CYCLE-2
LAB-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 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("\n generator:%s\n",g);
crc(n);
printf("\n\n quotient:%s",q);
caltrans(n);
printf("\n transmitted frame:%s",m);
printf("\nEnter transmitted frame:");
scanf("\n%s",m);
printf("CRC checking\n");
crc(n);
printf("\n\n last 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 frame 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("\n intermediate 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:

```
remainder 1:01100100001000010
remainder 2:11001000010000100
remainder 3:10000000101001010
remainder 4:00010001011010110
remainder 5:00100010110101100
remainder 6:01000101101011000
remainder 7:1000101101011000

 quotient:1011000
 transmitted frame:101110110001011010111000
Enter transmitted frame:101110110001011010111000
CRC checking

 intermediate remainder

remainder 1:01100110000011000
remainder 2:11001100000110001
remainder 3:10001000000100001
remainder 4:00000000000000000
remainder 5:00000000000000000
remainder 6:00000000000000000
remainder 7:0000000000000000

 last remainder:0000000000000000

Received frame is correct
Process returned 0 (0x0)   execution time : 23.338 s
Press any key to continue.
```

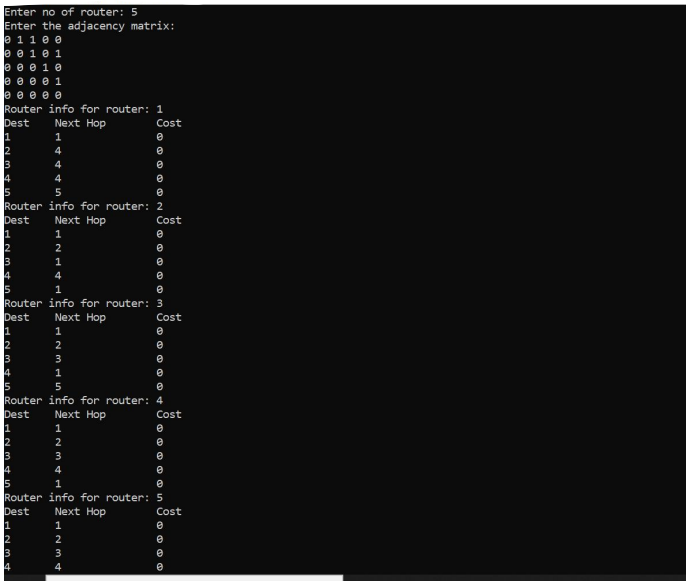
CYCLE-2
LAB-2:Write a program for distance vector algorithm to find suitable path for transmission.

Program:

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

using namespace std;

struct node
{
   int dist[20];
   int from[20];
} route[10];

int main()
{
   int dm[20][20], no;

   cout << "Enter no of router: "

      ;
   cin >> no;
   cout << "Enter the adjacency matrix:" << endl;
   for (int i = 0; i < no; i++)
   {
      for (int j = 0; j < no; j++)
      {
         cin >> dm[i][j];
         /*  Set distance from i to i as 0 */
         dm[i][i] = 0;
         route[i].dist[j] = dm[i][j];
         route[i].from[j] = j;
      }
   }

   int flag;
   do
   {
      flag = 0;
      for (int i = 0; i < no; i++)
      {
         for (int j = 0; j < no; j++)
         {
            for (int k = 0; k < no; k++)
            {
               if ((route[i].dist[j]) > (route[i].dist[k] + route[k].dist[j]))
               {
                  route[i].dist[j] = route[i].dist[k] + route[k].dist[j];
                  route[i].from[j] = k;
                  flag = 1;
               }
            }
         }
```

```
        }
    } while (flag);

    for (int i = 0; i < no; i++)
    {
        cout << "Router info for router: " << i + 1 << endl;
        cout << "Dest\tNext Hop\tCost" << endl;
        for (int j = 0; j < no; j++)
            printf("%d\t%d\t\t%d\n", j + 1, route[i].from[j] + 1, route[i].dist[j]);
    }
    return 0;
}
```

Output:



CYCLE-2
LAB-3:Implement Dijkstra's algorithm to compute the shortest path for a given topology.

Program:

