Lab- 8

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

Program:-

```
def xor(a,b):
    result = []

    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)

def mod2div(dividend, divisor):
    length = len(divisor)
    tmp = dividend[0:length]

    while length < len(dividend)
        if (tmp[0] == '1'):
            tmp = xor(divisor, tmp)
                dividend[length]

        else:
            tmp = xor('0'*length, tmp) +
                dividend[length]


        length += 1
        if tmp[0] == '1':
            tmp = xor(divisor, tmp).

        else:
            tmp = xor('0'*length, tmp)
    Checkword = tmp
    return Checkword
```

```
def encodeData(data, key):
    keylength = len(key)

appended_data = data + '0' * (keylength-1)
rem = mod2div(appended_data, key)

codeword = data + rem
print("Remainder : ", rem)
print("Encoded data (Data + Remainder) : ", codeword)

data = input("Enter the dataword : ")
data = input("Enter the generator : ")
encodeData = encodeData(data, key)
decodeData = (encodeData, key)
```

Output :

Enter the dataword - 10100101001010100101010

Enter the generator - 10001000000100001

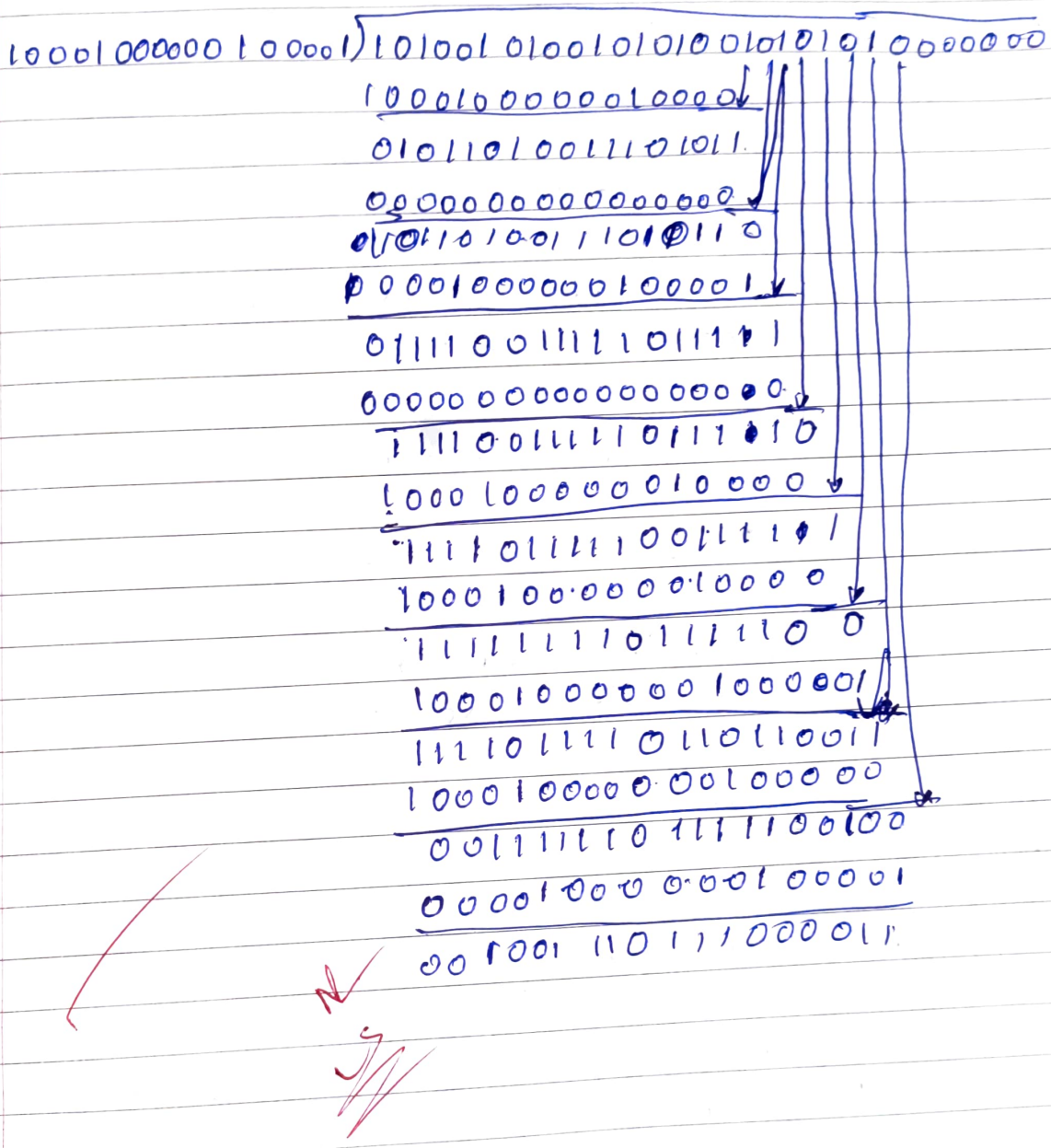
Remainder of mod2 division is : 0010101110011001

Encoded Data :- (10100101001010010101010110
10011001).

