CN LAB RECORD CYCLE 2

NAME:-MITHIL RAJ USN:-1BM19CS086
BATCH:- B2
LAB-1CRC:-
WRITE A PROGRAM FOR ERROR DETECTION CODE USING CRC-CCITT(16 BITS)

```
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):

    pick = len(divisor)
    tmp = dividend[0: pick]

    while pick < len(dividend):

        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[pick]

        else:
            tmp = xor('0' * pick, tmp) + dividend[pick]
        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)

    checkword = tmp
```

```
    return checkword


def encodeData(data, key):
    l_key = len(key)

    appended_data = data + '0' * (l_key - 1)
    remainder = mod2div(appended_data, key)

    codeword = data + remainder
    return codeword

def decodeData(code, key):
    remainder = mod2div(code, key)
    return remainder

data=input("Enter Data: ")
print("dataword:"+str(data))

key = "10001000000100001"
print("generating polynomial:"+key)
codeword = encodeData(data, key)
print("Checksum: ",codeword)
print("Transmitted Codeword:"+str(codeword))
code = input("enter transmitted codeword:")

recieved_data = int(decodeData(code, key))

if recieved_data == 0:
    print("NO ERROR")
else:
    print("ERROR")
    print(recieved_data)
```

OUTPUT:-

```
PS D:\5th Sem\CN\CN Lab\Cycle 2> python -u "d:\5th Sem\CN\CN
Enter Data: 1011001001011101
dataword:1011001001011101
generating polynomial:10001000000100001
Checksum:  10110010010111011111001100110111
Transmitted Codeword:10110010010111011111001100110111
enter transmitted codeword:10110010010111011111001100110111
NO ERROR
PS D:\5th Sem\CN\CN Lab\Cycle 2>
```

LAB 2:-DISTANCE VECTOR ALGORITM:-

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()
```

OUTPUT:-

```
PS D:\5th Sem\CN\CN Lab\Cycle 2> python -u "d:\5th
Enter the no. of routers:
5
Enter the adjacency matrix : Enter 99 for infinity
0 1 5 99 99
5 3 0 4 99
99 99 4 0 2
99 9 99 2 0
Routing table for  0
Dest      Cost    Next Hop
0         0       0
1         1       1
2         4       1
3         8       1
4         10      1
Routing table for  1
Dest      Cost    Next Hop
0         1       0
1         0       1
2         3       2
3         7       2
4         9       4
Routing table for  2
Dest      Cost    Next Hop
0         4       1
1         3       1
2         0       2
3         4       3
4         6       3
Routing table for  3
Dest      Cost    Next Hop
0         8       2
1         7       2
2         4       2
3         0       3
4         2       4
Routing table for  4
Dest      Cost    Next Hop
0         10      1
1         9       1
2         6       3
3         2       3
4         0       4
PS D:\5th Sem\CN\CN Lab\Cycle 2>
```

LAB 3:-DIJKSTRAS ALGORITHM

IMPLEMENT DIJKSTRAS ALGORITHM TO COMPUTE THE SHORTEST PATH FOR A GIVEN TOPOLOGY

```
import sys

class Graph:
    def __init__(self,vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printSolution(self, dist):
        print("Vertex \tDistance from Source")
        for node in range(self.V):
            print(node, "\t", dist[node])

    def minDistance(self, dist, sptSet):

        min = sys.maxsize

        for v in range(self.V):
            if dist[v] < min and sptSet[v] == False:
                min = dist[v]
                min_index = v

        return min_index

    def dijkstra(self, src):
        dist = [sys.maxsize] * self.V
        dist[src] = 0
        sptSet = [False] * self.V

        for cout in range(self.V):

            u = self.minDistance(dist, sptSet)

            sptSet[u] = True
```

```
            for v in range(self.V):
                if self.graph[u][v] > 0 and sptSet[v] == False and dist[v] > dist[u] +
self.graph[u][v]:
                        dist[v] = dist[u] + self.graph[u][v]

        self.printSolution(dist)

g = Graph(9)
g.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]
        ]