```
#include <iostream>
using namespace std;
int a[30][30], source, dist[30], path[30];

void dijkstar(int a[][30], int n)
{
   int visited[n];
   for (int i = 0; i < n; i++)
   {
      dist[i] = a[source][i];
      path[i] = source;
      visited[i] = 0;
   }
   visited[source] = 1;
   for (int c = 0; c < n; c++)
   {
      int min = 999, u;
      for (int j = 0; j < n; j++)
      {
         if (dist[j] < min && visited[j] != 1)
         {
            min = dist[j];
            u = j;
         }
      }
      visited[u] = 1;
      for (int i = 0; i < n; i++)
      {
         if (min + a[u][i] < dist[i])
         {
            dist[i] = min + a[u][i];
            path[i] = u;
         }
      }
   }
}
int main()
{
   int n;
   cout << "Enter the no. of vertices :" << endl;
   cin >> n;
   cout << "Enter the adjacency matrix(Enter 9999 for infinity):" << endl;
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         cin >> a[i][j];
      }
   }
```

```
   cout << "Enter the source vertex :" << endl;
   cin >> source;
   cout << "The shortest paths from vertex ' " << source << " ' are :" << endl;
   cout << "Vertex paths" << endl;
   dijkstar(a, n);
   for (int i = 0; i < n; i++)
   {
      int k = i;
      while (k != source)
      {

         cout << k << " <- ";
         k = path[k];
      }
      cout << source << "  =  ";
      cout << "Path cost:" << dist[i] << endl;
   }
   return 0;
}
```

Output:

```
Enter the no. of vertices :
5
Enter the adjacency matrix(Enter 9999 for infinity):
0 10 9999 9999 6
9999 0 1 9999 2
9999 9999 0 5 9999
6 9999 7 0 9999
9999 3 9 2 0
Enter the source vertex :
1
The shortest paths from vertex ' 1 ' are :
Vertex paths
0 <- 3 <- 4 <- 1  =  Path cost:10
1  =  Path cost:0
2 <- 1  =  Path cost:1
3 <- 4 <- 1  =  Path cost:4
4 <- 1  =  Path cost:2

Process returned 0 (0x0)   execution time : 35.891 s
Press any key to continue.
```

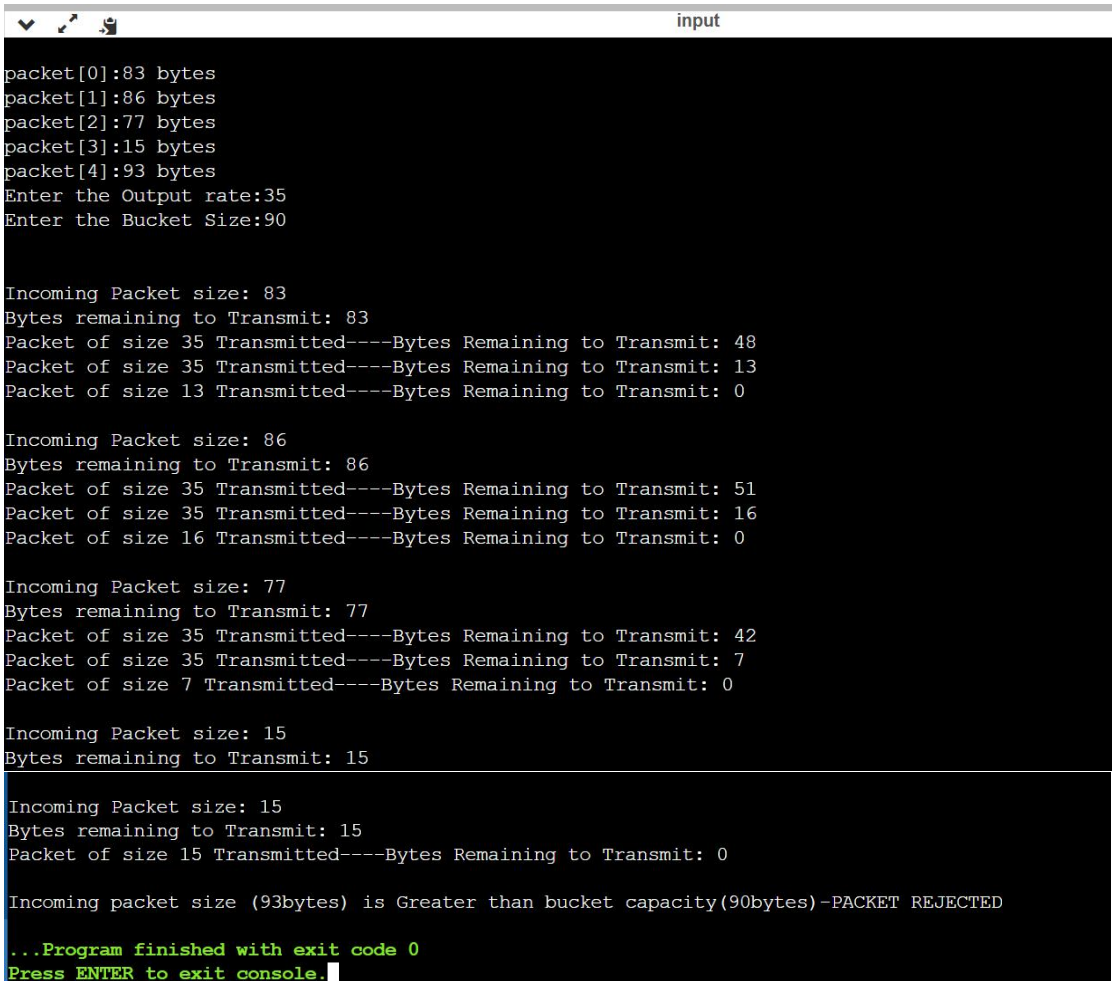
CYCLE-2

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

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define NOF_PACKETS 5
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] = random() % 100;
 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);
 while(p_sz_rm>0)
 {
 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("\nNo packets to transmit!!");
 }
 }
 }
 }
}
```

