

**Name :- HRITIK SINGH**

**USN :- 1BM19CS063**

**SECTION :- 5-B**

**BATCH :- 1**

## **CN LAB CYCLE 2 PROGRAMS, INPUT AND OUTPUT**

### **LAB PROGRAM 7 :**

#### **QUESTION :**

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

#### **INPUT :**

```
#include
<iostream>

#include <string.h>
using namespace std;
int crc(char *ip, char *op, char *poly, int mode)
{
    strcpy(op, ip);
    if (mode) {
        for (int i = 1; i < strlen(poly); i++)
            strcat(op, "0");
    }
    for (int i = 0; i < strlen(ip); i++) {
        if (op[i] == '1') {
            for (int j = 0; j < strlen(poly); j++) {
                if (op[i + j] == poly[j])
                    op[i + j] = '0';
                else
                    op[i + j] = '1';
            }
        }
    }
    for (int i = 0; i < strlen(op); i++)
```

```

        if (op[i] == '1')
            return 0;
        return 1;
    }

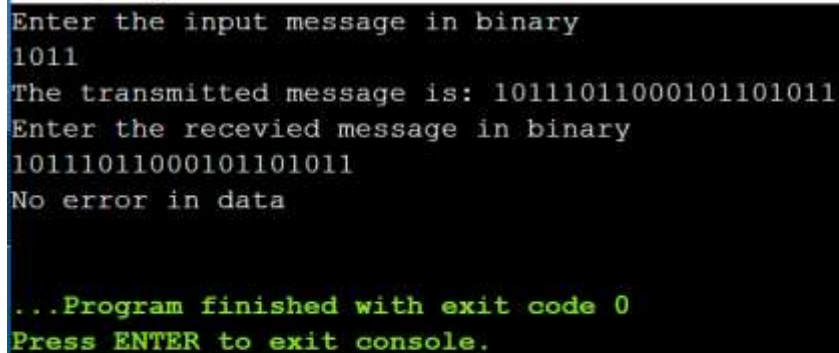
    int main()
    {
        char ip[50], op[50], recv[50];
        char poly[] = "10001000000100001";

        cout << "Enter the input message in binary : "<< endl;
        cin >> ip;
        crc(ip, op, poly, 1);
        cout << "The transmitted message is : " << ip << op + strlen(ip) <<
endl;
        cout << "Enter the received message in binary : " << endl;
        cin >> recv;
        if (crc(recv, op, poly, 0))
            cout << "No error in data" << endl;
        else
            cout << "Error in data transmission has occurred" << endl;

        return 0;
    }

```

## OUTPUT :



```

Enter the input message in binary
1011
The transmitted message is: 10111011000101101011
Enter the received message in binary
10111011000101101011
No error in data

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

```

## LAB PROGRAM 8 :

### QUESTION :

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

## INPUT :

```
#include<stdio.h>

struct node
{
    unsigned dist[20];
    unsigned from[20];
}rt[10];

int main()
{
    int dmat[20][20];
    int n,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&n);
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++){
            scanf("%d",&dmat[i][j]);
            dmat[i][i]=0;
            rt[i].dist[j]=dmat[i][j];
            rt[i].from[j]=j;
        }
    do{
        count=0;
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                for(k=0;k<n;k++)

                    if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]){

                        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<n;i++)
    {
        printf("\n\nState value for router %d is \n",i+1);
        for(j=0;j<n;j++){
            printf(" \t\nnode %d via %d Distance :
%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
        }
    }
}
```

## OUTPUT :

```

Enter the number of nodes : 4

Enter the cost matrix :
0 5 99 99
5 0 3 99
99 3 0 1
99 99 1 0

State value for router 1 is

node 1 via 1 Distance : 0
node 2 via 2 Distance : 5
node 3 via 2 Distance : 8
node 4 via 2 Distance : 9

State value for router 2 is

node 1 via 1 Distance : 5
node 2 via 2 Distance : 0
node 3 via 3 Distance : 3
node 4 via 3 Distance : 4

State value for router 3 is

node 1 via 2 Distance : 8
node 2 via 2 Distance : 3
node 3 via 3 Distance : 0
node 4 via 4 Distance : 1

State value for router 4 is

node 1 via 3 Distance : 9
node 2 via 3 Distance : 4
node 3 via 3 Distance : 1
node 4 via 4 Distance : 0

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

```

## LAB PROGRAM 9 :

### QUESTION :

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

### INPUT :

```

#include<bits/stdc++.h>

#include <limits.h>
#include <stdio.h>
using namespace std;
#define V 9
int minDistance(int dist[], bool sptSet[])

```

```

{
    int min = INT_MAX, 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 printSolution(int dist[])
{
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}

void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, 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] !=
INT_MAX && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }
    printSolution(dist);
}

int main()
{
    int graph[V][V] ;
    cout<<"Enter the graph "<<endl;
    for(int i = 0; i<V; i++)
        for(int j = 0; j<V; j++)
            cin>>graph[i][j];
    dijkstra(graph, 0);
    return 0;
}

```

OUTPUT :

```

Enter the graph
0 4 0 0 0 0 0 8 0
4 0 8 0 0 0 0 11 0
0 8 0 7 0 4 0 0 2
0 0 7 0 9 14 0 0 0
0 0 0 9 0 10 0 0 0
0 0 4 14 10 0 2 0 0
0 0 0 0 0 2 0 1 6
8 11 0 0 0 0 1 0 7
0 0 2 0 0 0 6 7 0
Vertex          Distance from Source
0                0
1                4
2                12
3                19
4                21
5                11
6                9
7                8
8                14

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

```

## LAB PROGRAM 10 :

### QUESTION :

Write a program for congestion control using Leaky bucket algorithm.

