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

```
def xor1(a, b):
    x = ""
    for i in range(1, len(a)):
        if a[i] == b[i]:
            x += "0"
        else:
            x += "1"
    return x

def modulo2(divident, divisor):
    divlen = len(divisor)
    temp = divident[0:divlen]
    while(divlen < len(divident)):
        if temp[0] == "1":
            temp = xor1(temp, divisor)+divident[divlen]
        else:
            temp = temp[1:divlen]+divident[divlen]
        divlen += 1
    if temp[0] == "1":
        temp = xor1(temp, divisor)
    if len(temp) < len(divisor):
        return "0"+temp
    return temp


def encode(data, key):
    append = data+"0"*(len(key))
    rem = modulo2(append, key)
    print("remaindar="+rem)
    code = data+rem
    print("code="+code)

    # Checking the logic:

    rem = modulo2(code, key)
    print("Remaindar we get when we do not have error="+rem)
```

```
    code = code.replace("011", "101")
    rem = modulo2(code, key)
    print("Remaindar we get when we have error="+rem)


def polytobin(string):
    keys = []
    key = ""
    for i in string:
        if i == '+':
            keys.append(int(key[1:]))
            key = ""
            continue
        key += i
    if key != "":
        keys.append(0)
    binary = ""
    j = 0
    print(keys)
    for i in range(keys[0], -1, -1):
        if i == (keys[j]):
            binary += "1"
            j += 1
        else:
            binary += "0"
    print(binary)
    return binary

string = input("Enter the key polynomial:\n")
key = polytobin(string)
string = input("Enter the data polynomial:\n")
data = polytobin(string)
print(key, data)
encode(data, key)
```

```
PS D:\engineering\sem5\computer networks> cd "d:\engineering\sem5

 Enter Frame size: 16

 Enter Frame:1 0 0 0 1 0 0 0 1 0 1 1
1 0 0 1

Enter the key polynomial:
x16+x12+x4+1
[16, 12, 4, 0]
10001000000010001
Enter the data polynomial:
x15+x12+x11+x8+x7+x4+x3+1
[15, 12, 11, 8, 7, 4, 3, 0]
1001100110011001
10001000000010001 1001100110011001
remaindar=0000100100010010
code=10011001100110010000100100010010
Remaindar we get when we do not have error=0000000000000000
Remaindar we get when we have error=0011001100110000
PS D:\engineering\sem5\computer networks>
```

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

```
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = []

    def add_edge(self, s, d, w):
        self.graph.append([s, d, w])

    def print_solution(self, dist, src, next_hop):
        print("Routing table for ", src)
        print("Dest \t Cost \t Next Hop")
        for i in range(self.V):
            print("{0} \t {1} \t {2}".format(i, dist[i], next_hop[i]))

    def bellman_ford(self, src):
        dist = [99] * self.V
        dist[src] = 0
        next_hop = {src: src}
        for _ in range(self.V - 1):
            for s, d, w in self.graph:
                if dist[s] != 99 and dist[s] + w < dist[d]:
                    dist[d] = dist[s] + w
                    if s == src:
                        next_hop[d] =d
                    elif s in next_hop:
                        next_hop[d] = next_hop[s]
        for s, d, w in self.graph:
            if dist[s] != 99 and dist[s] + w < dist[d]:
                print("Graph contains negative weight cycle")
                return   self.print_solution(dist, src, next_hop)

def main():
    matrix = []
    print("Enter the no. of routers:")
    n = int(input())
```

```
    print("Enter the adjacency matrix : Enter 99 for infinity")
    for i in range(0,n):
        a = list(map(int, input().split(" ")))
        matrix.append(a)

    g = Graph(n)
    for i in range(0,n):
        for j in range(0,n):
            g.add_edge(i,j,matrix[i][j])

    for k in range(0, n):
        g.bellman_ford(k)
main()
```

```
PS D:\engineering\sem5\computer networks> python -u "d:\eng
Enter the no. of routers:
4
Enter the adjacency matrix : Enter 99 for infinity
0 99 3 7
4 0 99 5
7 1 0 5
99 5 8 0
Routing table for  0
Dest     Cost    Next Hop
0        0       0
1        4       2
2        3       2
3        7       3
Routing table for  1
Dest     Cost    Next Hop
0        4       0
1        0       1
2        7       0
3        5       3
Routing table for  2
Dest     Cost    Next Hop
0        5       1
1        1       1
2        0       2
3        5       3
Routing table for  3
Dest     Cost    Next Hop
0        9       1
1        5       1
2        8       2
3        0       3
```

