

# CN LAB RECORD

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

## CODE:

```
#include<stdio.h>
char m[50],g[50],r[50],q[50],temp[50];
void caltrans(int);
void crc(int);
void calram();
void shiftl();
int main()
{
    int n,i=0;
    char ch,flag=0;
    printf("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';
    printf("Message after appending 16 zeros:%s",m);
    for(i=0;i<=16;i++)
        g[i]='0';
    g[0]=g[4]=g[11]=g[16]='1';g[17]='\0';
    printf("\ngenerator:%s\n",g);
    crc(n);
    printf("\n\nquotient:%s",q);
    caltrans(n);
    printf("\ntransmitted frame:%s",m);
    printf("\nEnter transmitted frame:");
    scanf("\n%s",m);
    printf("CRC checking\n");
    crc(n);
    printf("\n\nlast remainder:%s",r);
    for(i=0;i<16;i++)
        if(r[i]!='0')
            flag=1;
    else
        continue;
    if(flag==1)
        printf("Error during transmission");
    else
        printf("\n\nReceived frame is correct");
}
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];
printf("\nintermediate remainder\n");
for(i=0;i<n-16;i++)
{
if(r[0]=='1')
{
q[i]='1';
calram();
}
else
{
q[i]='0';
shifftl();
}
r[16]=m[17+i];
r[17]='\0';
printf("\nremainder %d:%s",i+1,r);
for(j=0;j<=17;j++)
temp[j]=r[j];
}
q[n-16]='\0';
}
void calram()
{
int i,j;
for(i=1;i<=16;i++)
r[i-1]=((int)temp[i]-48)^((int)g[i]-48)+48;
}
void shifftl()
{
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';
}

```

## OUTPUT:

```
PS C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2> cd "c:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN\" ; if ($?) { gcc CRC.c -o CRC } ,
Enter the frame bits:110110
Message after appending 16 zeros:110110000000000000000000
generator:10001000000100001

intermediate remainder

remainder 1:10100000001000010
remainder 2:01010000011000110
remainder 3:10100000110001100
remainder 4:01010001101011010
remainder 5:10100011010110100
remainder 6:0101011010010101

quotient:110101
transmitted frame:1101100101011010010101
Enter transmitted frame:1101100101011010010101
CRC checking

intermediate remainder

remainder 1:10100010100101011
remainder 2:01010101000010100
remainder 3:10101010000101001
remainder 4:01000100000010000
remainder 5:10001000000100001
remainder 6:00000000000000000

last remainder:00000000000000000

Received frame is correct
```

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

**CODE:**

```
#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 nodes." << endl;
    cin >> no;
    cout << "Enter the distance 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\tDist" << 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:\Users\PUNEETH K\OneDrive\Desktop\LAB_2> cd "c:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN\" ; if ($?) { g++ DVR.cpp -o DVR }  
Enter no of nodes.  
4  
Enter the distance matrix:  
10 2 5 6  
7 2 1 9  
45 2 8 1  
5 4 3 8  
Router info for router: 1  
Dest    Next Hop    Dist  
1        1            0  
2        2            2  
3        2            3  
4        3            4  
Router info for router: 2  
Dest    Next Hop    Dist  
1        1            7  
2        2            0  
3        3            1  
4        3            2  
Router info for router: 3  
Dest    Next Hop    Dist  
1        4            6  
2        2            2  
3        3            0  
4        4            1  
Router info for router: 4  
Dest    Next Hop    Dist  
1        1            5  
2        2            4  
3        3            3  
4        4            0
```

### LAB 3: Implement Dijkstra's algorithm to compute the shortest path for a given topology.

#### CODE:

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

#define V 4

int minDistance(int dist[], bool sptSet[])
{
    int min = 9999, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

void printPath(int parent[], int j)
{
    if (parent[j] == - 1)
        return;

    printPath(parent, parent[j]);

    cout<<j<<" ";
}

void printSolution(int dist[], int n, int parent[])
{
    int src = 0;
    cout<<"Vertex\t Distance\tPath"<<endl;
    for (int i = 1; i < V; i++)
    {
        cout<<"\n"<<src<<" -> "<<i<<" \t
\t"<<dist[i]<<"\t\t"<<src<<" ";
        printPath(parent, i);
    }
}

void dijkstra(int graph[V][V], int src)
{
    int dist[V];

    bool sptSet[V];

    int parent[V];

