

CN LAB RECORD

PANKAJ GUPTA

1BM19CS110

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

CODE:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#define N strlen(g)
char t[28], cs[28],g[28];
char g[] = "10001000000100001";
int a,e,c,b;
void xor(){
    for(c=1;c<N;c++)
        cs[c] = ((cs[c]==g[c])?'0':'1');
}

void crc()
{
    for(e=0;e<N;e++)
        cs[e]=t[e];
    do
    {
        if(cs[0]=='1')
            xor();
        for(c=0;c<N-1;c++)
            cs[c]=cs[c+1];
        cs[c]=t[e++];
    }while(e<=a+N-1);
}

int main(){
    printf("Enter the polynomial\n");
    scanf("%s",t);
    printf("Generating polynomial = %s\n",g);
    a = strlen(t);
    for(e=a;e<a+N-1;e++)
        t[e] = '0';
    printf("Polynomial after adding zeros %s\n",t);
    crc();
    printf("Remainder = %s\n",cs);
}
```

```

for(e=a;e<a+N-1;e++)
    t[e] = cs[e-a];
printf("Final codeword = %s\n",t);
printf("Test error detection 0(yes)or 1(no)\n");
scanf("%d",&e);
if(e==0){
    printf("Position where error need to be inserted\n");
    scanf("%d",&e);
    t[e] = (t[e]=='0')?'1':'0';
    printf("Data contains error %s\n",t);
}
crc();
for(e=0;(e<N-1) && (cs[e]!='1'); e++);
if(e<N-1)
    printf("Error Detected\n");
else
    printf("No error detected\n");
return 0;
}

```

OUTPUT:

```
C:\Users\panka\Desktop\tomorrow1\CRC.exe
Enter the polynomial
1011
Generating polynomial = 10001000000100001
Polynomial after adding zeros 1011000000000000000
Remainder = 1011000101101011
Final codeword = 10111011000101101011
Test error detection 0(yes)or 1(no)
0
Position where error need to be inserted
2
Data contains error 10011011000101101011
Error Detected

Process returned 0 (0x0)   execution time : 20.743 s
Press any key to continue.
```

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

CODE:

```
#include<stdio.h>
#define inf 999
struct routing{
    int dist[10];
    int hop[10];
};
struct routing nodes[10];

void init(int n){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            if(i!=j){
                nodes[i].dist[j] = inf;
                nodes[i].hop[j] = -20;
            }
            else{
                nodes[i].dist[j] = 0;
                nodes[i].hop[j] = -20;
            }
        }
    }
}

void update(int i,int j,int k){
    nodes[i].hop[j] = k;
    nodes[i].dist[j] = nodes[i].dist[k] + nodes[k].dist[j];
}

void dvr(int n){
    int i,j,k;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            for(k=0;k<n;k++)
                if(nodes[i].dist[j]>(nodes[i].dist[k] + nodes[k].dist[j]))
                    update(i,j,k);
}

int main(){
    int i, j, n;
    printf("Enter the number of nodes\n");
    scanf("%d",&n);
    init(n);
    printf("Enter the distance vector\n");
    for(i=0;i<n;i++){
        printf("Enter for node %d\n",i);
        for(j=0;j<n;j++){
            scanf("%d",&nodes[i].dist[j]);
        }
    }
}
```

```

        }
    }
    dvr(n);
    printf("\nUpdated distance vector table\n");
    for(i=0;i<n;i++){
        printf("Updated node %c table\n",65+i);
        printf("To\t cost\t hop\n");
        for(j=0;j<n;j++){
            printf("%c\t %d\t %c\n",65+j,nodes[i].dist[j],
65+nodes[i].hop[j]);
        }
    }

    return 0;
}

```

OUTPUT:

```
C:\Users\panka\Desktop\tomorrow1\distanceVectorRouting.exe
999
3
4
999
0

Updated distance vector table
Updated node A table
To      cost  hop
A        0    -
B        5    -
C        2    -
D        3    -
E        6    C
Updated node B table
To      cost  hop
A        5    -
B        0    -
C        4    -
D        8    A
E        3    -
Updated node C table
To      cost  hop
A        2    -
B        4    -
C        0    -
D        5    A
E        4    -
Updated node D table
To      cost  hop
A        3    -
B        8    A
C        5    A
D        0    -
E        9    A
Updated node E table
To      cost  hop
A        6    C
B        3    -
C        4    -
D        9    A
E        0    -

