```
#include<stdio.h>
#include<string.h>

char data[100], concatdata[117], src_crc[17], dest_crc[17], frame[120],
divident[18];
char divisor[18];
char res[17] = "0000000000000000";

void crc_cal(int node) {
    int i, j;

    for (j = 17; j <= strlen(concatdata); j++) {
        if (divident[0] == '1') {
            for (i = 1; i <= 16; i++)
                if (divident[i] != divisor[i])
                    divident[i - 1] = '1';
                else
                    divident[i - 1] = '0';
        } else {
            for (i = 1; i <= 16; i++)
                divident[i - 1] = divident[i];
        }

        if (node == 0)
            divident[i - 1] = concatdata[j];
        else
            divident[i - 1] = frame[j];
    }

    divident[i] = '\0';
    printf("\ncrc is %s\n", divident);

    if (node == 0) {
        strcpy(src_crc, divident);
    } else {
        strcpy(dest_crc, divident);
    }
}

int main() {
    int i;

    printf("enter the generator bits\n");
    gets(divisor);

    if (strlen(divisor) < 17 || strlen(divisor) > 17) {
        printf("please enter the generator length minimum of 17 bits\n");
        exit(0);
    }
```

```
    printf("\n At src node :\n Enter the msg to be sent :");
    gets(data);

    strcpy(concatdata, data);
    strcat(concatdata, "0000000000000000");

    for (i = 0; i <= 16; i++)
        divident[i] = concatdata[i];

    divident[i] = '\0';
    crc_cal(0);

    printf("\ndata is:\t");
    puts(data);

    printf("\n The frame transmitted is :\t");
    printf("\n%s%s", data, src_crc);

    printf("\n\t\tSOURCE NODE TRANSMITTED THE FRAME---->");

    printf("\n\n\n\n\t\t\tAT DESTINATION NODE\nenter the received frame:\t");
    gets(frame);

    for (i = 0; i <= 16; i++)
        divident[i] = frame[i];

    divident[i] = '\0';
    crc_cal(1);

    if ((strcmp(dest_crc, res)) == 0)
        printf("\nReceived frame is error-free.\n ");
    else
        printf("\nReceived frame contains one or more errors.\n");

    return 1;
}
```

```
================================================================================
```

```
#include<stdio.h>

struct rtable {
    int dist[20], nextnode[20];
} table[20];

int cost[10][10], n;

void distvector() {
    int i, j, k, count;
```

```
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            table[i].dist[j] = cost[i][j];

    do {
        count = 0;
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                for (k = 0; k < n; k++) {
                    if (table[i].dist[j] > cost[i][k] + table[k].dist[j]) {
                        table[i].dist[j] = table[i].dist[k] + table[k].dist[j];
                        table[i].nextnode[j] = k;
                        count++;
                    }
                }
            }
        }
    } while (count != 0);
}

int main() {
    int i, j;

    printf("\nEnter the number of vertices: ");
    scanf("%d", &n);

    printf("\nEnter the cost matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &cost[i][j]);

    distvector();

    for (i = 0; i < n; i++) {
        printf("\nRouting table for router %c\n", i + 'A');
        printf("\nDestNode\tNextNode\tDistance\n");

        for (j = 0; j < n; j++) {
            if (table[i].dist[j] == 99)
                printf("%c\t\t-\t\tinfinite\n", j + 'A');
            else
                printf("%c\t\t%c\t\t%d\n", j + 'A', table[i].nextnode[j] + 'A',
table[i].dist[j]);
        }
    }

    return 0;
}
==========================================33333333aaaaaa==========================
====================================
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/fcntl.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int fd, sockfd, newsockfd, clilen, portno, n;
    struct sockaddr_in seradd, cliadd;
    char buffer[4096];

    if (argc < 2) {
        fprintf(stderr, "\n\n No port\n");
        exit(1);
    }

    portno = atoi(argv[1]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    if (sockfd < 0)
        error("\n error opening socket.\n");

    bzero((char *)&seradd, sizeof(seradd));
    seradd.sin_family = AF_INET;
    seradd.sin_addr.s_addr = (htonl)INADDR_ANY;
    seradd.sin_port = htons(portno);

    if (bind(sockfd, (struct sockaddr *)&seradd, sizeof(seradd)) < 0)
        perror("\n IP addr can't bind");

    listen(sockfd, 5);
    clilen = sizeof(cliadd);

    printf("\n Server waiting for client request\n");

    while (1) {
        newsockfd = accept(sockfd, (struct sockaddr *)&cliadd, &clilen);

        if (newsockfd < 0)
            perror("\n Server cannot accept connection request ");

        bzero(buffer, 4096);
        read(newsockfd, buffer, 4096);
        fd = open(buffer, O_RDONLY);

        if (fd < 0)
            perror("\n File does not exist");
```

```
        while (1) {
            n = read(fd, buffer, 4096);

            if (n <= 0)
                exit(0);

            write(newsockfd, buffer, n);
            printf("\n File transfer complete\n");
        }

        close(fd);
        close(newsockfd);
    }

    return 0;
}
--------------------------------------3333333bbbbbb-------------------------------
----

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

int main(int argc, char *argv[]) {
    int sockfd, portno, n;
    struct sockaddr_in seradd;
    char buffer[4096], *serip;

    if (argc < 4) {
        fprintf(stderr, "usage %s serverip filename port", argv[0]);
        exit(0);
    }

    serip = argv[1];
    portno = atoi(argv[3]);

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        perror("\n Error in creating socket.\n");

    perror("\n Client on line.");

    bzero((char *)&seradd, sizeof(seradd));
    seradd.sin_family = AF_INET;
    seradd.sin_addr.s_addr = inet_addr(serip);
```

```
    seradd.sin_port = htons(portno);

    if (connect(sockfd, (struct sockaddr *)&seradd, sizeof(seradd)) < 0)
        perror("\n Error in connection setup \n");

    write(sockfd, argv[2], strlen(argv[2]) + 1);

    bzero(buffer, 4096);
    n = read(sockfd, buffer, 4096);

    if (n <= 0) {
        perror("\n File not found");
        exit(0);
    }

    write(1, buffer, n);

    return 0;
}
==================================444444bbbbbb====================================
==============

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"

int main() {
    char p[100], c[5000], ch;
    int num, fd, fd2, f1;

    mknod(FIFO1, S_IFIFO | 0666, 0);
    mknod(FIFO2, S_IFIFO | 0666, 0);

    printf("\n Server online...\n");
    fd = open(FIFO1, O_RDONLY);
    fd2 = open(FIFO2, O_WRONLY);

    printf("Server online\n waiting for client \n\n");

    if ((num = read(fd, p, 100)) == -1)
        perror("\n Read Error ");
    else {
        p[num] = '\0';
```

```
        printf("\n File is %s .\n", p);

        if ((f1 = open(p, O_RDONLY)) < 0) {
            write(fd2, "File not found", 15);
            return 1;
        } else {
            stdin = fdopen(f1, "r");
            num = 0;

            while ((ch = fgetc(stdin)) != EOF)
                c[num++] = ch;

            c[num] = 0;
            printf(" Server: Transfering the contents of :%s ", p);

            if ((num = write(fd2, c, strlen(c))) == -1)
                printf("\n Error in writting to FIFO\n");
            else
                printf("\n File transfer completed \n");
        }
    }

    return 0;
}
-----------------------------------44444444bbbbbb---------------------------------
-------

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"

int main() {
    char p[100], c[5000];
    int num, fd, fd2, f1;

    mknod(FIFO1, S_IFIFO | 0666, 0);
    mknod(FIFO2, S_IFIFO | 0666, 0);

    printf("\n Client online...\n");
    fd = open(FIFO1, O_WRONLY);
    fd2 = open(FIFO2, O_RDONLY);

    printf("Client : Enter the filename . \n\n");
```

```
    scanf("%s", p);

    num = write(fd, p, strlen(p));

    if (num == -1) {
        perror("\nWrite Error.\n");
        return 1;
    } else {
        printf("\n Waiting for reply\n");

        if ((num = read(fd2, c, 5000)) == -1)
            perror("\nError while reading from fifo \n");
        else {
            c[num] = 0;
            printf("%s", c);
        }
    }

    return 1;
}
```

```
================================================================================
```

