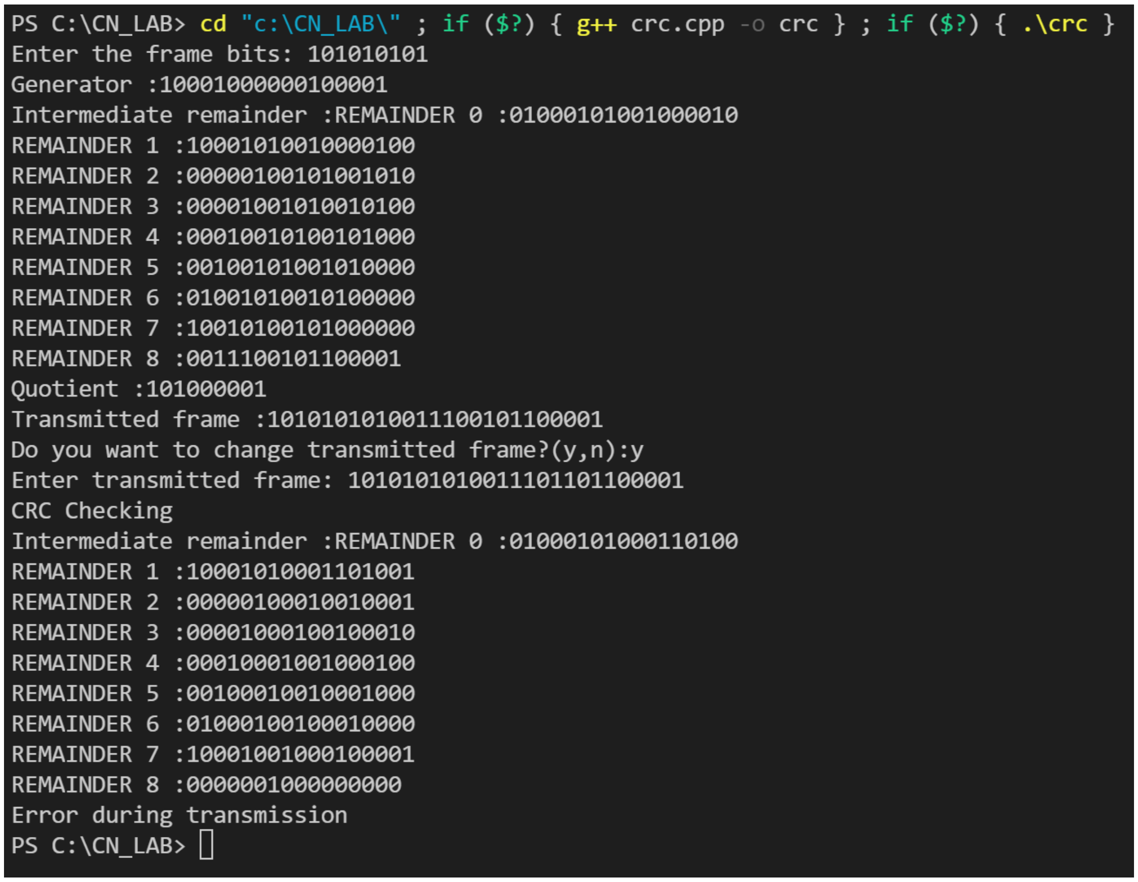
CN LAB PROGRAMS (CYCLE-2)

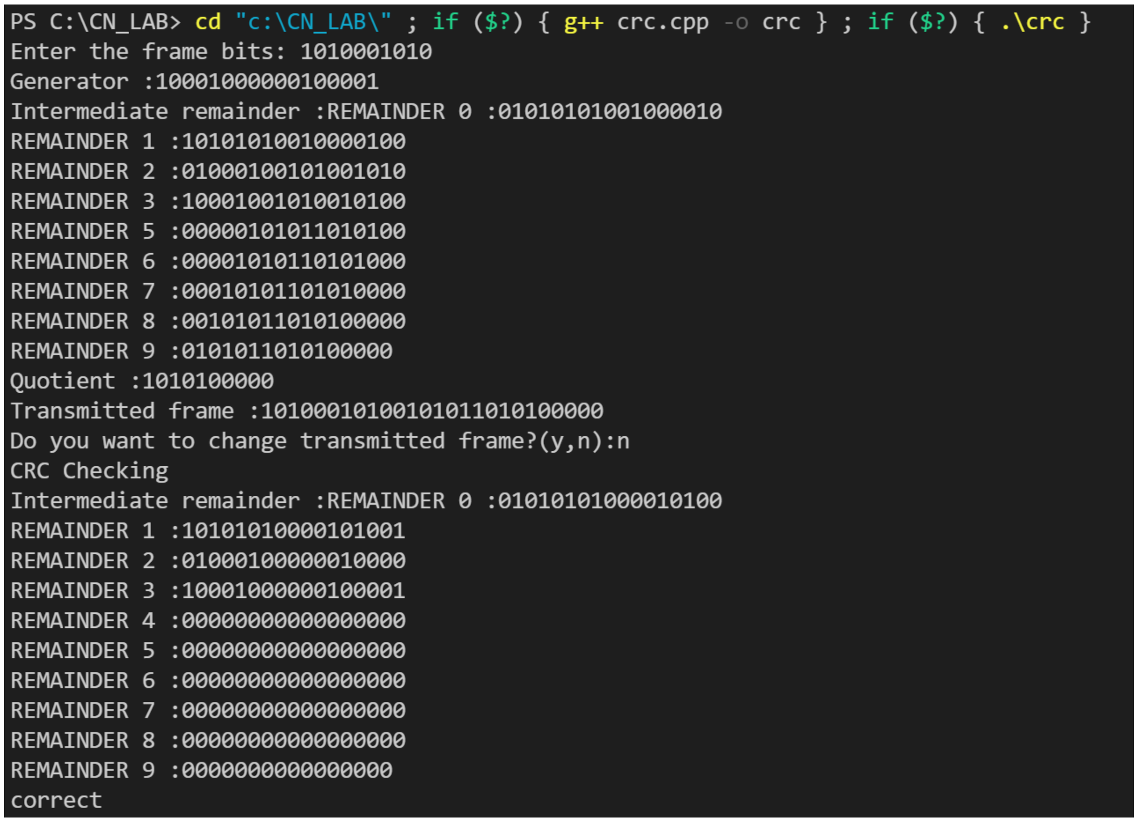
NAME : PRATHIKSHA KAMATH
USN: 1BM19CS118

