main(){
    queue<pair<int,pair<int,char>>> waiting;
    unordered_map<int,int> waitingList;
    initialise();
    int i = 0;
    while(true){
        while(waitingList.size() > 0){
            cout<<waitingList.size()<<endl;
            int processId = waiting.front().first;
            int type = waiting.front().second.first;
            char resourceName = waiting.front().second.second;
            if(type == 0){
                bool possible = resourceList[resourceName -
'a']->getsharedLock(processId);
                if(possible){
                    waitingList[processId]--;
                    if(waitingList[processId] <= 0)
                        waitingList.erase(processId);
                    cout<<"Process"<<processId<<": acquired shared lock over resource
"<<resourceName<<endl;
                }else
                    break;
            }else if(type == 1){
                bool possible = resourceList[resourceName -
'a']->getexclusiveLock(processId);
                if(possible){
                    waitingList[processId]--;
                    if(waitingList[processId] <= 0)
                        waitingList.erase(processId);
                    cout<<"Process"<<processId<<": acquired exclusive lock over
resource "<<resourceName<<endl;
                }else
                    break;
            }else if(type == 2){
                waitingList[processId]--;
                if(waitingList[processId] <= 0)
                    waitingList.erase(processId);
                resourceList[resourceName - 'a']->removesharedLock(processId);
                cout<<"Process"<<processId<<": released shared lock over resource
"<<resourceName<<endl;
            }else{
                waitingList[processId]--;
                if(waitingList[processId] <= 0)
                    waitingList.erase(processId);
                resourceList[resourceName - 'a']->removeexclusiveLock(processId);
                cout<<"Process"<<processId<<": released exclusive lock over resource
"<<resourceName<<endl;
            }
        }

        if(i < resourceRequest.size()){
            int processId = resourceRequest[i].first;
            int type = resourceRequest[i].second.first;
            char resourceName = resourceRequest[i].second.second;
            if(waitingList.find(processId) != waitingList.end()){
                waitingList[processId]++;
                waiting.push(resourceRequest[i]);
            }else{
                if(type == 0){
                    bool possible = resourceList[resourceName -
'a']->getsharedLock(processId);
                    if(possible){
                        cout<<"Process"<<processId<<": acquired shared lock over
resource "<<resourceName<<endl;
                    }else{
                        waitingList[processId]++;
                        waiting.push(resourceRequest[i]);
                    }
                }else if(type == 1){
                    bool possible = resourceList[resourceName -
'a']->getexclusiveLock(processId);
                    if(possible){
                        cout<<"Process"<<processId<<": acquired exclusive lock over
resource "<<resourceName<<endl;
                    }else{
                        waitingList[processId]++;
                        waiting.push(resourceRequest[i]);
                    }
                }else if(type == 2){
                    resourceList[resourceName - 'a']->removesharedLock(processId);
                    cout<<"Process"<<processId<<": released shared lock over resource
"<<resourceName<<endl;
                }else{
                    resourceList[resourceName - 'a']->removeexclusiveLock(processId);
                    cout<<"Process"<<processId<<": released exclusive lock over
resource "<<resourceName<<endl;
                }
            }
            i++;
        }
        if(waitingList.size() == 0 && i >= resourceRequest.size())
            break;
      }
      return 0;
}
```

OUTPUT:



```
Process0: acquired shared lock over resource a
Process1: acquired shared lock over resource a
Process0: acquired exclusive lock over resource b
Process0: released shared lock over resource a
Process1: acquired exclusive lock over resource c
Process0: released exclusive lock over resource b
Process1: released shared lock over resource a
Process1: released exclusive lock over resource c