    for (int i = 0; i < V; i++)
    {
```

```

        parent[0] = -1;
        dist[i] = 9999;
        sptSet[i] = false;
    }

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++)
    {
        int u = minDistance(dist, sptSet);

        sptSet[u] = true;

        for (int v = 0; v < V; v++)

            if (!sptSet[v] && graph[u][v] &&
                dist[u] + graph[u][v] < dist[v])
            {
                parent[v] = u;
                dist[v] = dist[u] + graph[u][v];
            }
    }

    printSolution(dist, V, parent);
}

int main()
{
    int graph[V][V];
    cout<<"Please Enter The Graph (!!! Use 99 for infinity):
"<<endl;
    for(int i = 0; i<V; i++)
    {
        for(int j = 0; j<V; j++)
            cin>>graph[i][j];
    }
    cout<<"Enter the source vertex: "<<endl;
    int src;
    cin>>src;

    dijkstra(graph, src);
    cout<<endl;
    return 0;
}

```

## OUTPUT:

```
PS C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2> cd "c:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN\" ; if ($?) { g++ Dijkstra.cpp -o
Please Enter The Graph (!!! Use 99 for infinity):
0 4 5 99
45 0 2 6
5 8 0 99
4 12 6 0
Enter the source vertex:
0
Vertex    Distance    Path
0 -> 1      4          0 1
0 -> 2      5          0 2
0 -> 3     10          0 1 3
PS C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN> █
```



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

**CODE:**

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>


#define NOF_PACKETS 5

/*
int rand (int a)
{
    int rn = (random() % 10) % a;
    return rn == 0 ? 1 : rn;
}
*/

#include <stdlib.h>


long int random(void);
```

The random() function uses a nonlinear additive feedback random number generator employing a default ta-

ble of size 31 long integers to return successive pseudo-random numbers in the range from 0 to RAND\_MAX.

The period of this random number generator is very large, approximately  $16 * ((2^{31}) - 1)$ .

```
*/
```

```

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] = random() % 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]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:82
Enter the Bucket Size:45

Incoming packet size (83bytes) is Greater than bucket capacity (45bytes)-PACKET REJECTED
Incoming packet size (86bytes) is Greater than bucket capacity (45bytes)-PACKET REJECTED
Incoming packet size (77bytes) is Greater than bucket capacity (45bytes)-PACKET REJECTED
Incoming Packet size: 15
Bytes remaining to Transmit: 15
Packet of size 15 Transmitted----Bytes Remaining to Transmit: 0
Incoming packet size (93bytes) is Greater than bucket capacity (45bytes)-PACKET REJECTED
...Program finished with exit code 0
Press ENTER to exit console.
```

**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.

**CODE:**

ServerTCP.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()
```

ClientTCP.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())
```