1. Write a program for error detecting code using CRC-CCITT (16-bits).

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

using namespace std;

char m[50],g[50],r[50],q[50],temp[50];
void shiftleft();
void calrem()
{
    int i,j;

    for(i=1;i<=16;i++)
        r[i-1] =((int)temp[i]-48)^((int)g[i]-48)+48;
}

void crc(int n)
{


    int i,j;

    for(i = 0;i<n;i++)
    {
        temp[i]=m[i];
    }

    for(i=0;i<16;i++)
        r[i]=m[i];

    cout<<"Intermediate remainder :";
    for(i=0;i<n-16;i++)
    {
        if(r[0]=='1')
        {
            q[i]='1';
            calrem();
        }
        else
        {
            q[i]='0';
            shiftleft();
        }

        r[16]=m[17+i];
        r[17]='\0';

        cout<<"REMAINDER "<<i<<" :"<<r<<endl;

        for(j=0;j<=17;j++)
            temp[j]=r[j];
    }
    q[n-16]='\0';
}

void shiftleft()
{

    int i;
    for(i=1;i<=16;i++)
        r[i-1]=r[i];
}

void caltrans(int n)
{
    int i,k=0;
    for(i=n-16;i<n;i++)
        m[i]=((int)m[i]-48)^((int)r[k++]-48)+48;
    m[i]='\0';
}

int main()
{
  int n,i=0;
  char ch;
  int flag=0;

  cout<<"Enter the frame bits: ";
  while((ch=getc(stdin))!='\n')
    m[i++]=ch;
  n=i;
  for(i=0;i<16;i++)
    m[n++]='0';
  m[n]='\0';

  //divisor
  for(i=0;i<16;i++)
    g[i]='0';
  g[0]=g[4]=g[11]=g[16]='1';
  g[17]='\0';

  cout<<"Generator :"<<g<<endl;
  crc(n);

  cout<<"Quotient :"<<q<<endl;
  caltrans(n);
  cout<<"Transmitted frame :"<<m;
  char d;
  cout<<endl<<"Do you want to change transmitted frame?(y,n";
  cin>>d;
if(d=='y')
{
    cout<<"Enter transmitted frame: ";
    cin>>m;
}
  cout<<"CRC Checking"<<endl;
  crc(n);

  for(i=0;i<16;i++)
  {
      if(r[i]!='0')
      flag =1;

  }

      if(flag==1)
        cout<<"Error during transmission";
      else
      cout<<"correct";