...Program finished with exit code 0
Press ENTER to exit console.
```

Q- Program to implement Remote Method Invocation.

Source Code:

Server.py:

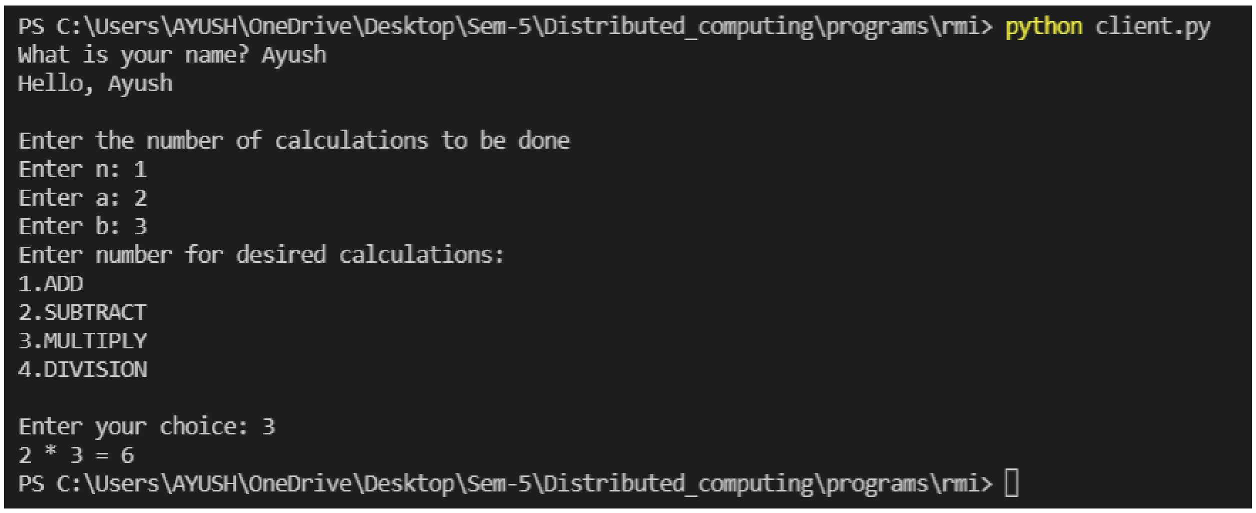
```
import Pyro4
@Pyro4.expose
class Server(object):
    def get_usid(self,name):
        return "Hello, {0}\n".format(name)
    def add(self,a,b):
        return "{0} + {1} = {2}".format(a,b,a+b)
    def subtract(self,a,b):
        return "{0} - {1} = {2}".format(a, b, a-b)
    def multiply(self,a,b):
        return "{0} * {1} = {2}".format(a, b, a*b)
    def divide(self,a,b):
        return "{0} / {1} = {2}".format(a, b, a/b)

daemon = Pyro4.Daemon()
ns = Pyro4.locateNS()
url = daemon.register(Server)
ns.register("RMI.calculator",url)
print("Server is now active")
daemon.requestLoop()
```

Client.py:

```
import Pyro4
Client = Pyro4.Proxy("PYRONAME:RMI.calculator")
name =input("What is your name? ").strip()
print(Client.get_usid(name))
print("Enter the number of calculations to be done")
n = int(input("Enter n: "))
while (n > 0):
```

```
n = n-1
a = int(input("Enter a: "))
b = int(input("Enter b: "))
print("Enter number for desired calculations: \n" + '1.ADD \n' +
     '2.SUBTRACT  \n' + '3.MULTIPLY  \n' + '4.DIVISION  \n')
c = int(input('Enter your choice: '))
if (c == 1):
   print(Client.add(a, b))
elif (c == 2):
      print(Client.subtract(a, b))
elif (c == 3):
   print(Client.multiply(a, b))
elif (c == 4):
   print(Client.division(a, b))
else:
   print('invalid input')
   break
```

OUTPUT:



```
PS C:\Users\AYUSH\OneDrive\Desktop\Sem-5\Distributed_computing\programs\rmi> python client.py
What is your name? Ayush
Hello, Ayush

Enter the number of calculations to be done
Enter n: 1
Enter a: 2
Enter b: 3
Enter number for desired calculations:
1.ADD
2.SUBTRACT
3.MULTIPLY
4.DIVISION