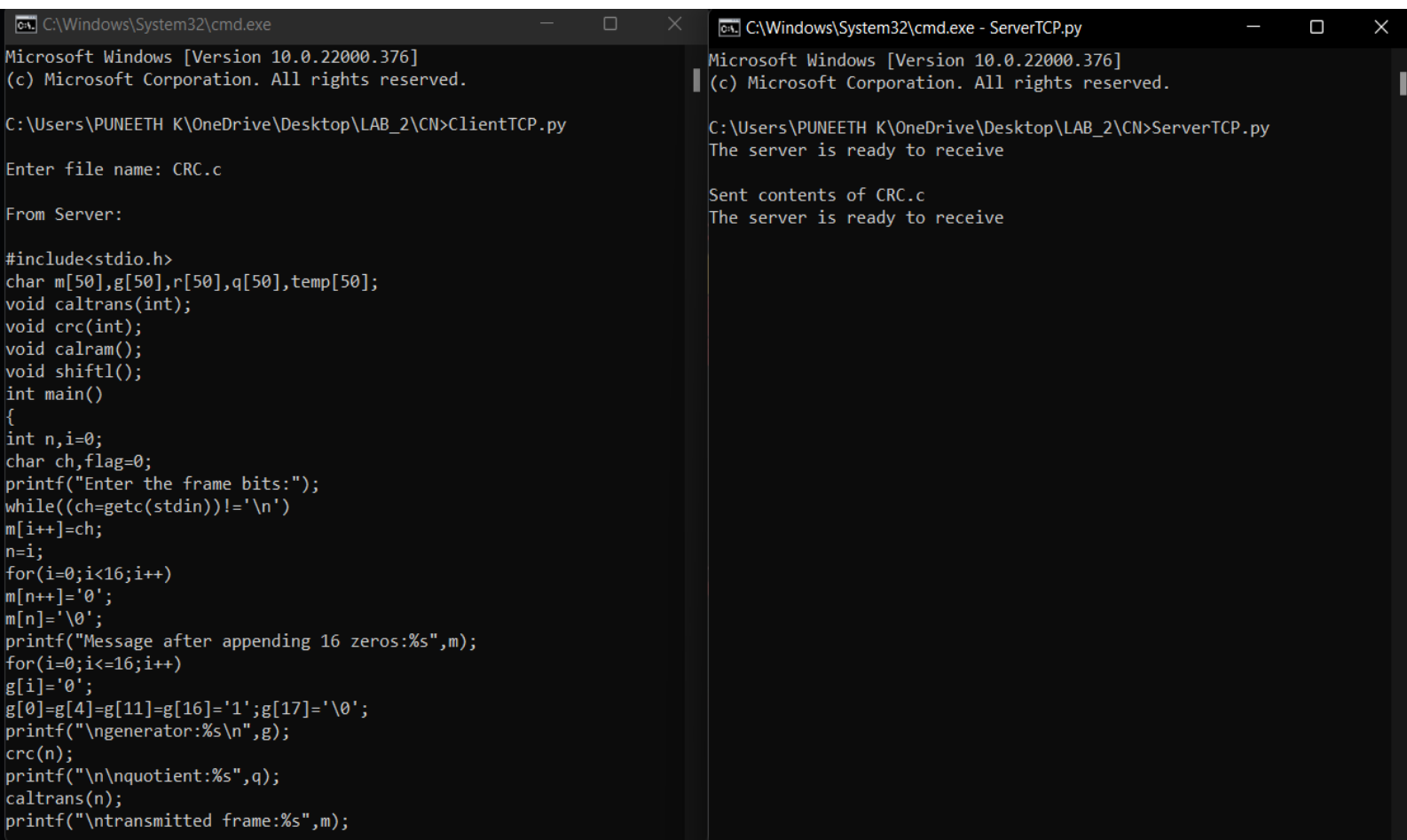
```
filecontents = clientSocket.recv(1024).decode()
```

```
print ('\nFrom Server:\n')
```

```
print(filecontents)
```

```
clientSocket.close()
```

## OUTPUT:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN>ClientTCP.py

Enter file name: CRC.c

From Server:

#include<stdio.h>
char m[50],g[50],r[50],q[50],temp[50];
void caltrans(int);
void crc(int);
void calram();
void shiftl();
int main()
{
    int n,i=0;
    char ch,flag=0;
    printf("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';
    printf("Message after appending 16 zeros:%s",m);
    for(i=0;i<16;i++)
        g[i]='0';
    g[0]=g[4]=g[11]=g[16]='1';g[17]='\0';
    printf("\ngenerator:%s\n",g);
    crc(n);
    printf("\n\nquotient:%s",q);
    caltrans(n);
    printf("\ntransmitted frame:%s",m);

C:\Windows\System32\cmd.exe - ServerTCP.py
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN>ServerTCP.py
The server is ready to receive

Sent contents of CRC.c
The server is ready to receive
```

**LAB 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.

**CODE:**

ServerUDP.py