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

```
#include<bits/stdc++.h>
using namespace std;
#define V 5

int minDistance(int dist[], bool sptSet[])
{
   int min = 9999, 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 printPath(int parent[], int j)
{
   if (parent[j] == - 1)
      return;
   printPath(parent, parent[j]);
   cout<<j<<" ";
}

void printSolution(int dist[], int n, int parent[])
{
   int src = 0;
   cout<<"Vertex\t Distance\tPath"<<endl;
   for (int i = 1; i < V; i++)
   {
      cout<<"\n"<<src<<" -> "<<i<<" \t "<<dist[i]<<"\t\t"<<src<<" ";
      printPath(parent, i);
   }
}

void dijkstra(int graph[V][V], int src)
{

   int dist[V];
```

```
    bool sptSet[V];
    int parent[V];
    for (int i = 0; i < V; i++)
    {
        parent[0] = -1;
        dist[i] = 9999;
        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] + graph[u][v] < dist[v])
            {
                parent[v] = u;
                dist[v] = dist[u] + graph[u][v];
            }
    }

    printSolution(dist, V, parent);
}

int main()
{
    int graph[V][V];
    cout<<"Enter the graph (Enter 99 for infinity): "<<endl;
    for(int i = 0; i<V; i++)
    {
        for(int j = 0; j<V; j++)
            cin>>graph[i][j];
    }
    cout<<"Enter the source: "<<endl;
    int src;
    cin>>src;
```

```
    dijkstra(graph, src);
    cout<<endl;
    return 0;
}
```

```
PS D:\engineering\sem5\computer networks> cd "d:\e
TRA }
Enter the graph (Enter 99 for infinity):
0 1 5 99 99
1 0 3 99 9
5 3 0 4 99
99 99 4 0 2
99 9 99 2 0
Enter the source:
0
Vertex   Distance       Path

0 -> 1   1              0 1
0 -> 2   4              0 1 2
0 -> 3   8              0 1 2 3
0 -> 4   10             0 1 4
PS D:\engineering\sem5\computer networks>
```

10. Write a program for congestion control using Leaky bucket algorithm

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

#define NOF_PACKETS 5

int main()
{
   int packet_sz[NOF_PACKETS], i, b_size, o_rate, p_sz_rm = 0, p_sz, op;
   for (i = 0; i < NOF_PACKETS; ++i)
      packet_sz[i] = rand() % 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) /*compare the packet siz with bucket 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) /*packet size remaining comparing with output 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!!");
          }
        }
      }
    }
}
```

```
PS D:\engineering\sem5\computer networks> cd "d:\engineering\sem5\comp
\LEAKY_BUCKET }

packet[0]:41 bytes
packet[1]:67 bytes
packet[2]:34 bytes
packet[3]:0 bytes
packet[4]:69 bytes
Enter the Output rate:30
Enter the Bucket Size:85


Incoming Packet size: 41
Bytes remaining to Transmit: 41
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 11
Packet of size 11 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 67
Bytes remaining to Transmit: 67
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 37
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 7
Packet of size 7 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 34
Bytes remaining to Transmit: 34
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 4
Packet of size 4 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 0
Bytes remaining to Transmit: 0

Incoming Packet size: 69
Bytes remaining to Transmit: 69
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 39
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 9
Packet of size 9 Transmitted----Bytes Remaining to Transmit: 0
PS D:\engineering\sem5\computer networks>
```

11. 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.

SERVER:

```
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()
```

CLIENT:

```
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:

```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========= RESTART: D:\engineering\sem5\computer networks\SERVER_TCP.py =========
The server is ready to receive

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

CLIENT:

```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========= RESTART: D:\engineering\sem5\computer networks\CLIENT_TCP.py =========

Enter file name: SERVER_TCP.py

From Server:

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()
```

12. 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.

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()
```

CLIENT:

```
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()
clientSocket.close()
```

OUTPUT:

SERVER:

```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========= RESTART: D:\engineering\sem5\computer networks\SERVER_UDP.py =========
The server is ready to receive

Sent contents of  SERVER_UDP.PY
```

CLIENT:

```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========= RESTART: D:\engineering\sem5\computer networks\CLIENT_UDP.py =========

Enter file name:  SERVER_UDP.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()
```