Enter your choice: 3
2 * 3 = 6
PS C:\Users\AYUSH\OneDrive\Desktop\Sem-5\Distributed_computing\programs\rmi>
```

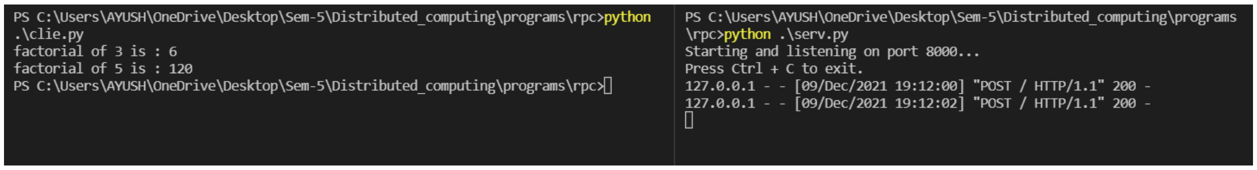
Q- Program to implement Remote Procedure Call.

Source Code:

Clie.py:
```
import xmlrpc.client
proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")
print("factorial of 3 is : %s" % str(proxy.factorial_rpc(3)))
print("factorial of 5 is : %s" % str(proxy.factorial_rpc(5)))
```

Serv.py
```
from xmlrpc.server import SimpleXMLRPCServer

def factorial(n):
    fact = 1
    for i in range(1,n+1):
        fact=fact*i
    return fact

server = SimpleXMLRPCServer(("localhost", 8000), logRequests=True)
server.register_function(factorial, "factorial_rpc")

try:
    print("Starting and listening on port 8000...")
    print("Press Ctrl + C to exit.")
    server.serve_forever()
except:
    print("Exit.")
```

OUTPUT:



```
PS C:\Users\AYUSH\OneDrive\Desktop\Sem-5\Distributed_computing\programs\rpc>python
.\clie.py
factorial of 3 is : 6
factorial of 5 is : 120
PS C:\Users\AYUSH\OneDrive\Desktop\Sem-5\Distributed_computing\programs\rpc>
```

```
PS C:\Users\AYUSH\OneDrive\Desktop\Sem-5\Distributed_computing\programs
\rpc>python .\serv.py
Starting and listening on port 8000...
Press Ctrl + C to exit.
127.0.0.1 - - [09/Dec/2021 19:12:00] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 19:12:02] "POST / HTTP/1.1" 200 -
```

Q- Program to implement Chat Server.

Source Code:

Server.py:

```
import socket
import threading

HEADER = 64
PORT = 1234
SERVER = socket.gethostbyname(socket.gethostname())

ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind(ADDR)


def handle_client(con, addr):
    print(f"[NEW CONNECTION] {addr} connected")

    connected = True
    while connected:
        msg_length = con.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg = con.recv(int(msg_length)).decode(FORMAT)
            if msg == DISCONNECT_MESSAGE:
                connected = False

            print(f"[{addr}]: {msg}")
    con.close()
```

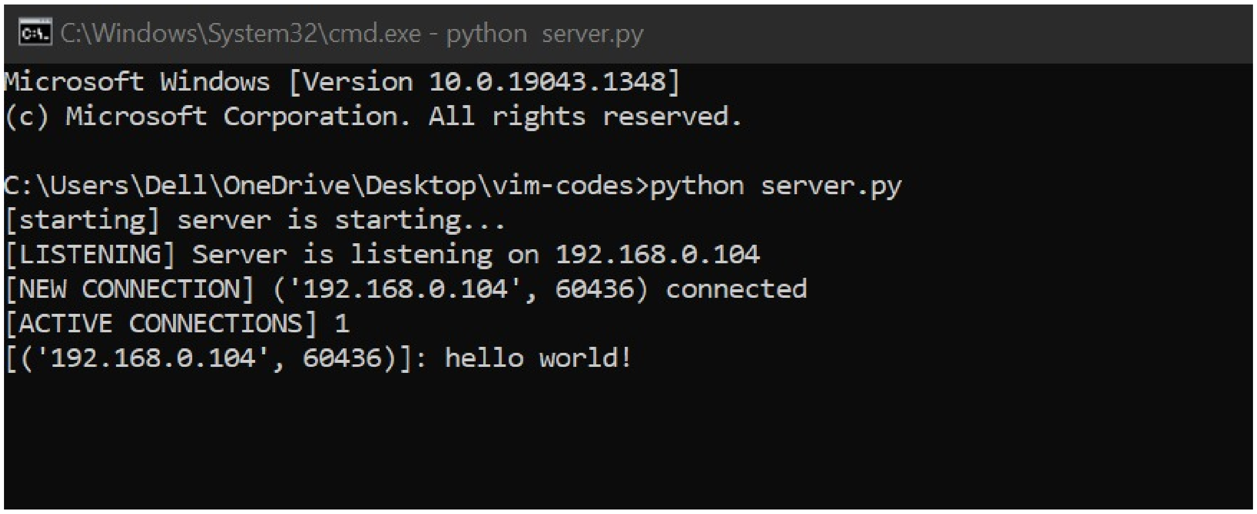
```
def start():
    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")
    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.active_count()-1}")