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int main (int argc, char *argv[]) {
    bool tracing = true;
    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

    NodeContainer nodes;
    nodes.Create (2);
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
    pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

    NetDeviceContainer devices = pointToPoint.Install (nodes);
    InternetStackHelper stack;
    stack.Install (nodes);

    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
```

```
    Ipv4InterfaceContainer interfaces = address.Assign (devices);

    UdpEchoServerHelper echoServer (9);
    ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
    serverApps.Start (Seconds (1.0));
    serverApps.Stop (Seconds (10.0));

    UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
    echoClient.SetAttribute ("MaxPackets", UintegerValue (4));
    echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
    echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
    clientApps.Start (Seconds (2.0));
    clientApps.Stop (Seconds (10.0));

    AnimationInterface anim("1.xml");
    anim.SetConstantPosition(nodes.Get (0), 10.0, 10.0);
    anim.SetConstantPosition(nodes.Get (1), 20.0, 30.0);

    if (tracing == true) {
        pointToPoint.EnablePcapAll("p2p");
    }

    Simulator::Run ();
    Simulator::Destroy ();

    return 0;
}
-=-=-=---=--=-=-=-=-=-=----=--=---=-=-=-=-=-=-=-=-=-=-=--=-=-=-=-=-=-=-=-
=-==-=-
```