```
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()
```

ClientUDP.py

```
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"))
```

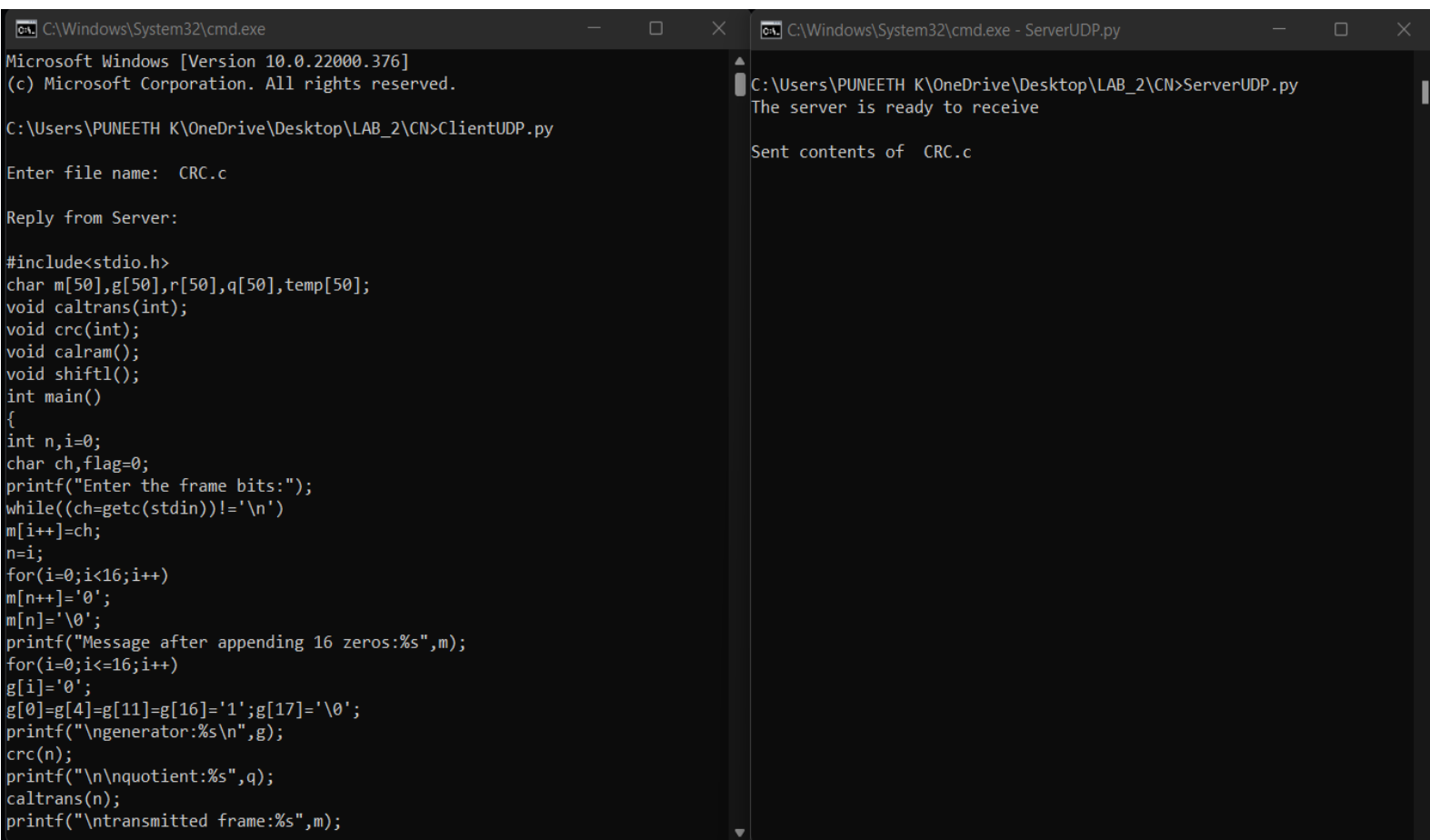
```
# for i in filecontents:
```

```
    # print(str(i), end = '')
```

```
clientSocket.close()
```

```
clientSocket.close()
```

## OUTPUT



The screenshot shows two terminal windows side-by-side. The left window, titled 'C:\Windows\System32\cmd.exe', shows the execution of 'ClientUDP.py'. It prompts for a file name, receives 'CRC.c', and then displays the source code of the client program. The code includes headers, defines arrays, and implements functions for CRC calculation and frame transmission. The right window, titled 'C:\Windows\System32\cmd.exe - ServerUDP.py', shows the execution of 'ServerUDP.py'. It displays the message 'The server is ready to receive' and 'Sent contents of CRC.c'.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN>ClientUDP.py

Enter file name: CRC.c

Reply from Server:

#include<stdio.h>
char m[50],g[50],r[50],q[50],temp[50];
void caltrans(int);
void crc(int);
void calram();
void shiftl();
int main()
{
int n,i=0;
char ch,flag=0;
printf("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';
printf("Message after appending 16 zeros:%s",m);
for(i=0;i<16;i++)
g[i]='0';
g[0]=g[4]=g[11]=g[16]='1';g[17]='\0';
printf("\ngenerator:%s\n",g);
crc(n);
printf("\n\nquotient:%s",q);
caltrans(n);
printf("\ntransmitted frame:%s",m);
```

```
C:\Windows\System32\cmd.exe - ServerUDP.py
C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN>ServerUDP.py
The server is ready to receive

Sent contents of CRC.c
```