print("[starting] server is starting...")

start()
```

Client.py:

```
import socket
import threading

HEADER = 64
PORT = 1234
SERVER = socket.gethostbyname(socket.gethostname())


ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# creating new socket
client.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
client.connect(ADDR)
```

```
def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' '*(HEADER -len(send_length))
    client.send(send_length)
    client.send(message)



send("hello world!")
```

OUTPUT:



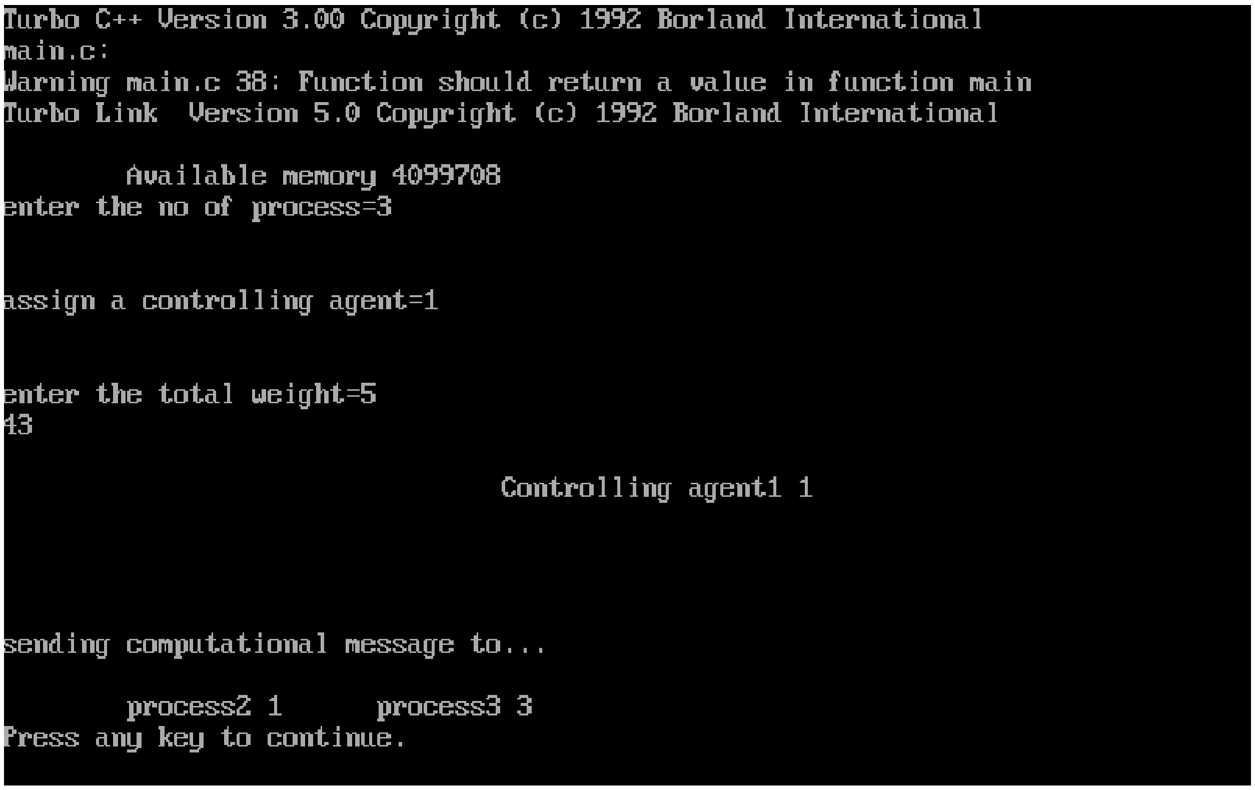
```
C:\Windows\System32\cmd.exe - python  server.py
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\OneDrive\Desktop\vim-codes>python server.py
[starting] server is starting...
[LISTENING] Server is listening on 192.168.0.104
[NEW CONNECTION] ('192.168.0.104', 60436) connected
[ACTIVE CONNECTIONS] 1
[('192.168.0.104', 60436)]: hello world!
```

Q- Program to implement termination detection.

Source Code:

```
#include<stdio.h>
#include<dos.h>
#include<stdlib.h>
#include<math.h>
int main(){
    int i,j,k=0,n,tw,total=0,we,ca,w[20];
    printf("enter the no of process=");
    scanf("%d",&n);
    printf("\n\nassign a controlling agent=");
    scanf("%d",&ca);
    printf("\n\nenter the total weight=");
    scanf("%d",&tw);
    while((k<n))
    {
        randomize();
        w[k]=random(tw);
        tw=tw-w[k];
        k++;
    }
    for(k=0;k<n;k++)
    {
        total=total+w[k];
    }
    printf("%d",total);
    w[n-1]=abs(tw-total);
    printf("%d",w[n-1]);
    printf("\n\n\t\t\t\tControlling agent%d %d\n\n\n",ca,w[ca]);
    printf("\n\nsending computational message to...\n\n");
    for(j=0;j<n;j++)
    {
        if(j!=(ca-1))
        {
           sound(700);
           delay(2000);
           printf("\tprocess%d %d",j+1,w[j]);
        }
    }
   nosound();
}
```

OUTPUT:

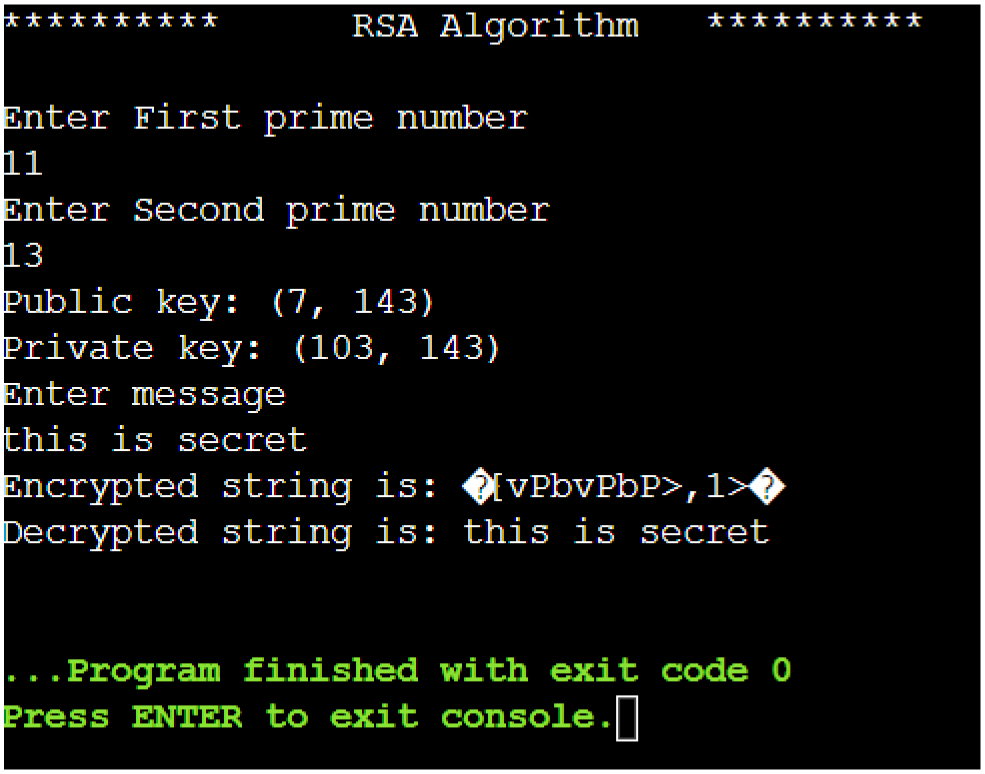