}
```



```
PS C:\CN_LAB> cd "c:\CN_LAB\" ; if ($?) { g++ crc.cpp -o crc } ; if ($?) { .\crc }
Enter the frame bits: 101010101
Generator :10001000000100001
Intermediate remainder :REMAINDER 0 :01000101001000010
REMAINDER 1 :10001010010000100
REMAINDER 2 :00000100101001010
REMAINDER 3 :00001001010010100
REMAINDER 4 :00010010100101000
REMAINDER 5 :00100101001010000
REMAINDER 6 :01001010010100000
REMAINDER 7 :10010100101000000
REMAINDER 8 :0011100101100001
Quotient :101000001
Transmitted frame :1010101010011100101100001
Do you want to change transmitted frame?(y,n):y
Enter transmitted frame: 1010101010011101101100001
CRC Checking
Intermediate remainder :REMAINDER 0 :01000101000110100
REMAINDER 1 :10001010001101001
REMAINDER 2 :00000100010010001
REMAINDER 3 :00001000100100010
REMAINDER 4 :00010001001000100
REMAINDER 5 :00100010010001000
REMAINDER 6 :01000100100010000
REMAINDER 7 :10001001000100001
REMAINDER 8 :0000001000000000
Error during transmission
PS C:\CN_LAB> 
```

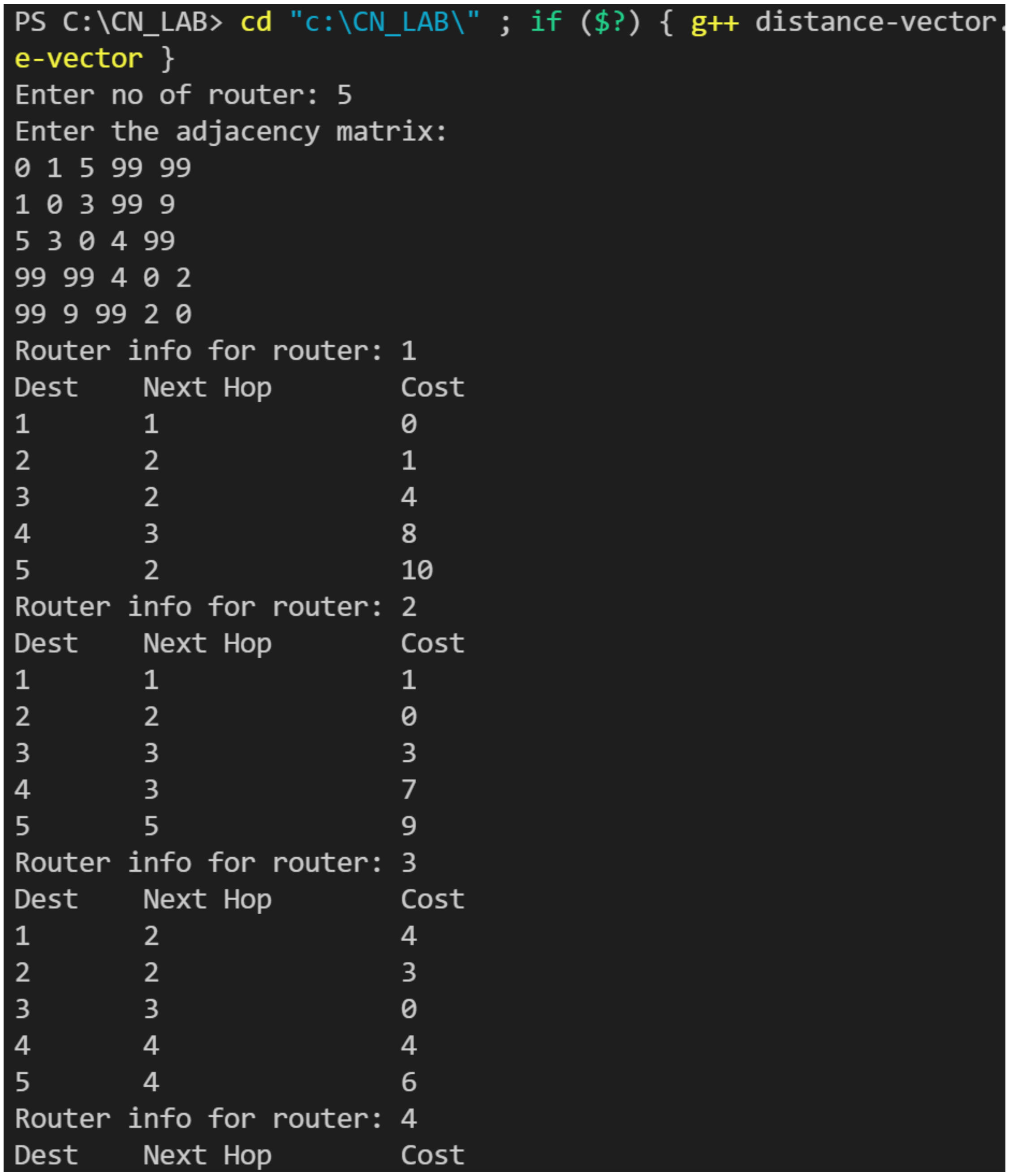


```
PS C:\CN_LAB> cd "c:\CN_LAB\" ; if ($?) { g++ crc.cpp -o crc } ; if ($?) { .\crc }
Enter the frame bits: 1010001010
Generator :10001000000100001
Intermediate remainder :REMAINDER 0 :01010101001000010
REMAINDER 1 :10101010010000100
REMAINDER 2 :01000100101001010
REMAINDER 3 :10001001010010100
REMAINDER 5 :00000101011010100
REMAINDER 6 :00001010110101000
REMAINDER 7 :00010101101010000
REMAINDER 8 :00101011010100000
REMAINDER 9 :0101011010100000
Quotient :1010100000
Transmitted frame :10100010100101011010100000
Do you want to change transmitted frame?(y,n):n
CRC Checking
Intermediate remainder :REMAINDER 0 :01010101000010100
REMAINDER 1 :10101010000101001
REMAINDER 2 :01000100000010000
REMAINDER 3 :10001000000100001
REMAINDER 4 :00000000000000000
REMAINDER 5 :00000000000000000
REMAINDER 6 :00000000000000000
REMAINDER 7 :00000000000000000
REMAINDER 8 :00000000000000000
REMAINDER 9 :0000000000000000
correct
```

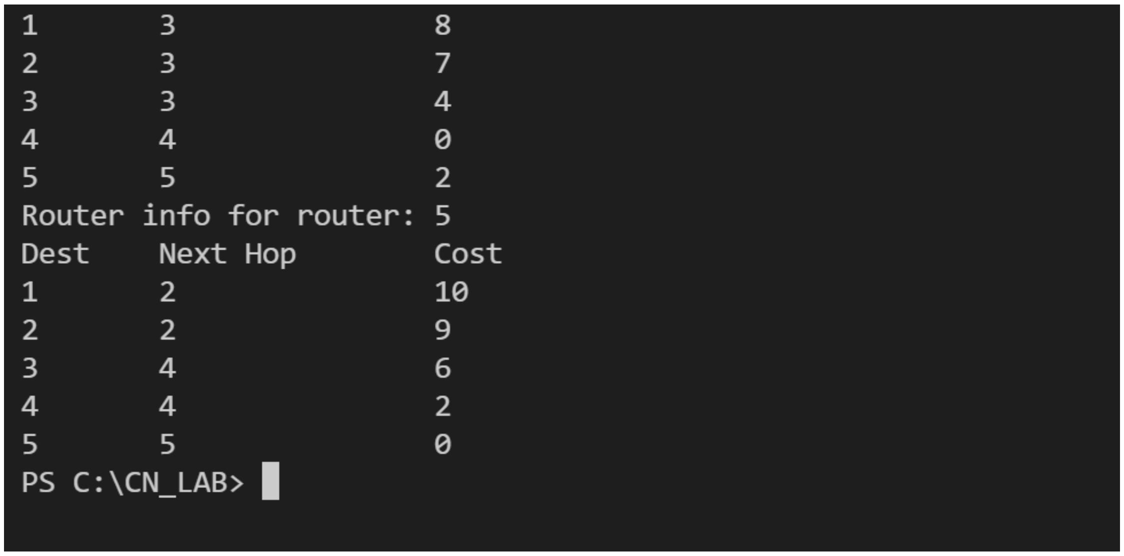
2. Write a program for distance vector algorithm to find suitable path for transmission.