Working

CRC-16 Generator word: 10001 000000010000 1

Dataword: 1010010100101 01001 01 01 010



Lab - 9

Aim: Implement Dijkstra's algorithm to compute the shortest path

```
#include <stdio.h>
void dijkstras();
int c[10][10], n, src;
void main()
{
int i, j;
printf("\n enter the no of vertices : \t");
scanf("%d", &n);
printf("\n enter the cost matrix : \n");
for(i=1; i<=n; i++)
 {
  for(j=1; j<=n; j++)
  {
  scanf("%d", &c[i][j]);
  }
printf("\n enter the source node : \t");
scanf("%d", &src);
dijkstras();
getch();
}

void dijkstras()
{
int vis[10], dist[10], u, j, count, min;
for(j=1; j<=n; j++)
 {
  dist[j] = c[src][j];
 }
for(j=1; j<=n; j++)
{
 if(dist[j] < min && vis[j] != 1)
```

```
        {
         min = dist [j];
         u = j;
        }
      }
      vis [u] = 1;
      count ++;
      for (j = 1 ; j <= n ; j++)
      {
       if (min + c [u] [j] < dist [j] && vis[j] != 1)
       {
        dist [j] = min + c[u] [j];
       }
      }
    }
    printf ("\n the shortest distance is : \n");
    for (j = 1; j <= n ; j++)
    {
     printf ("\n %d ----> %d = %d", src, j, dist[j]);
    }
   }
```

Output

Enter the num. of vertices = 5.

Enter the cost matrix

9999 3 9999 7 9999

3 9999 4 2 9999

9999 4 9999 5 6

7 2 5 9999 4

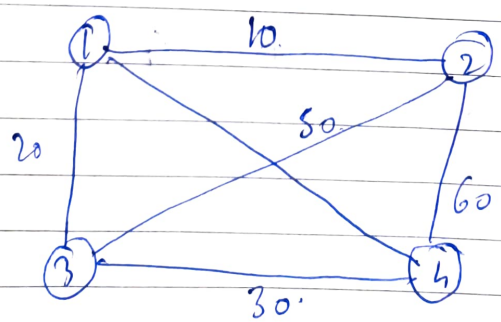
9999 9999 6 4 9999

Enter the source node: 1
The shortest distance is:
1 → 1 = 0
1 → 2 = 23
1 → 3 = 27
1 → 4 = 35
1 → 5 = 9.

Output:
Enter the number of nodes: 4
Enter the cost matrix: 9999 10 20 40
10 9999 50 60
20 50 9999 30
40 60 30 9999

Enter the source node: 1
The shortest distance is
1 → 1 = 0
1 → 2 = 10
1 → 3 = 20
1 → 4 = 40

Topology.



Lab-10

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

```
#include <stdio.h>
#include <stdlib.h>
int transmit (int);
void main ()
{  int bucketsize, ch, incoming Data, content=0, i=0;
   printf ("Enter the bucket size : \t");
   scanf ("%d", &bucketsize);
  while (i<10).
  {
   printf ("1 --> Enter the incoming data packet size
                \t \n 2 --> Exit \n");
   scanf ("%d", &ch);
   switch (ch)
   {
   case 1;
      printf ("Enter the packet size \n");
      scanf ("%d", &incoming Data);
      if ((content + incoming Data) < bucketsize)
      {
        content + = incoming Data;
        content = transmit (content);
      }
      else
      {
      printf ("Bucket size exceeded can not take
                this data input \n");
      content = transmit (content);
      }
     break;
```

```
case 2:
    exit (1);
    break;
default:
    printf ("Invalid input \n");
    break;
}
i ++;
}
}

int transmit (int content)
{
if (content == 0)
{
printf ("The Bucket has nothing to transmit
}
else if (content <= 3)
{
printf ("Data Transmitted : \t%d \n Data
        remaining in the bucket : \t0 \n",
        Content);
content = 0;
}
else
{
content -= 3;
printf ("Data Transmitted : \t3 \n Data
        remaining in the bucket : \t%d \n", content
}
return content;
```

```
        content = 0;
    }
    else
    {
        content -= 3;
        printf ("Data transmitted: \t3 \n Data
                remaining in the bucket: \t%d\n",
                content);
    }
    return content;
}
```

Output:

Enter the bucket size : 10

1 -> Enter the incoming data packet size;

2 -> Exit

1

Enter the packet size

5.

Data Transmitted : 3

Data remaining in the bucket: 2

1 -> Enter the incoming data packet size

2 -> Exit

2.