```
Turbo C++ Version 3.00 Copyright (c) 1992 Borland International
main.c:
Warning main.c 38: Function should return a value in function main
Turbo Link  Version 5.0 Copyright (c) 1992 Borland International

        Available memory 4099708
enter the no of process=3


assign a controlling agent=1


enter the total weight=5
43

                                Controlling agent1 1




sending computational message to...

        process2 1      process3 3
Press any key to continue.
```

Q- Program to implement the RSA algorithm.

Source Code:
```
#include<iostream>

typedef long long int ll;
using namespace std;

ll p,q,n,phi,e,d;
string msg;
ll msgNumeric[100],encrypted[100],decrypted[100];
ll length;

bool isPrime(ll number){
    for(ll i=2;i*i<=number;i++){
        if(number % i == 0)
            return false;
    }
    return true;
}

ll gcd(ll a,ll b){
    if(b == 0)
        return a;
    return gcd(b,a%b);
}

void encypt(){
    for(int i=0;i<length;i++){
        ll val = 1;
        for(int j=0;j<e;j++){
            val = val * msgNumeric[i];
            val = val % n;
        }
        encrypted[i] = val % n;
    }
    cout<<"Encrypted string is: ";
    for(int i=0;i<length;i++){
        cout<<(char)encrypted[i];
    }
    cout<<endl;
}

void decrypt(){
    for(int i=0;i<length;i++){
        ll val = 1;
        for(int j=0;j<d;j++){
            val = val * encrypted[i];
            val = val % n;
        }
        decrypted[i] = val % n;
    }
    cout<<"Decrypted string is: ";
    for(int i=0;i<length;i++){
        cout<<(char)decrypted[i];
    }
    cout<<endl;
}

int main(){
    cout<<"**********\tRSA Algorithm\t**********\n\n";
    cout<<"Enter First prime number\n";
    cin>>p;
    if(!isPrime(p)){
        cout<<"Wrong Input\n";
        return 1;
    }
    cout<<"Enter Second prime number\n";
    cin>>q;
    if(!isPrime(q)){
        cout<<"Wrong Input\n";
        return 1;
    }