```
#include <stdio.h>
#include <iostream>
using namespace std;

struct node
{
    int dist[20];
    int from[20];
} route[10];

int main()
{
    int dm[20][20], no;

    cout << "Enter no of router: "

        ;
    cin >> no;
    cout << "Enter the adjacency matrix:" << endl;
    for (int i = 0; i < no; i++)
    {
        for (int j = 0; j < no; j++)
        {
            cin >> dm[i][j];
            /*  Set distance from i to i as 0 */
            dm[i][i] = 0;
            route[i].dist[j] = dm[i][j];
            route[i].from[j] = j;
        }
    }

    int flag;
    do
    {
        flag = 0;
        for (int i = 0; i < no; i++)
        {
            for (int j = 0; j < no; j++)
            {
                for (int k = 0; k < no; k++)
                {
                    if ((route[i].dist[j]) > (route[i].dist[k] + route[k].dist[j]))
                    {
                        route[i].dist[j] = route[i].dist[k] + route[k].dist[j];
                        route[i].from[j] = k;
                        flag = 1;
                    }
                }
            }
        }
    } while (flag);

    for (int i = 0; i < no; i++)
    {
        cout << "Router info for router: " << i + 1 << endl;
        cout << "Dest\tNext Hop\tCost" << endl;
        for (int j = 0; j < no; j++)
            printf("%d\t%d\t\t%d\n", j + 1, route[i].from[j] + 1, route[i].dist[j]);
    }
    return 0;
}
