CN LAB PROGRAMS

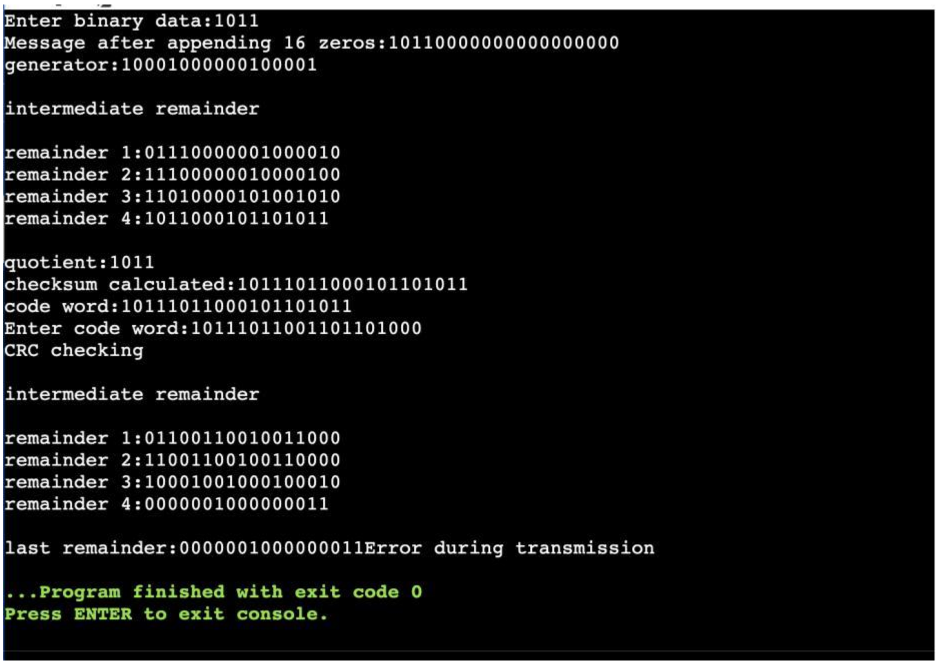
1. Write a program for error detecting code using CRC-CCITT (16-bits).

PROGRAM:

```
#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 binary data:");
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("\nchecksum calculated:%s",m);
printf("\ncode word:%s",m);
printf("\nEnter code word:");
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\nNo error in message");
}
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';
shiftl();
}
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 shiftl()
{
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';
}
```

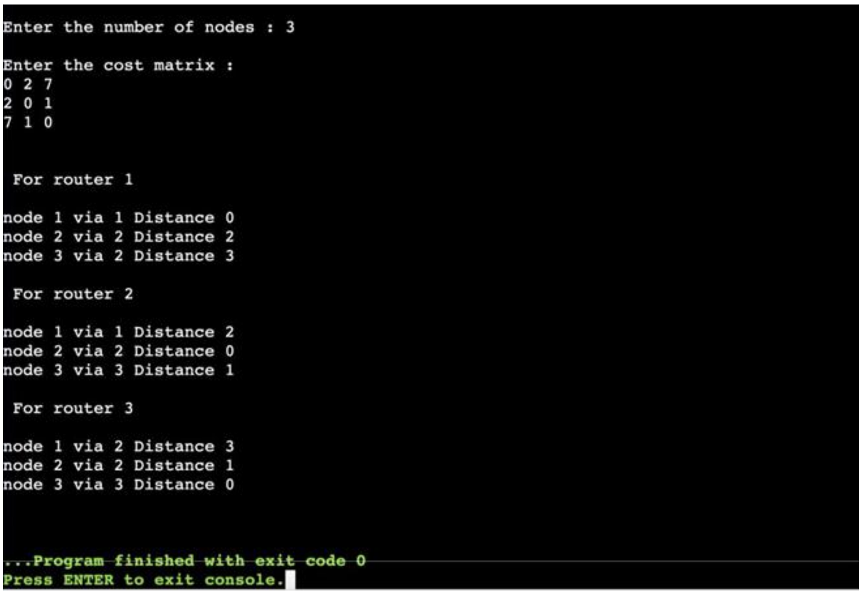


```
Enter binary data:1011
Message after appending 16 zeros:10110000000000000000
generator:10001000000100001

intermediate remainder

remainder 1:01110000001000010
remainder 2:11100000010000100
remainder 3:11010000101001010
remainder 4:1011000101101011

quotient:1011
checksum calculated:10111011000101101011
code word:10111011000101101011
Enter code word:10111011001101101000
CRC checking

intermediate remainder

remainder 1:01100110010011000
remainder 2:11001100100110000
remainder 3:10001001000100010
remainder 4:0000001000000011

last remainder:0000001000000011Error during transmission

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

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

PROGRAM:

```
#include<stdio.h>
struct node
{
unsigned dist[20]; unsigned from[20];
}rt[10];
int main()
{
int costmat[20][20];
int nodes,i,j,k,count=0;
printf("\nEnter the number of nodes : ");
scanf("%d",&nodes);
//Enter the nodes printf("\nEnter the cost matrix :\n"); for(i=0;i<nodes;i++)
{
for(j=0;j<nodes;j++)
{
scanf("%d",&costmat[i][j]);
costmat[i][i]=0;
rt[i].dist[j]=costmat[i][j];
matrix rt[i].from[j]=j;
}
}
do
{
count=0;
for(i=0;i<nodes;i++)
for(j=0;j<nodes;j++)
for(k=0;k<nodes;k++)
if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
{//We calculate the minimum distance
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]; rt[i].from[j]=k;
count++;
}
}
while(count!=0);
for(i=0;i<nodes;i++)
{
printf("\n\n For router %d\n",i+1);
for(j=0;j<nodes;j++)
{
printf("\t\nnode %d via %d Distance %d",j+1,rt[i].from[j]+1,rt[i].dist[j]); }
}
printf("\n\n"); getch();
}
```



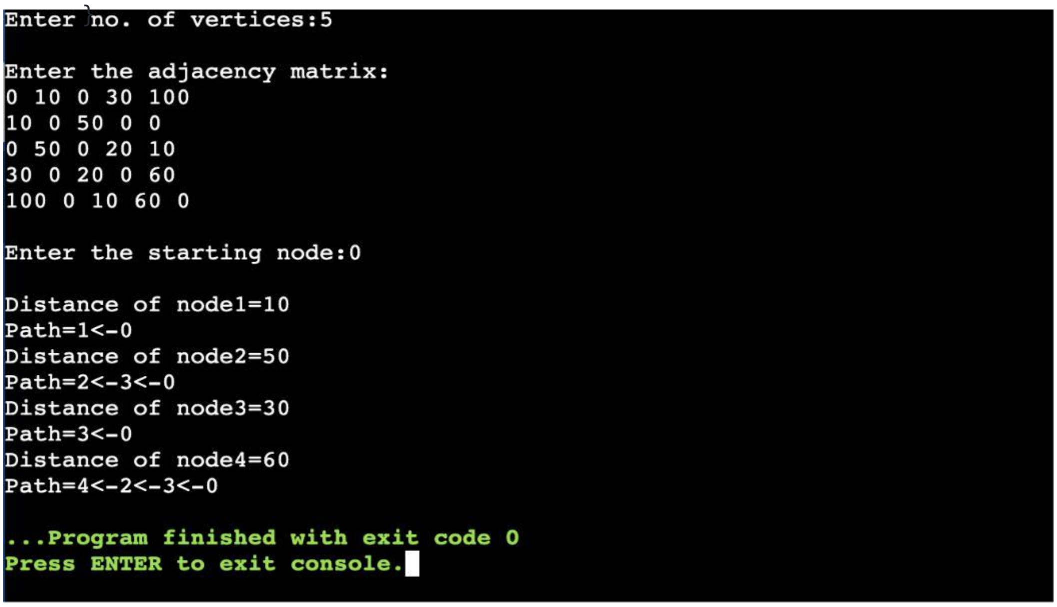
```
Enter the number of nodes : 3