Lab-11

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

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 10
int n;
class Router
{
    char adj_new[MAX], adj_old[MAX];
    int table_new[MAX], table_old[MAX];
public:
    router()
    {
        for(int i=0; i<MAX; i++)
            table_old[i] = table_new[i] = 99;
    }
    void copy()
    {
        for(int i=0; i<n; i++)
        {
            adj_old[i] = adj_new[i];
            table_old[i] = table_new[i];
        }
    }
    int equal()
    {
        for(int i=0; i<n; i++)
            if(table_old[i] != table_new[i] ||
               adj_new[i] != adj_old[i])
                return 0;
        return 1;
    }
```

```
Void input (int j)
{
  cout << "Enter 1 if the corresponding router is
          adjacent to router "
       << (char) ('A' + j) << "else enter 999; " << endl
       << " ";
  for (int i = 0; i<n; i++)
    if (i != j)
      cout << (char) ('A' + i) << " ";
  Cout << "\nEnter metrix : ";
  for (int i = 0; i<n; i++)
  {
    if (i==j)
      table_new[i] = 0;
    else
      cin >> table_new[i];
    adj_new[i] = (char) ('A' + i);
  }
  cout << endl;
}
Void display()
{
  Cout << "\n Destination Routers: ";
  for (int i = 0; i<n; i++)
    cout << (char) ('A' + i) << " ";
  Cout << "\n Outgoing Line: ";
  for (int i = 0; i<n; i++)
    cout << adj_new[i] << " ";
  Cout << "\n Hop count : ";
  for (int i = 0; i<n; i++)
    Cout << table_new[i] << " ";
}
```

```
void build (int g)
{
   for (int i = 0; i<n; i++)
      for (int k = 0; (i != j) && (k < n); k++
         if (table_old[i] != qq)
          if ((table_new[i] + table_new[k])
              < table_new[k])
          {
           table_new[k] = table_new[i] + table_ne
           obj-new[k] = (char)('A'+j);
          }
   }
} r[max];
void build table()
{
   int i=0, j=0;
   while (i != n)
   {
      for (i = j; i<n; i++)
      {
         r[i].copy();
         r[i].build(i);
      }
      for (i=0; i<n; i++)
        if (!r[i].equal())
        {
          j = g;
          break;
        }
   }
}
```

```
int main()
{
  cout << "Enter the number of routers(<="<<MAX
       << "):";
  cin >>n;
  for (int i=0 ; i<n; i++)
    r[i].input(i);
  build_table();
  for (int i=0 ; i<n; i++)
  {
    cout << "Router Table entries for router "
         <<(char)('A'+i) << ":-";
    r[i].display();
    cout <<endl
         <<endl;
  }
}
```

Output:-

```
Enter the number of nodes: 4
Enter 999 if there is no direct route.
Enter distance to the node:
   5 3 7
Enter distance from node 2 to other nodes:
5 999 6
Enter distance from node 3 to other nodes:-
   3 999 999
Enter distance from node 4 to other nodes.
: 999 999 999
Routing table:
 1 0 1
2 5 2
3 3 3
4 7 4
```

LAB-12

Aim: Using TCP/IP sockets, write a client program to make client sending the file and the server to send back the contents of the requested file if present.

Server.Py

```
From socket import *
serverName = 'DESKTOP-4AC6LOM'
serverPort = 12001
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("server running")
    connectionSocket, addr = serverSocket.a
    sentence = connectionSocket.recv(1024)
                .decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l.encode())
    print("\n Sent contents", sentence)
    file.close()
    connectionSocket.close()
```

Clint.Py

```
ServerName = "DESKTOP-4AC6LOM"
serverPort = 12001
ClientSocket = socket(AF_INET, SOCK_STREAM)
ClientSocket.connect((serverName, serverPort))
Sentence = input("\n Enter filename:")
clintSocket.sent(sentence.encode())
```

```
file = clientSocket.recv(1024).decode()
print ("\n From Server \n")
print (file)
clientSocket.close()
```

Output:

Server.py

```
Server running
```

Client.py

```
Enter FileName : Hello.txt
From Server : Hello-World by Vinayak
```

Program 6:

Using UDP Sockets, WAP to make client send the filename & server to send back the contents of the requested file if present.

Client UDP.Py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = input("Enter Filename:")
clientSocket.sendto(
clientSocket.serverAddress = (clientSocket.recvfrom(2048)
print("In Reply from Server: \n")
print(file contents.decode("utf-8"))
clientSocket.close()
```

Server UDP.Py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
server.Socket.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("\n Sent contents of ", end = ' ')
print (sentence).
file.close()
```

Output:

Server.

```
python server UDP.py
The server is ready to receive
```

Sent contents of Hello.txt

Client.

```
Python client UDP.py
Enter the filename: Hello.txt

Reply from Server.
Hello world!
)
```