```

OUTPUT:



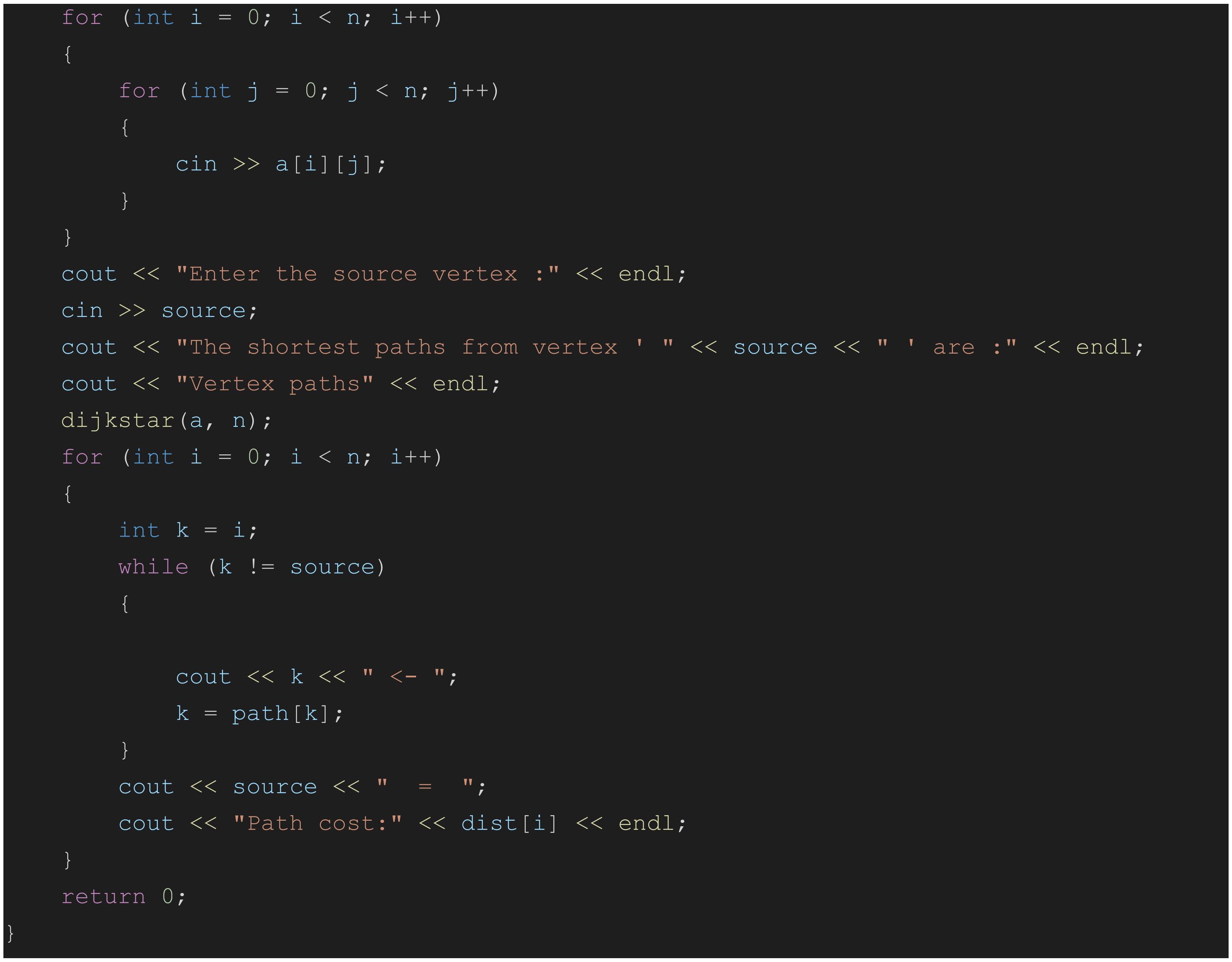
```
PS C:\CN_LAB> cd "c:\CN_LAB\" ; if ($?) { g++ distance-vector.
e-vector }
Enter no of router: 5
Enter the adjacency matrix:
0 1 5 99 99
1 0 3 99 9
5 3 0 4 99
99 99 4 0 2
99 9 99 2 0
Router info for router: 1
Dest    Next Hop        Cost
1       1               0
2       2               1
3       2               4
4       3               8
5       2               10
Router info for router: 2
Dest    Next Hop        Cost
1       1               1
2       2               0
3       3               3
4       3               7
5       5               9
Router info for router: 3
Dest    Next Hop        Cost
1       2               4
2       2               3
3       3               0
4       4               4
5       4               6
Router info for router: 4
Dest    Next Hop        Cost
```



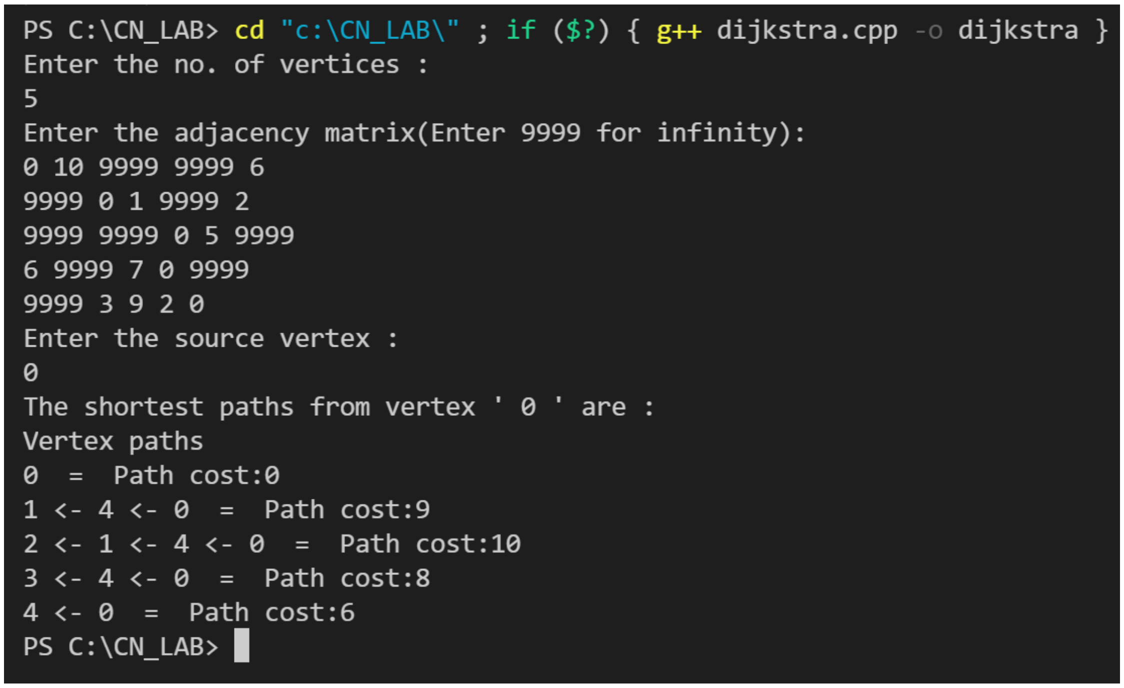
```
1       3               8
2       3               7
3       3               4
4       4               0
5       5               2
Router info for router: 5
Dest    Next Hop        Cost
1       2               10
2       2               9
3       4               6
4       4               2
5       5               0
PS C:\CN_LAB> 
```

3. Implement Dijkstra’s algorithm to compute the shortest path for a given topology.

```
#include <iostream>
using namespace std;
int a[30][30], source, dist[30], path[30];