Enter the cost matrix :
0 2 7
2 0 1
7 1 0


 For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 2 Distance 3

 For router 2

node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1

 For router 3

node 1 via 2 Distance 3
node 2 via 2 Distance 1
node 3 via 3 Distance 0


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

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

PROGRAM:

```
#include<conio.h>
```

```
#include<stdio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
int G[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&G[i][j]);
printf("\nEnter the starting node:");
scanf("%d",&u);
dijkstra(G,n,u);
return 0;
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{

int cost[MAX][MAX],distance[MAX],pred[MAX];
int visited[MAX],count,mindistance,nextnode,i,j;
//pred[] stores the predecessor of each node
//count gives the number of nodes seen so far
//create the cost matrix
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(G[i][j]==0)
cost[i][j]=INFINITY;
else
cost[i][j]=G[i][j];
//initialize pred[],distance[] and visited[]
for(i=0;i<n;i++)
{
distance[i]=cost[startnode][i];
pred[i]=startnode;
visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
count=1;
while(count<n-1)
{
mindistance=INFINITY;
//nextnode gives the node at minimum distance
for(i=0;i<n;i++)
if(distance[i]<mindistance&&!visited[i])
{
mindistance=distance[i];
nextnode=i;
}
//check if a better path exists through nextnode
visited[nextnode]=1;
for(i=0;i<n;i++)
if(!visited[i])
if(mindistance+cost[nextnode][i]<distance[i])
{
distance[i]=mindistance+cost[nextnode][i];
pred[i]=nextnode;
}
count++;
}

//print the path and distance of each node
for(i=0;i<n;i++)
if(i!=startnode)
{
printf("\nDistance of node%d=%d",i,distance[i]);
printf("\nPath=%d",i);
j=i;
do
{
j=pred[j];
printf("<-%d",j);
}while(j!=startnode);
}
}
```

