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

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

// CRC-CCITT polynomial: x^16 + x^12 + x^5 + 1 (0x1021)
//#define CRC_POLY 0x1021

// Function to perform bitwise XOR on binary strings  void
binaryXOR(char *result, const char *a, const char *b) {  for
(int i = 0; i < 16; i++) {  result[i] = (a[i] == b[i]) ? '0' : '1';
    }   result[16] =
   '\0';
}

// Function to calculate CRC-CCITT checksum  void
calculateCRC(const char *data, int length, char *checksum) {  char
crc[17];  for (int i = 0; i < 16; i++) {  crc[i] = '0';
    }   crc[16] =
   '\0';

    for (int i = 0; i < length; i++) {
      for (int j = 0; j < 8; j++) {
          char msb = crc[0];  for (int
         k = 0; k < 16; k++) {  crc[k]
         = crc[k + 1];
          }   crc[15] =
         '0';

          if (msb == '1') {
              char temp[17];  binaryXOR(temp, crc, "10001000000100001"); // CRC_POLY
             in binary  strcpy(crc, temp);
          }   }   crc[15] = (data[i] == '1')
      ? '1' : '0';
    }
    strcpy(checksum, crc);
}

void main() {
    char data[100]; // Replace with your actual
```

```
    data  printf("Enter data in binary: ");
    scanf("%s", data);

     int dataLength = strlen(data);  char
    checksum[17];  calculateCRC(data,
    dataLength, checksum);  printf("Calculated
    CRC: %s\n", checksum);

     // Simulating error by changing a bit
     // data[2] ^= 0x01; // Uncomment this line to introduce an error

     // Verify the received data  char
    receivedChecksum[17];
    printf("Enter received CRC: ");
    scanf("%s",
    receivedChecksum);

     if (strcmp(receivedChecksum, checksum) == 0)
        printf("Data is error-free.\n");

     else   printf("Data contains
        errors.\n");
}
```

Aim : Write a program for congestion control using Leaky bucket algorithm.

```
#include<stdio.h>

void main()
{
  int
  psize,bsize,outgoing,emptyspace,choice;
  printf("Enter the Bucket size = ");
  scanf("%d",&bsize);  emptyspace=bsize;
  printf("Enter the outgoing rate = ");
  scanf("%d",&outgoing);  while(1)
   {  printf("\nEnter the packet size =
     ");
     scanf("%d",&psize);

      if(psize<bsize&&psize<=emptyspace)
      {
          emptyspace=emptyspace-psize;  printf("The Packet of size %d is
         added and in the bucket \n",psize);  emptyspace+=outgoing;
      }

      else
      {
         printf("The Packet of size %d is dropped due to lack of space in the bucket\n");
      }

      printf("\nEnter 1 to Continue or 0 to Stop:
     ");   scanf("%d",&choice);   if(choice==0)
     break;

   }
}
```

Output :

```
"C:\Users\HP\Downloads\Bur

Enter the Bucket size = 5000
Enter the outgoing rate = 200

Enter the packet size = 3000
The Packet of size 3000 is added and in the bucket

Enter 1 to Continue or 0 to Stop: 1

Enter the packet size = 2000
The Packet of size 2000 is added and in the bucket

Enter 1 to Continue or 0 to Stop: 1

Enter the packet size = 1500
The Packet of size 6422296 is dropped due to lack of space in the bucket

Enter 1 to Continue or 0 to Stop: 1

Enter the packet size = 100
The Packet of size 100 is added and in the bucket

Enter 1 to Continue or 0 to Stop: 0

Process returned 0 (0x0)   execution time : 33.269 s
Press any key to continue.
```

Experiment 15

Aim : 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:

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 serve 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()
```

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()
```

Output :

Server instance:

```
*IDLE Shell 3.11.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====================== RESTART: F:/CN/cycle2/ServerTCP.py ======================
The server is ready to receive

Sent contents ofServerTCP.py
The server is ready to receive
```

Client instance:

```
IDLE Shell 3.11.2
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====================== RESTART: F:/CN/cycle2/ClientTCP.py ======================

Enter the file name: ServerTCP.py

From sever:

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()

>>>
```

Experiment 16

Aim : 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.

Program:

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 *  serverPort=12000
serverName="127.0.0.1"
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 :

Server instance :

```
*Python 3.6.7 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 (
4)] on win32
Type "help", "copyright", "credits" or "license()" for more informati
>>>
============ RESTART: D:\AUG_DEC 2021\CN\LAB\cycle 3\ServerUDP.py ===
The server is ready to receive

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

Client instance :

```
Python 3.6.7 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: D:\AUG_DEC 2021\CN\LAB\cycle 3\ClientUDP.py ============

Enter file name:  ServerUDP.py

Reply from Server:

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))

while 1:
     print ("The server is ready to receive")
     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()

>>>
```