Process returned 0 (0x0)   execution time : 102.575 s
Press any key to continue.
```

```
C:\Users\panka\Desktop\tomorrow1\distanceVectorRouting.exe
5
Enter the distance vector
Enter for node 0
0
5
2
3
999
Enter for node 1
5
0
4
999
3
Enter for node 2
2
4
0
999
4
Enter for node 3
3
999
999
0
999
Enter for node 4
999
3
4
999
0

Updated distance vector table
Updated node A table
To      cost  hop
A        0    -
B         5    -
C         2    -
D         3    -
E         6    C
Updated node B table
To      cost  hop
A         5    -
B         0    -
C         4    -
D         8    A
E         3    -
Updated node C table
```

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:

```
input
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

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

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())
```

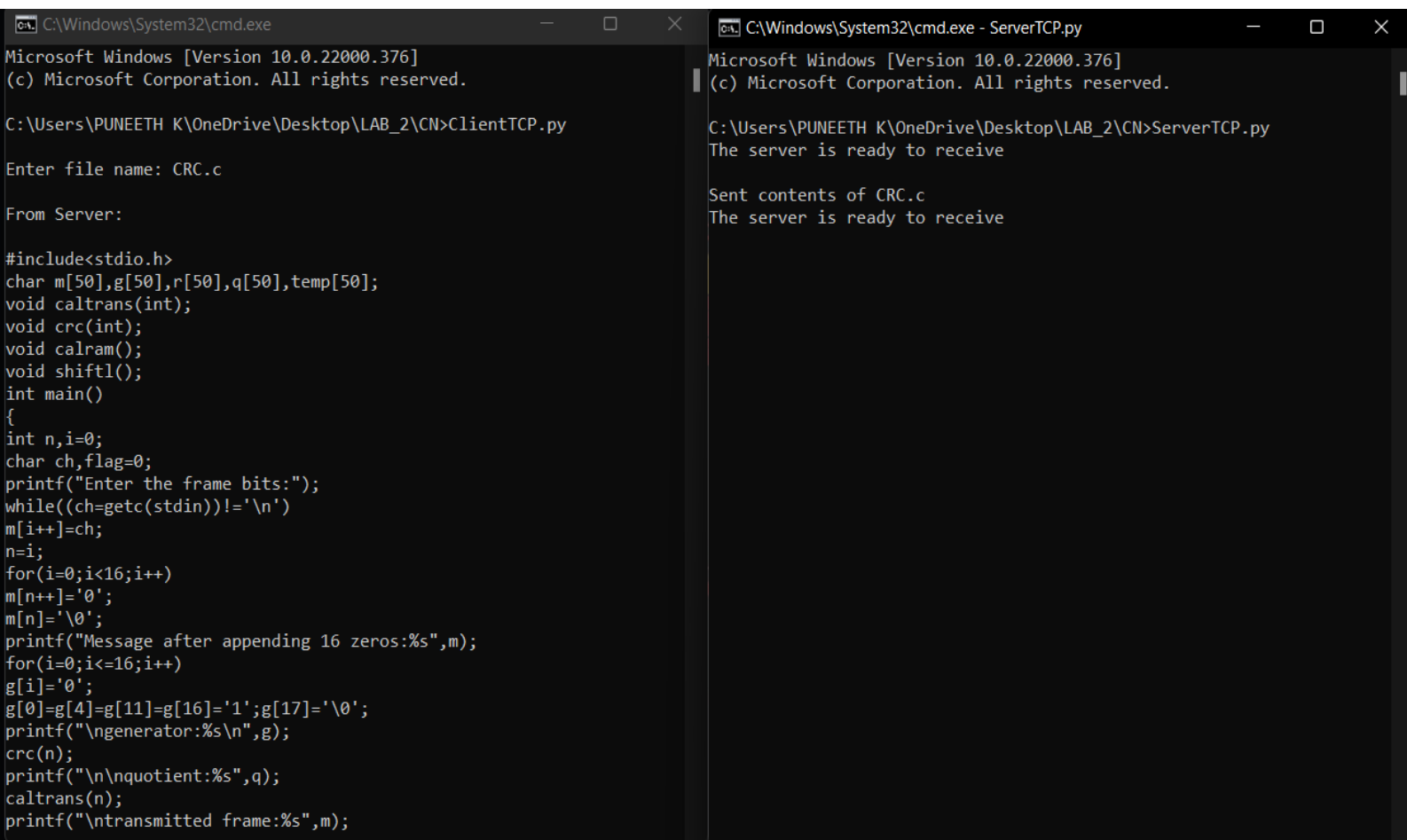
```
filecontents = clientSocket.recv(1024).decode()
```