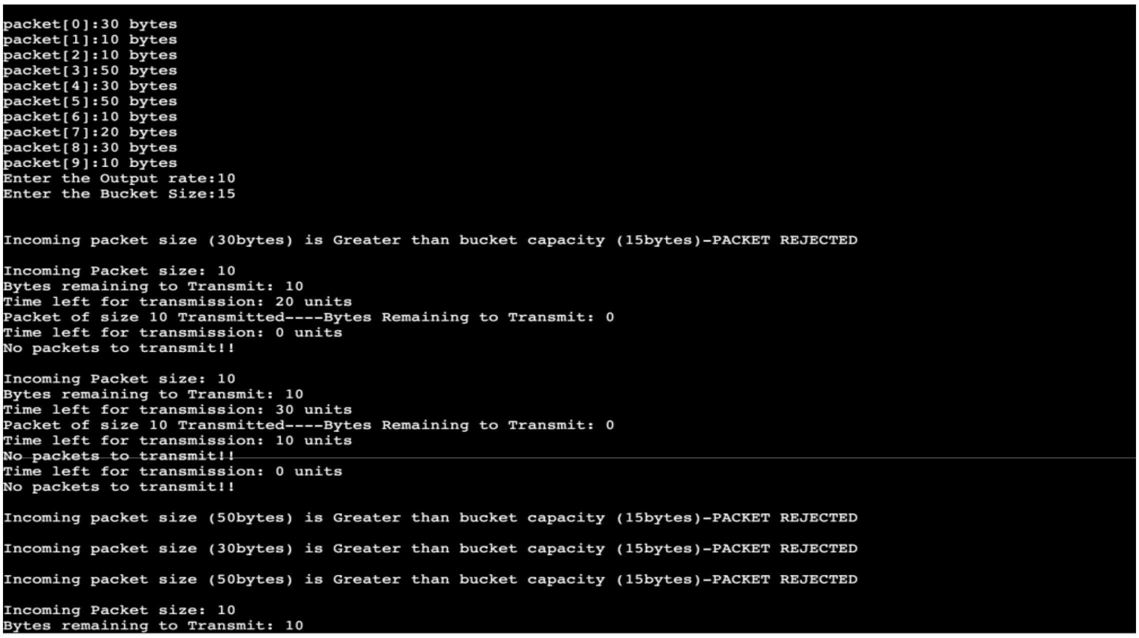


```
Enter no. of vertices:5

Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node1=10
Path=1<-0
Distance of node2=50
Path=2<-3<-0
Distance of node3=30
Path=3<-0
Distance of node4=60
Path=4<-2<-3<-0

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

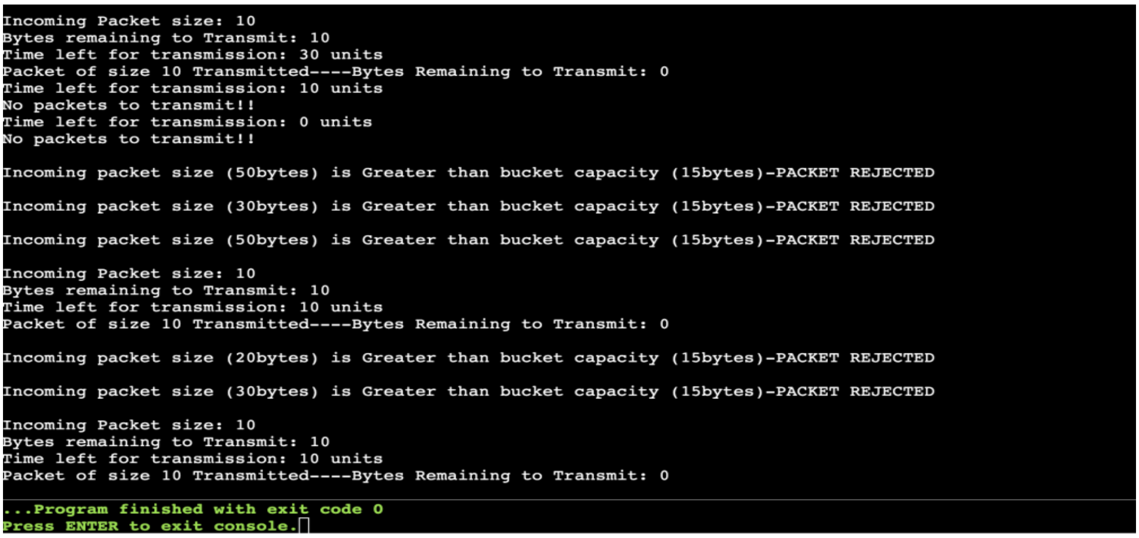
4. Write a program for congestion control using Leaky bucket algorithm.

PROGRAM:

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

#define NOF_PACKETS 10

int rand(int a)
{
int rn = (random() % 10) % a;
return rn == 0 ? 1 : rn;
}
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(6) * 10;
  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 = rand(4) * 10;
printf("\nTime left for transmission: %d units", p_time);
  for(clk = 10; clk <= p_time; clk += 10)
{
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("\nTime left for transmission: %d units", p_time-clk);
printf("\nNo packets to transmit!!"); }
}
}
}
       }
```



```
packet[0]:30 bytes
packet[1]:10 bytes
packet[2]:10 bytes
packet[3]:50 bytes
packet[4]:30 bytes
packet[5]:50 bytes
packet[6]:10 bytes
packet[7]:20 bytes
packet[8]:30 bytes
packet[9]:10 bytes
Enter the Output rate:10
Enter the Bucket Size:15


Incoming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 20 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 30 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!

Incoming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming Packet size: 10
Bytes remaining to Transmit: 10
```



```
Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 30 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 10 units
No packets to transmit!!
Time left for transmission: 0 units
No packets to transmit!!

Incoming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming packet size (50bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0

Incoming packet size (20bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming packet size (30bytes) is Greater than bucket capacity (15bytes)-PACKET REJECTED

Incoming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0

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

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.

PROGRAM:

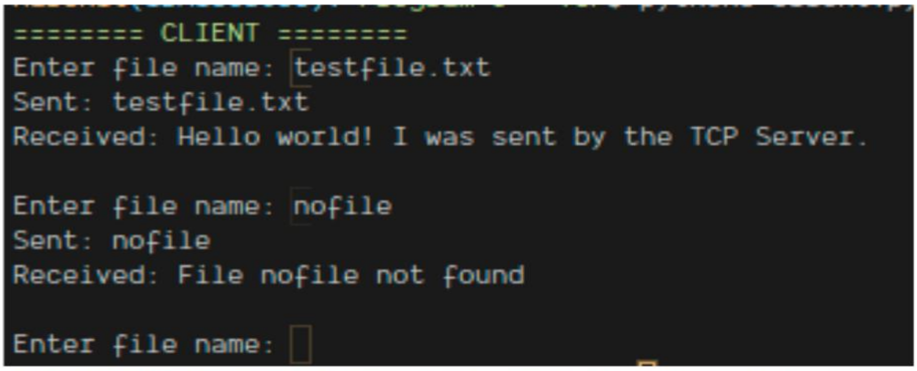
Client.py