g.dijkstra(0)
```

OUTPUT:-

```
PS D:\5th Sem\CN\CN Lab\Cycle 2> python
Vertex  Distance from Source
0       0
1       4
2       12
3       19
4       21
5       11
6       9
7       8
8       14
PS D:\5th Sem\CN\CN Lab\Cycle 2>
```

LAB 4:-

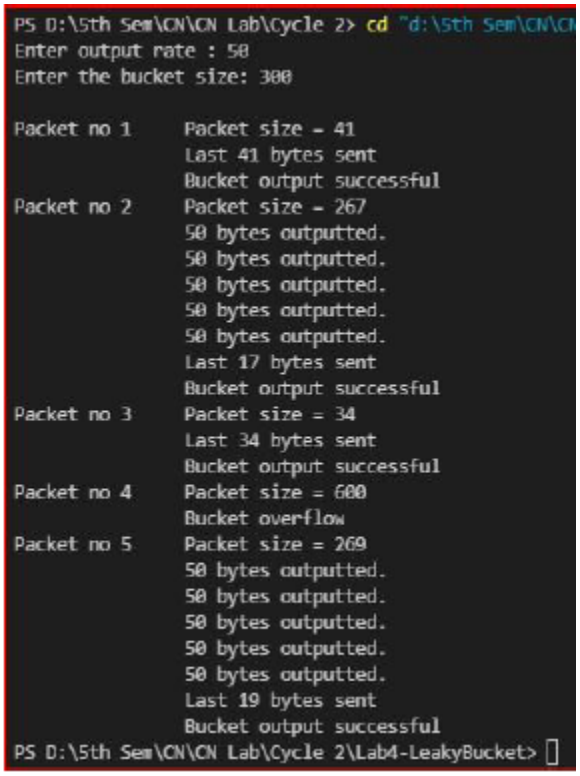
WRITE A PROGRAM FOR CONGESTION CONTROL USING LEAKY BUCKET ALGORITHM.

```
#include<bits/stdc++.h>
#include<unistd.h>
using namespace std;

int bucketSize;
void bucketInput(int a,int b)
{
    if(a > bucketSize)
        cout<<"\n\t\tBucket overflow";
    else{
        sleep(1);
        while(a > b){
            cout<<"\n\t\t"<<b<<" bytes outputted.";
            a-=b;
            sleep(1);
        }
        if(a > 0)
            cout<<"\n\t\tLast "<<a<<" bytes sent\t";
        cout<<"\n\t\tBucket output successful";
    }
}
int main()
{
    int op,pktSize;
    cout<<"Enter output rate : ";
    cin>>op;
    cout<<"Enter the bucket size: ";
    cin>>bucketSize;
    for(int i=1;i<=5;i++)
    {
        // sleep(rand()%10);
        pktSize=rand()%700;
        cout<<"\nPacket no "<<i<<"\tPacket size = "<<pktSize;
        bucketInput(pktSize,op);
    }
```

```
    return 0;
}
```

OUTPUT:-



LAB 5-TCP

USING TCP/IP SOCKET WRITE A CLINET SERVER PROGRAM TO MAKE CLENT SENDING THE FILE NAME AND THE SERVER TO SEND BACK THE CONENTS OF THE REQUESTED FILE IF PRESENT

CLIENT:-

```
import socket

serverName = '127.0.0.1'
serverPort = 12345

client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
client_socket.connect((serverName,serverPort))
sentence = input("Enter file name: ")

client_socket.send(sentence.encode())
filecontents = client_socket.recv(1024).decode()
print ('From Server:\n', filecontents)
client_socket.close()
```

SERVER:-

```
import socket

serverName = '127.0.0.1'
serverPort = 12345
#create
server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

#bind
server_socket.bind((serverName, serverPort ))

#listen
server_socket.listen(5)

while True:
    print("Server waiting for connection")
```

```
    client_socket, addr = server_socket.accept()
    print("Client connected from",addr)
    sentence = client_socket.recv(1024).decode()

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

    client_socket.send(l.encode())
    file.close()
    client_socket.close()
```

CLIENT:-

```
PS D:\5th Sem\CN\CN Lab\Cycle 2> python clientTCP.py
Enter file name: bmsce.txt
From Server:
 ********************************************
This is one of the top colleges in bangalore
********************************************

PS D:\5th Sem\CN\CN Lab\Cycle 2>
```

SERVER:-

```
PS D:\5th Sem\CN\CN Lab\Cycle 2> python -u "
Server waiting for connection
Client connected from ('127.0.0.1', 55469)
Server waiting for connection
```

LAB 6-UDP:-

USING THE UDP SOCKET,WRITE THE CLIENTSERVER PROGRAM TO MAKE CLIENT SENDING THE FILE NAME AND THE SERVER TO SEND BACK THE CONENETS OF THE REQUESTED FILE IF PRESENT

CLEINT:-

```
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:-

```
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:-
CLIENT:-

```
PS D:\5th Sem\CN\CN Lab\Cycle 2> python clientUDP.py

Enter file name:  bmsce.txt

Reply from Server:

*******************************************
This is one of the top colleges in bangalore
*******************************************

PS D:\5th Sem\CN\CN Lab\Cycle 2>
```

SERVER:-

```
PS D:\5th Sem\CN\CN Lab\Cycle 2> python
The server is ready to receive

Sent contents of  bmsce.txt
```

----------X----------