```
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"))
```

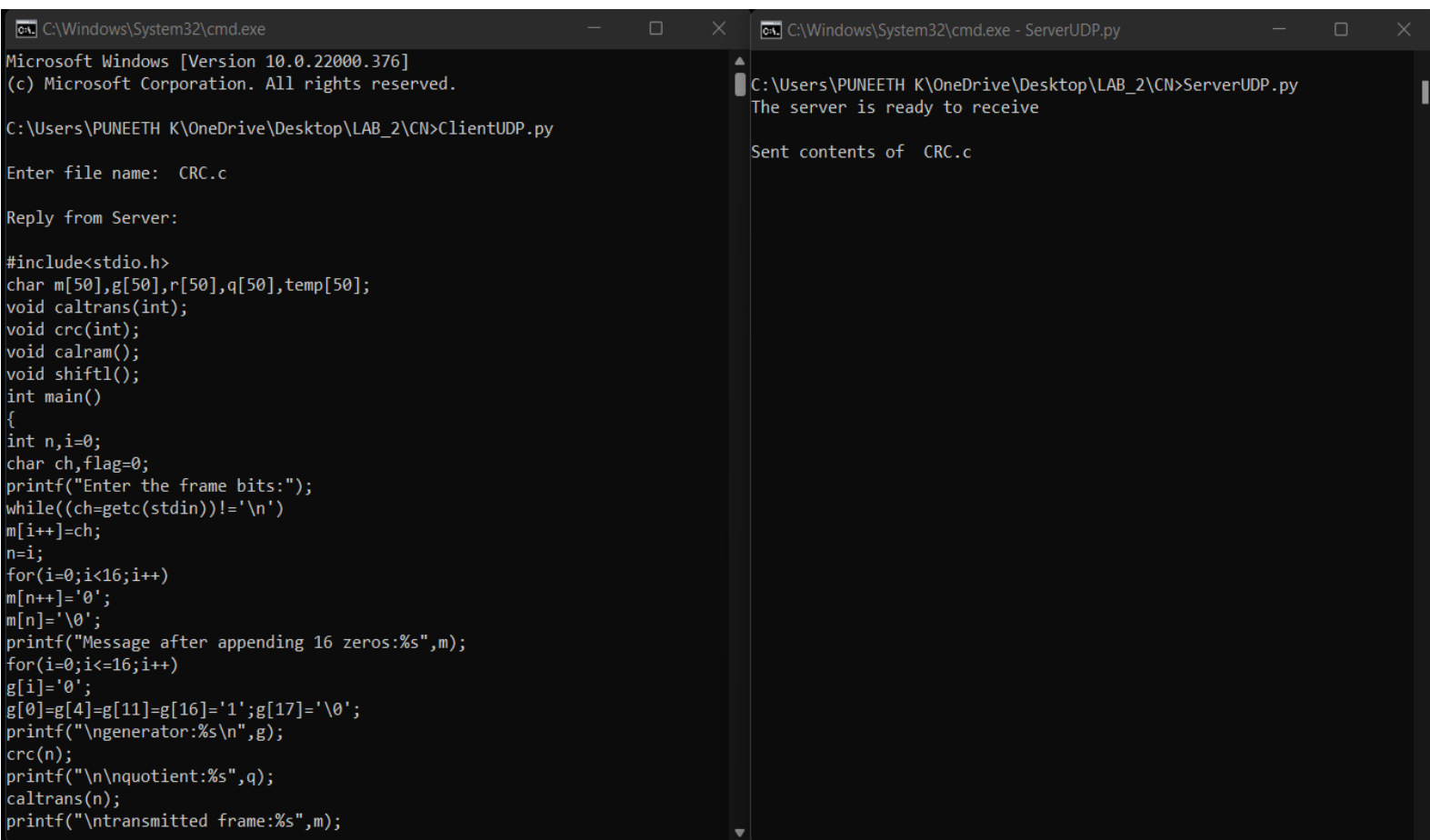
```
# for i in filecontents:
```

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

```
clientSocket.close()
```

```
clientSocket.close()
```

OUTPUT



The screenshot shows two side-by-side Windows command prompt windows. 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
```