void dijkstar(int a[][30], int n)
{
    int visited[n];
    for (int i = 0; i < n; i++)
    {
        dist[i] = a[source][i];
        path[i] = source;
        visited[i] = 0;
    }
    visited[source] = 1;
    for (int c = 0; c < n; c++)
    {
        int min = 999, u;
        for (int j = 0; j < n; j++)
        {
            if (dist[j] < min && visited[j] != 1)
            {
                min = dist[j];
                u = j;
            }
        }
        visited[u] = 1;
        for (int i = 0; i < n; i++)
        {
            if (min + a[u][i] < dist[i])
            {
                dist[i] = min + a[u][i];
                path[i] = u;
            }
        }
    }
}
int main()
{
    int n;
    cout << "Enter the no. of vertices :" << endl;
    cin >> n;
    cout << "Enter the adjacency matrix(Enter 9999 for infinity):" << endl;
```



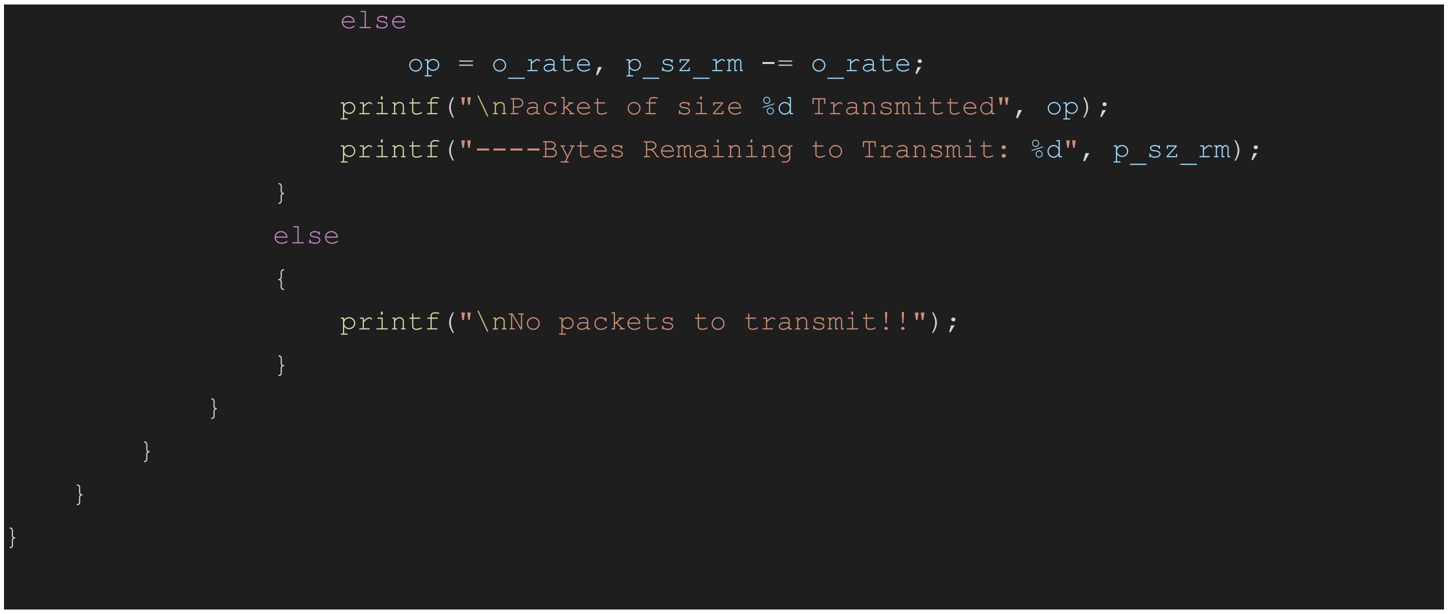
```cpp
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cin >> a[i][j];
        }
    }
    cout << "Enter the source vertex :" << endl;
    cin >> source;
    cout << "The shortest paths from vertex ' " << source << " ' are :" << endl;
    cout << "Vertex paths" << endl;
    dijkstar(a, n);
    for (int i = 0; i < n; i++)
    {
        int k = i;
        while (k != source)
        {

            cout << k << " <- ";
            k = path[k];
        }
        cout << source << "   =   ";
        cout << "Path cost:" << dist[i] << endl;
    }
    return 0;
}
```