```
import socket
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 65432
print('\033[32m======== CLIENT ========\033[0m')
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.connect((SERVER_HOST, SERVER_PORT))

    while True:

        filename = input('Enter file name: ')
        if not filename:
            break


        sock.sendall(bytes(filename, 'utf-8'))

        print(f'Sent: {filename}')
        data = sock.recv(1024)
        contents = data.decode('utf-8')
        print(f'Received: {contents}')
        print()
```

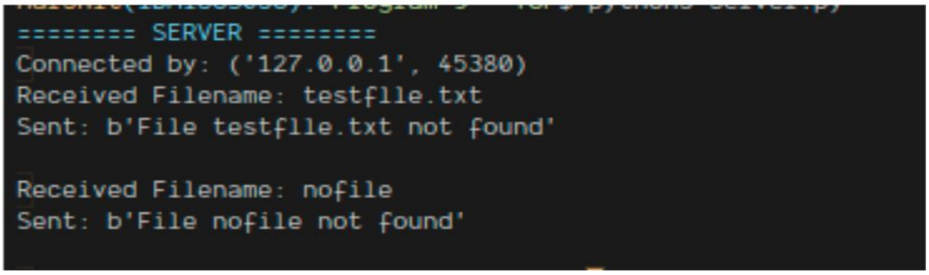
Server.py

```
import socket
HOST = '127.0.0.1'
PORT = 65432
print('\033[36m======== SERVER ========\033[0m')
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.bind((HOST, PORT))
    sock.listen(1)
    conn, addr = sock.accept()
    with conn:
        print(f'Connected by: {addr}')
```

```
while True:

    data = conn.recv(1024)

    if not data:

        break
    filename = data.decode('utf-8')
    print(f'Received Filename: {filename}')
    try:
        with open(filename, 'r') as f:

            data = f.read()

        data = bytes(data, 'utf-8')

    except:

        data = bytes(f'File {filename} not found', 'utf-8')
    conn.sendall(data)
    print(f'Sent: {data}')
    print()
```



```
======== CLIENT ========
Enter file name: testfile.txt
Sent: testfile.txt
Received: Hello world! I was sent by the TCP Server.

Enter file name: nofile
Sent: nofile
Received: File nofile not found

Enter file name:
```



```
======== SERVER ========
Connected by: ('127.0.0.1', 45380)
Received Filename: testflle.txt
Sent: b'File testflle.txt not found'

