**Cycle 2**

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

#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 freme:");  
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 freme 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';  
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';  
}

Output





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

#include<stdio.h>  
  
int main(){  
    int incoming, outgoing, buck\_size, n, store = 0;  
    printf("Enter bucket size, outgoing rate and no of inputs: ");  
    scanf("%d %d %d", &buck\_size, &outgoing, &n);  
  
    while (n != 0) {  
        printf("Enter the incoming packet size : ");  
        scanf("%d", &incoming);  
        printf("Incoming packet size %d\n", incoming);  
        if (incoming <= (buck\_size - store)){  
            store += incoming;  
            printf("Bucket buffer size %d out of %d\n", store, buck\_size);  
        } else {  
            printf("Dropped %d no of packets\n", incoming - (buck\_size - store));  
            printf("Bucket buffer size %d out of %d\n", store, buck\_size);  
            store = buck\_size;  
        }  
        store = store - outgoing;  
        printf("After outgoing %d packets left out of %d in buffer\n", store, buck\_size);  
        n--;  
    }  
}

Output



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

**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()

**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)

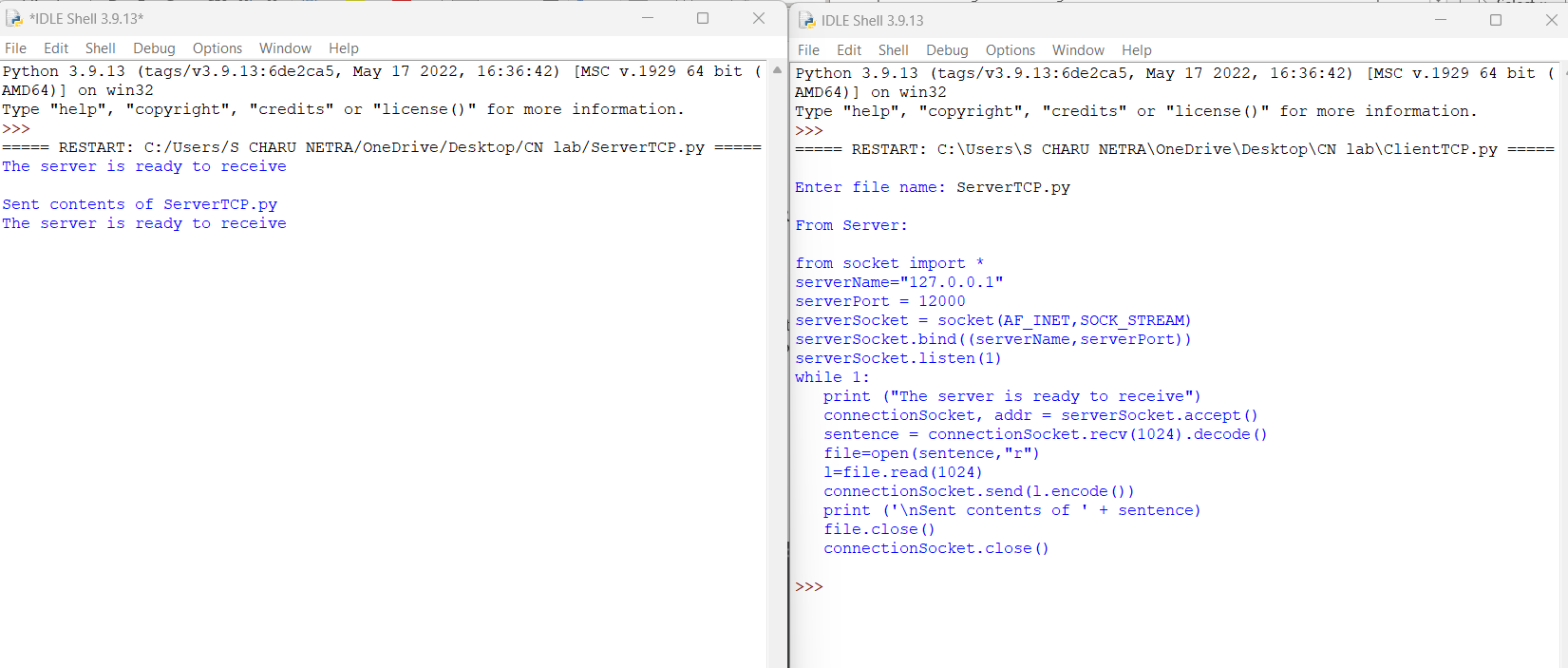
connectionSocket.send(l.encode())

print ('\nSent contents of ' + sentence)

file.close()

connectionSocket.close()

Output



4. 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.

**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")

con=file.read(2048)

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

print ('\nSent contents of ', end = ' ')

print (sentence)

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"))

clientSocket.close()

clientSocket.close()

Output