```
PS C:\CN_LAB> cd "c:\CN_LAB\" ; if ($?) { g++ dijkstra.cpp -o dijkstra }
Enter the no. of vertices :
5
Enter the adjacency matrix(Enter 9999 for infinity):
0 10 9999 9999 6
9999 0 1 9999 2
9999 9999 0 5 9999
6 9999 7 0 9999
9999 3 9 2 0
Enter the source vertex :
0
The shortest paths from vertex ' 0 ' are :
Vertex paths
0  =  Path cost:0
1 <- 4 <- 0  =  Path cost:9
2 <- 1 <- 4 <- 0  =  Path cost:10
3 <- 4 <- 0  =  Path cost:8
4 <- 0  =  Path cost:6
PS C:\CN_LAB> 
```

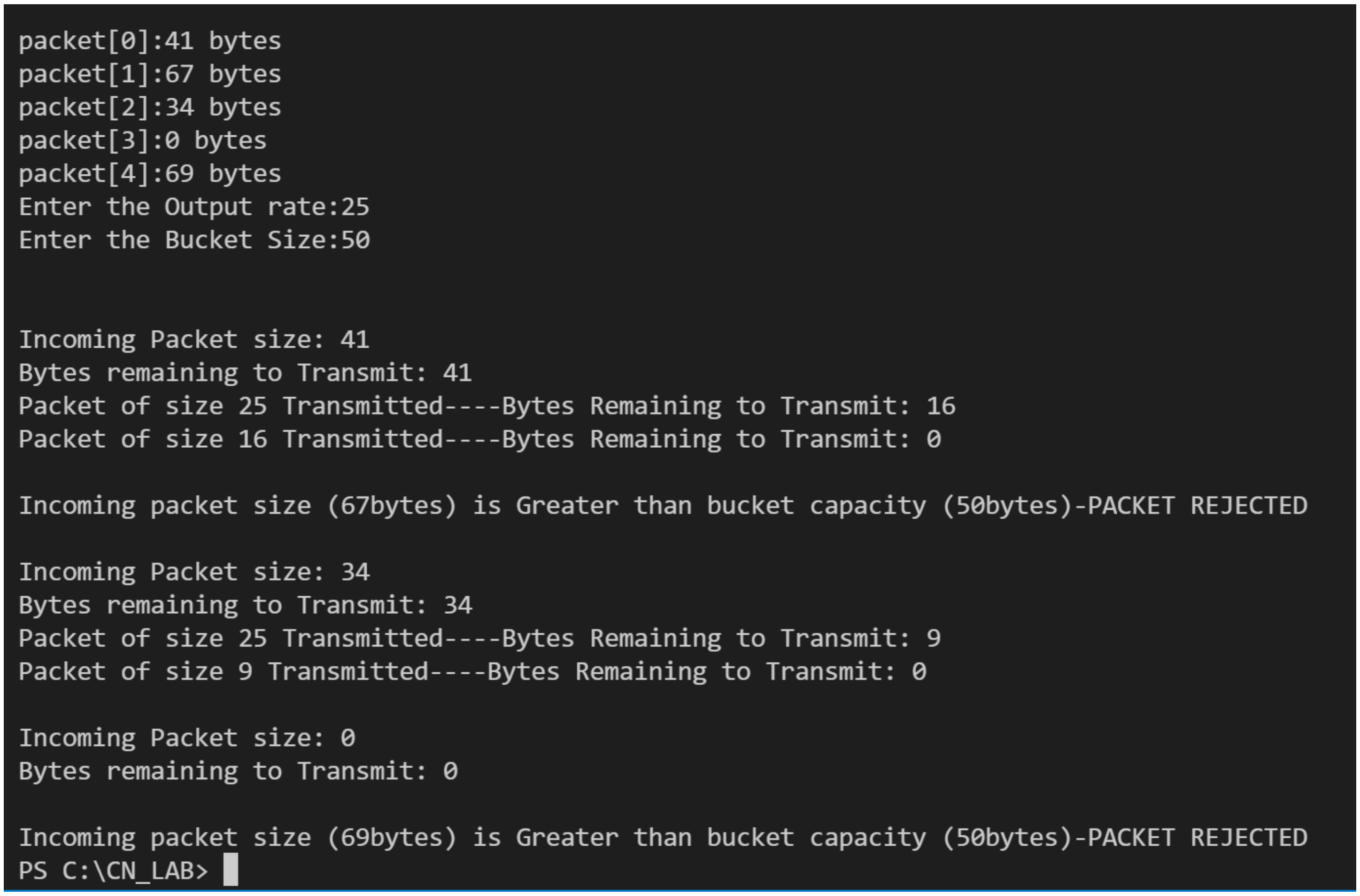
4. Write a program for congestion control using Leaky bucket algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NOF_PACKETS 5

int main()
{
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm = 0, p_sz, p_time,
op;
    for (i = 0; i < NOF_PACKETS; ++i)
        packet_sz[i] = rand() % 100;
    for (i = 0; i < NOF_PACKETS; ++i)
        printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
    printf("\nEnter the Output rate:");
    scanf("%d", &o_rate);
    printf("Enter the Bucket Size:");
    scanf("%d", &b_size);
    for (i = 0; i < NOF_PACKETS; ++i)
    {
        if ((packet_sz[i] + p_sz_rm) > b_size)
            if (packet_sz[i] > b_size) /*compare the packet siz with bucket size*/
                printf("\n\nIncoming packet size (%dbytes) is Greater than bucket
capacity (%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
            else
                printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
        else
        {
            p_sz_rm += packet_sz[i];
            printf("\n\nIncoming Packet size: %d", packet_sz[i]);
            printf("\nBytes remaining to Transmit: %d", p_sz_rm);
            //p_time = random() * 10;
            //printf("\nTime left for transmission: %d units", p_time);
            //for(clk = 10; clk <= p_time; clk += 10)
            while (p_sz_rm > 0)
            {
                sleep(1);
                if (p_sz_rm)
                {
                    if (p_sz_rm <= o_rate) /*packet size remaining comparing with
output rate*/
                        op = p_sz_rm, p_sz_rm = 0;
```



```
                    else
                        op = o_rate, p_sz_rm -= o_rate;
                    printf("\nPacket of size %d Transmitted", op);
                    printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
                }
                else
                {
                    printf("\nNo packets to transmit!!");
                }
            }
        }
    }
}
```

OUTPUT:



```
packet[0]:41 bytes
packet[1]:67 bytes
packet[2]:34 bytes
packet[3]:0 bytes
packet[4]:69 bytes
Enter the Output rate:25
Enter the Bucket Size:50


Incoming Packet size: 41
Bytes remaining to Transmit: 41
Packet of size 25 Transmitted----Bytes Remaining to Transmit: 16
Packet of size 16 Transmitted----Bytes Remaining to Transmit: 0

Incoming packet size (67bytes) is Greater than bucket capacity (50bytes)-PACKET REJECTED

Incoming Packet size: 34
Bytes remaining to Transmit: 34
Packet of size 25 Transmitted----Bytes Remaining to Transmit: 9
Packet of size 9 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 0
Bytes remaining to Transmit: 0

Incoming packet size (69bytes) is Greater than bucket capacity (50bytes)-PACKET REJECTED
PS C:\CN_LAB> 
```

5. Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present

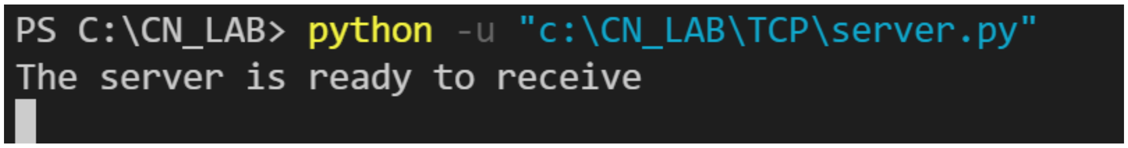
server.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file = open(sentence, "r")
    l = file.read(1024)