Received Filename: nofile
Sent: b'File nofile not found'
```

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.

PROGRAM:

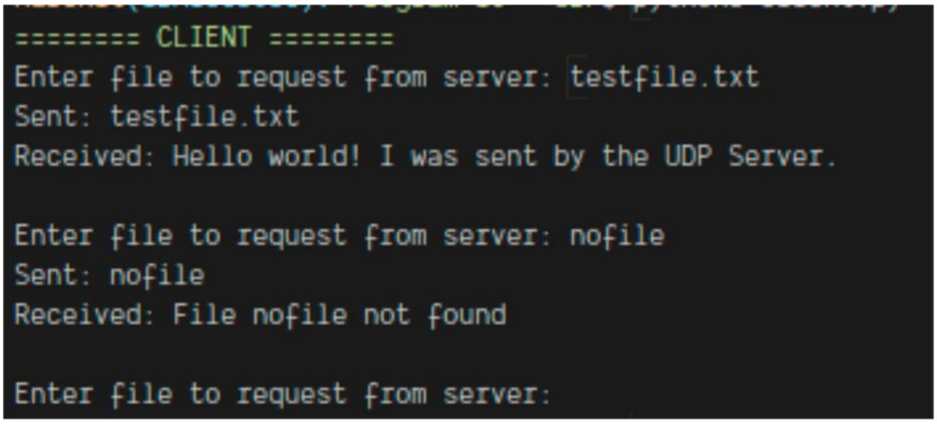
Client.py

```
import socket
HOST = '127.0.0.1'
PORT = 65432
print('\033[32m======== CLIENT ========\033[0m')
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
    sock.connect((HOST, PORT))
    while True:

        filename = input('Enter file to request from server: ')
        if not filename:

            break
        sock.sendall(bytes(filename, 'utf-8'))
        print(f'Sent: {filename}')
        data = sock.recv(1024).decode('utf-8')

        print(f'Received: {data}')
        print()
```

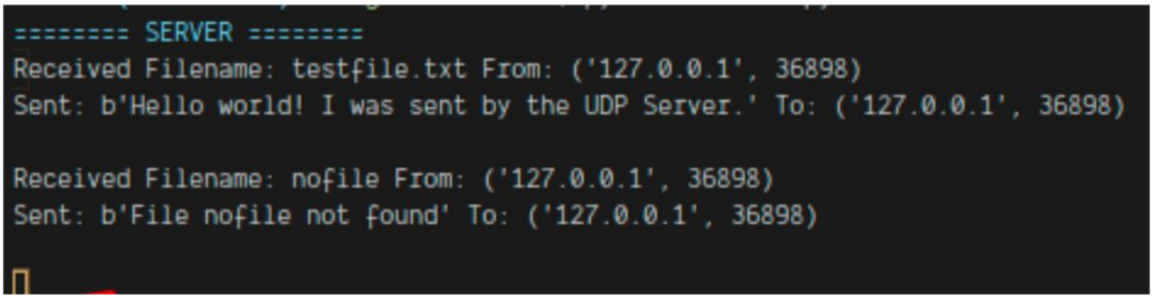
Server.py

```
import socket
HOST = '127.0.0.1'
PORT = 65432
print('\033[36m======== SERVER ========\033[0m')
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
    sock.bind((HOST, PORT))
    while True:
        data, addr = sock.recvfrom(1024)
        if not data:

            break
        filename = data.decode('utf-8')
        print(f'Received Filename: {filename} From: {addr}')
        try:


            with open(filename, 'r') as f:
```

```
        data = f.read()
    data = bytes(data, 'utf-8')
except:
    data = bytes(f'File {filename} not found', 'utf-8')
sock.sendto(data, addr)
print(f'Sent: {data} To: {addr}')
print()
```



```
======== CLIENT ========
Enter file to request from server: testfile.txt
Sent: testfile.txt
Received: Hello world! I was sent by the UDP Server.

Enter file to request from server: nofile
Sent: nofile
Received: File nofile not found

Enter file to request from server:
```



```
======== SERVER ========
Received Filename: testfile.txt From: ('127.0.0.1', 36898)
Sent: b'Hello world! I was sent by the UDP Server.' To: ('127.0.0.1', 36898)

Received Filename: nofile From: ('127.0.0.1', 36898)
Sent: b'File nofile not found' To: ('127.0.0.1', 36898)
```