```
crc.cpp
=============
#include<iostream>
#include<string.h>
using namespace std;
#define N strlen(g)

char d[32],cs[32],d1[32],cs1[32],g[35];
int n,i,k,error;
int codecheck();
void xor0(){
    for(k = 1;k < N; k++)
    cs[k] = (( cs[k] == g[k])?'0':'1');
//return 0;
}
void xor1(){
    for(k = 1;k < N; k++)
    cs1[k] = (( cs1[k] == g[k])?'0':'1');
}

void crc(){
    for(i=0;i<N;i++)
        cs[i]=d[i];
    do{
        if(cs[0]=='1')
            xor0();
        for(k=0;k<N-1;k++)
            cs[k]=cs[k+1];
        cs[k]=d[i++];
    }while(i<=n+N-1);
}
void crc1(){
    for(i=0;i<N;i++)
        cs1[i]=d1[i];
    do{
        if(cs1[0]=='1')
            xor1();
        for(k=0;k<N-1;k++)
            cs1[k]=cs1[k+1];
        cs1[k]=d1[i++];
    }while(i<=n+N-1);
}
int codecheck(){
for(i=0;i<strlen(cs1);i++){
if(cs1[i]!='0'){
        return 0;
}
}
return 1;

}
int main()
{
    cout<<"\nEnter data : ";
    cin>>d;
    cout<<"\nEnter Generatng polynomial :";
    cin>>g;
    n=strlen(d);
    for(i=n;i<n+N-1;i++)
        d[i]='0';
    crc();
    cout<<"\nChecksum is : "<<cs;
    for(i=n;i<n+N-1;i++)
        d[i]=cs[i-n];
    cout<<"\nFinal codeword sent  is : "<<d;
    cout<<"\nEnter data recieved: ";
    cin>>d1;
    for(i=n;i<n+N-1;i++)
        d1[i]=cs[i-n];
    crc1();
if(codecheck())
        cout<<"\ncorrect data recieved\n";
    else
        {for(i=0;i<n;i++){
        if(d[i]!=d1[i]){
        error=i;
        break;
        }
        }
        cout<<"error at"<<error+1<<endl;
        return 0;
}
}

==========================================
client.c
====
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
    int soc, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;

    /*  socket creates an endpoint for communication and returns a file descriptor */
    /*soc is our socket descriptor for later use returns -1 if we have a error
    int socket(int domain,int type,nt protocol)*/
    soc = socket(PF_INET, SOCK_STREAM, 0);//protocol family

    /*
     * sockaddr_in is used for ip manipulation
     * we define the port and IP for the connection.
     */
    addr.sin_family = AF_INET;//address family
    addr.sin_port = htons(7891);//htons() function converts the unsigned sort integer hostshort from
host byte order to network byte order
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");//LINE I GO WRONG

    /*  keep trying to esatablish connection with server */
    while(connect(soc, (struct sockaddr *) &addr, sizeof(addr))) ;
    printf("\nClient is connected to Server");
    printf("\nEnter file name: ");
    scanf("%s", fname);
    /*  send the filename to the server */
    send(soc, fname, sizeof(fname), 0);

    printf("\nRecieved response\n");
    /*  keep printing any data received from the server */
    while ((n = recv(soc, buffer, sizeof(buffer), 0)) > 0)
        printf("%s", buffer);

    return 0;
}

====================
server.c
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
    int welcome, new_soc, fd, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;

    welcome = socket(PF_INET, SOCK_STREAM, 0);

    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    bind(welcome, (struct sockaddr *) &addr, sizeof(addr));
    printf("\nServer is Online");
    /*  listen for connections from the socket */
    listen(welcome, 5);
    /*  accept a connection, we get a file descriptor */
    new_soc = accept(welcome, NULL, NULL);

    /*  receive the filename */
    recv(new_soc, fname, 50, 0);
    printf("\nRequesting for file: %s\n", fname);

    /*  open the file and send its contents */
    fd = open(fname, O_RDONLY);

    if (fd < 0)
        send(new_soc, "\nFile not found\n", 15, 0);
    else
        while ((n = read(fd, buffer, sizeof(buffer))) > 0)
            send(new_soc, buffer, n, 0);
    printf("\nRequest sent\n");
    close(fd);

    return 0;
}

==================================
distance.c
================
#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];//initialise the distance equal to cost matrix
            rt[i].from[j]=j;
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance
from the node i to k using the cost matrix
            //and add the distance from k to node j
            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");

}

===========================

#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--;
    }
}
===================================
```