    connectionSocket.send(l.encode())
    print('\nSent contents of ' + sentence)
    file.close()
    connectionSocket.close()
```

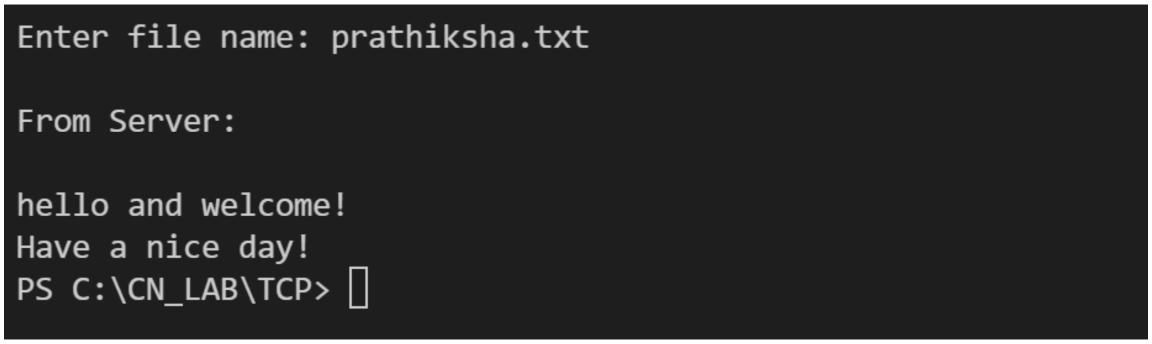
client.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\nEnter file name: ")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print('\nFrom Server:\n')
print(filecontents)
clientSocket.close()
```

OUTPUT:



```
PS C:\CN_LAB> python -u "c:\CN_LAB\TCP\server.py"
The server is ready to receive
```



```
Enter file name: prathiksha.txt

From Server:

hello and welcome!
Have a nice day!
PS C:\CN_LAB\TCP>
```

6. Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

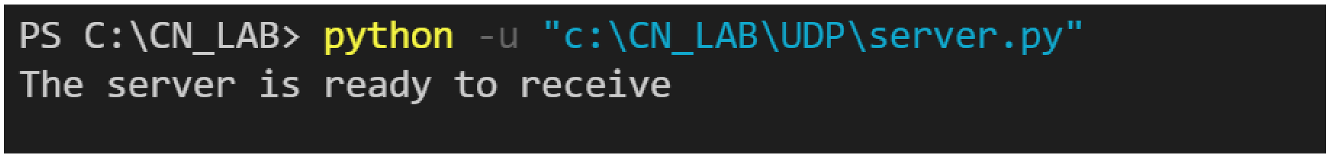
server.py

```python
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    l = file.read(2048)

    serverSocket.sendto(bytes(l, "utf-8"), clientAddress)

    print('\nSent contents of ', end=' ')
    print(sentence)
    #  for i in sentence:
    #  print (str(i), end = '')
    file.close()
```

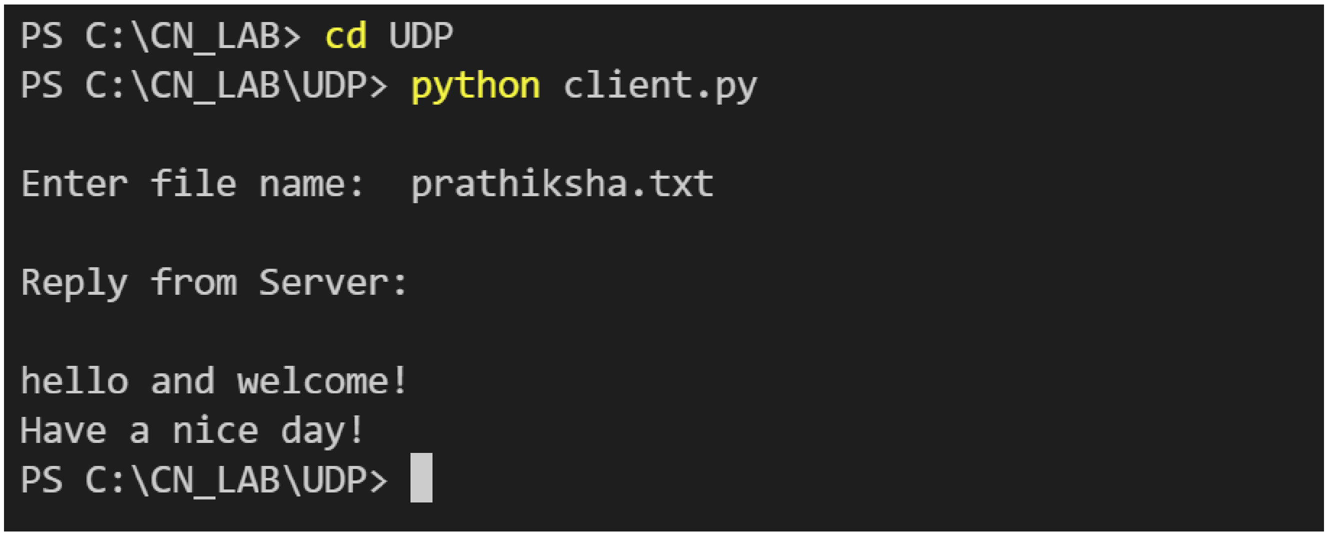
client.py

```python
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("\nEnter file name:  ")

clientSocket.sendto(bytes(sentence, "utf-8"), (serverName, serverPort))

filecontents, serverAddress = clientSocket.recvfrom(2048)
print('\nReply from Server:\n')
print(filecontents.decode("utf-8"))
# for i in filecontents:
# print(str(i), end = '')
clientSocket.close()
```

OUTPUT:



```
PS C:\CN_LAB> python -u "c:\CN_LAB\UDP\server.py"
The server is ready to receive
```



```
PS C:\CN_LAB> cd UDP
PS C:\CN_LAB\UDP> python client.py

Enter file name:  prathiksha.txt

Reply from Server:

hello and welcome!
Have a nice day!
PS C:\CN_LAB\UDP>
```