    n = p * q;
    phi = (p - 1)*(q - 1);
    e = 2;
    while(e < phi){
        if(gcd(e,phi) == 1)
            break;
        e++;
    }
    ll product_of_e_d = 1;
    while(product_of_e_d % e != 0){
        product_of_e_d += phi;
    }
    d = product_of_e_d/e;
    cout<<"Public key: ("<<e<<", "<<n<<")\n";
    cout<<"Private key: ("<<d<<", "<<n<<")\n";
    cout<<"Enter message\n";
    cin.ignore();
    getline(cin,msg);
    length = msg.length();
    for(int i=0;i<length;i++){
        msgNumeric[i] = msg[i];
    }
    encypt();
    decrypt();
    return 0;
}
```

Output:



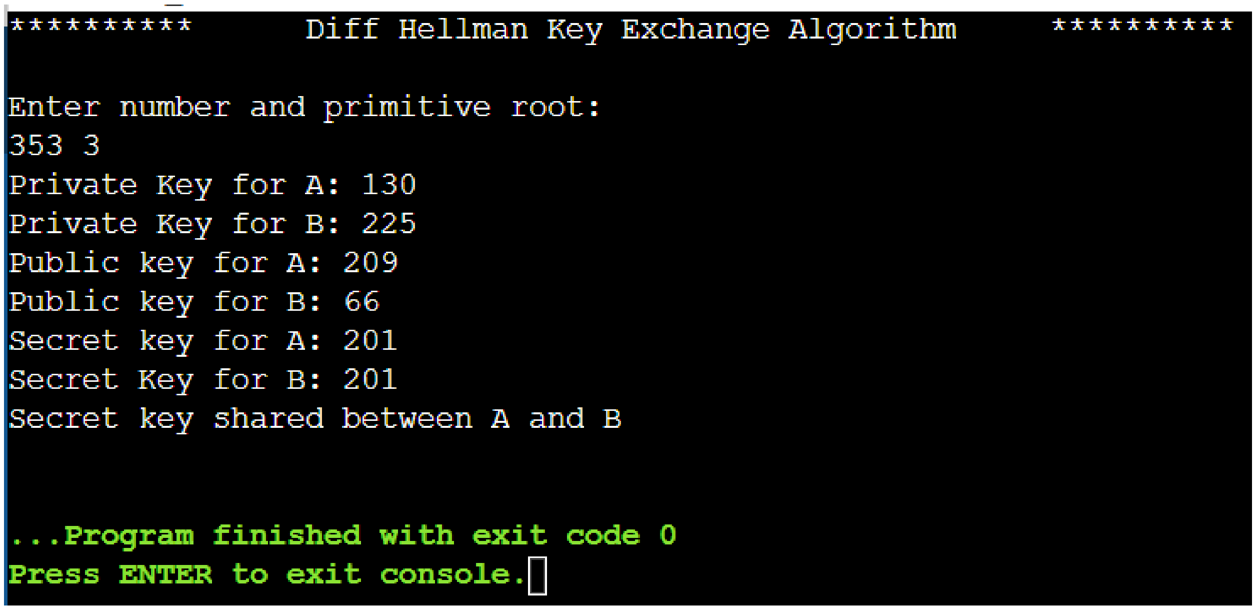
```
**********       RSA Algorithm    **********

Enter First prime number
11
Enter Second prime number
13
Public key: (7, 143)
Private key: (103, 143)
Enter message
this is secret
Encrypted string is: �[vPbvPbP>,1>�
Decrypted string is: this is secret


...Program finished with exit code 0
Press ENTER to exit console.
```

Q- Program to implement the Diff Hellman Key Exchange Algorithm.

Source Code:

```
#include<iostream>
typedef long long int ll;
using namespace std;


int main(){
    cout<<"**********\tDiff Hellman Key Exchange
Algorithm\t**********\n\n";
    ll number,primitive_root,Xa,Ya,Xb,Yb,Ka,Kb;
    cout<<"Enter number and primitive root:\n";
    cin>>number>>primitive_root;
    Xa = rand()%number;
    while(Xa == 0){
        Xa = rand()%number;
    }
    Xb = rand()%number;
    while(Xb == 0){
        Xb = rand()%number;
    }
    cout<<"Private Key for A: "<<Xa<<endl;
    cout<<"Private Key for B: "<<Xb<<endl;
    Ya = Yb = 1;
    for(int i=0;i<Xa;i++){
        Ya = Ya * primitive_root;
        Ya = Ya % number;
    }
    Ya = Ya % number;
    for(int i=0;i<Xb;i++){
        Yb = Yb * primitive_root;
        Yb = Yb % number;
    }
    Yb = Yb%number;
    cout<<"Public key for A: "<<Ya<<endl;
    cout<<"Public key for B: "<<Yb<<endl;
    Ka = Kb = 1;
    for(int i=0;i<Xa;i++){
        Ka = Ka * Yb;
        Ka = Ka % number;
    }
    Ka = Ka % number;
    cout<<"Secret key for A: "<<Ka<<endl;
    for(int i=0;i<Xb;i++){
        Kb = Kb * Ya;
        Kb = Kb % number;
    }
    Kb = Kb%number;
    cout<<"Secret Key for B: "<<Kb<<endl;
    cout<<"Secret key shared between A and B\n";
    return 0;
}
```

Output:



```
**********      Diff Hellman Key Exchange Algorithm      **********

Enter number and primitive root:
353 3
Private Key for A: 130
Private Key for B: 225
Public key for A: 209
Public key for B: 66
Secret key for A: 201
Secret Key for B: 201
Secret key shared between A and B


...Program finished with exit code 0
Press ENTER to exit console.
```