### INPUT :

```

#include<stdio.h>

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

#define NOF_PACKETS 10

int rando(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] = rando(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)
                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 = rando(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)
                        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!!");
    }
}
}
}
}

```

## OUTPUT :

```

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:25

Incoming packet size (30bytes) is Greater than bucket capacity (25bytes)-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 (25bytes)-PACKET REJECTED

Incoming packet size (30bytes) is Greater than bucket capacity (25bytes)-PACKET REJECTED

Incoming packet size (50bytes) is Greater than bucket capacity (25bytes)-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: 20
Bytes remaining to Transmit: 20
Time left for transmission: 10 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 10

Incoming packet size (30bytes) is Greater than bucket capacity (25bytes)-PACKET REJECTED

Incoming Packet size: 10
Bytes remaining to Transmit: 20
Time left for transmission: 30 units
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 10
Packet of size 10 Transmitted----Bytes Remaining to Transmit: 0
Time left for transmission: 0 units
No packets to transmit!!

...Program finished with exit code 0

```



## **LAB PROGRAM 11 :**

### **QUESTION :**

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.

### **INPUT :**

#### **CLIENT TCP.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()
```

#### **SERVER TCP.py :**

```
from
socket
import
*

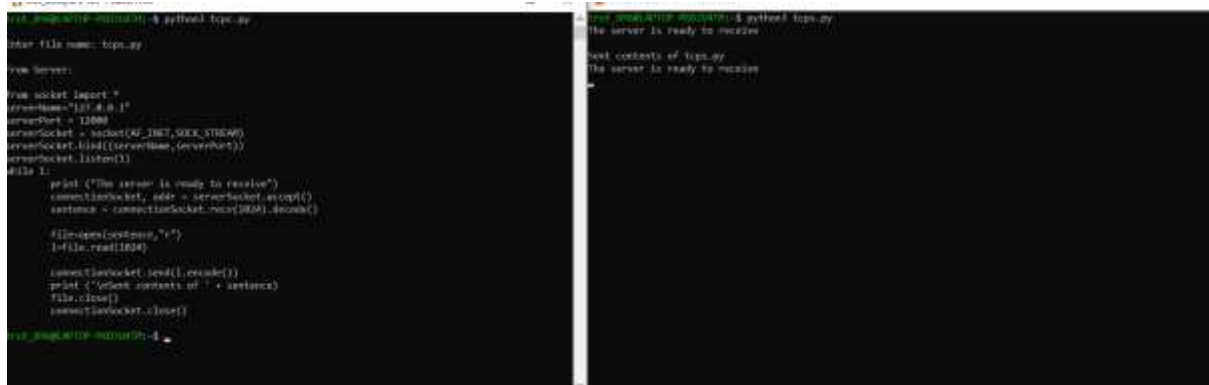
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file=open(sentence,"r")
    l=file.read(1024)

    connectionSocket.send(l.encode())
    print ('\nSent contents of ' + sentence)
```

```
file.close()
connectionSocket.close()
```

## OUTPUT :



```
root@kali:~/Python-Projects# python3 tip.py
Enter file name: tip.py

Run socket input *
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()

    #designs(sentence,"")
    #file.read(1024)

    connectionSocket.sendall.encode())
    print ("Sent contents of " + sentence)
    file.close()
    connectionSocket.close()

root@kali:~/Python-Projects#
```

```
root@kali:~/Python-Projects# python3 tip.py
The server is ready to receive
Sent contents of tip.py
The server is ready to receive
```

## LAB PROGRAM 12 :

### QUESTION :

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.

### INPUT :

#### CLIENT UDP.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()
```

#### SERVER UDP.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)
    file.close()

```

## OUTPUT :

The image shows two terminal windows side-by-side. The left window shows the execution of a Python script named 'udp.py' which acts as a server. It prints 'The server is ready to receive' and then receives a message 'udp.py' from a client. The right window shows the execution of a Python script named 'udp.py' which acts as a client. It prints 'The server is ready to receive' and then sends the contents of 'udp.py' to the server.

```

Term_Bug@ATOP-PRESENT:~$ python3 udp.py
Enter file name: udp.py
Reply from Server:
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("Enter file name: ")

clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))

(l,clientAddress,serverAddress = clientSocket.recvfrom(2048))
print ("Reply from Server:\n")
print (l.decode("utf-8"))
clientSocket.close()
clientSocket.close()

Term_Bug@ATOP-PRESENT:~$

```

```

Term_Bug@ATOP-PRESENT:~$ python3 udp.py
The server is ready to receive
Enter contents of udp.py

```