Output:


input
packet[0]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:35
Enter the Bucket Size:90
Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 48
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 13
Packet of size 13 Transmitted----Bytes Remaining to Transmit: 0
Incoming Packet size: 86
Bytes remaining to Transmit: 86
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 51
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 16
Packet of size 16 Transmitted----Bytes Remaining to Transmit: 0
Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 42
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 7
Packet of size 7 Transmitted----Bytes Remaining to Transmit: 0
Incoming Packet size: 15
Bytes remaining to Transmit: 15
Incoming Packet size: 15
Bytes remaining to Transmit: 15
Packet of size 15 Transmitted----Bytes Remaining to Transmit: 0
Incoming packet size (93bytes) is Greater than bucket capacity(90bytes)-PACKET REJECTED
...Program finished with exit code 0
Press ENTER to exit console.

CYCLE-2

LAB5: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:

```
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()
```

```
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

```
The server is ready to receive

Sent contents of ServerTCP.py
The server is ready to receive
```

```
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()
```

CYCLE-2

LAB6: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.

CLIENT.PY

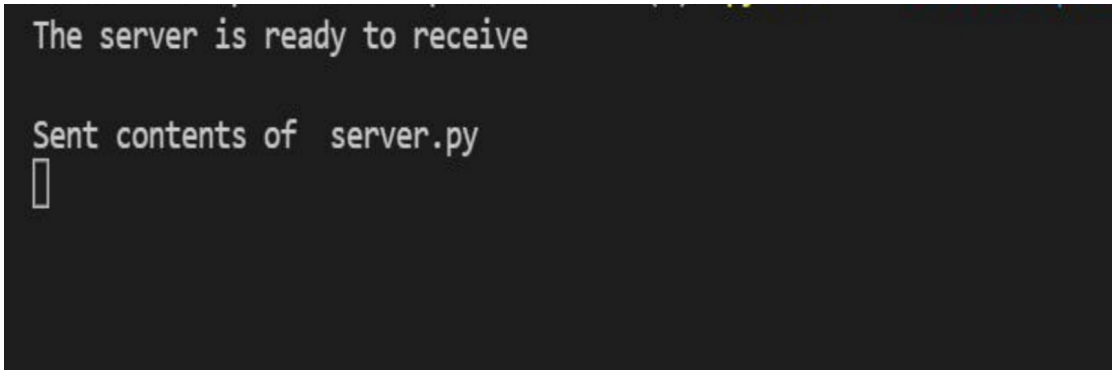
```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

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"))
# for i in filecontents:
# print(str(i), end = '')
clientSocket.close()
```

SERVER.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)
    #  for i in sentence:
    #  print (str(i), end = '')
    file.close()
```

Output



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Enter file name:  server.py

Reply from Server:

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)
    #  for i in sentence:
    #  print (str(i), end = '')
    file.close()
```