CN CYCLE- 2 REPORT

NAME: CHAITANYA GADGIL

USN: 1BM19CS223

BATCH: B4

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

```
import hashlib

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)
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ python3 LAB1/crc.py
Enter Data: 1001
dataword:1001
generating polynomial:10001000000100001
Checksum:  1001101000100101001
Transmitted Codeword:1001101000100101001
enter transmitted codeword:1001101000100101000
ERROR
1
```

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

```
class Topology:

    def __init__(self, array_of_points):
        self.nodes = array_of_points
        self.edges = []

    def add_direct_connection(self, p1, p2, cost):
        self.edges.append((p1, p2, cost))
        self.edges.append((p2, p1, cost))

    def distance_vector_routing(self):
        import collections
        for node in self.nodes:
            dist = collections.defaultdict(int)
            next_hop = {node: node}
            for other_node in self.nodes:
                if other_node != node:
                    dist[other_node] = 100000000 # infinity

            # Bellman Ford Algorithm
            for i in range(len(self.nodes)-1):
                for edge in self.edges:
                    src, dest, cost = edge
                    if dist[src] + cost < dist[dest]:
                        dist[dest] = dist[src] + cost
                        if src == node:
                            next_hop[dest] =dest
                        elif src in next_hop:
                            next_hop[dest] = next_hop[src]

            self.print_routing_table(node, dist, next_hop)
```

```
            print()

    def print_routing_table(self, node, dist, next_hop):
        print(f'Routing table for {node}:')
        print('Dest \t Cost \t Next Hop')
        for dest, cost in dist.items():
            print(f'{dest} \t {cost} \t {next_hop[dest]}')

    def start(self):
        pass
nodes = ['A', 'B', 'C', 'D', 'E']

t = Topology(nodes)

t.add_direct_connection('A', 'B', 1)
t.add_direct_connection('A', 'C', 5)
t.add_direct_connection('B', 'C', 3)
t.add_direct_connection('B', 'E', 9)
t.add_direct_connection('C', 'D', 4)
t.add_direct_connection('D', 'E', 2)

t.distance_vector_routing()
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ python3 LAB2/dist_vec.py
Routing table for A:
Dest     Cost    Next Hop
B        1       B
C        4       B
D        8       B
E        10      B
A        0       A

Routing table for B:
Dest     Cost    Next Hop
A        1       A
C        3       C
D        7       C
E        9       E
B        0       B

Routing table for C:
Dest     Cost    Next Hop
A        4       B
B        3       B
D        4       D
E        6       D
C        0       C

Routing table for D:
Dest     Cost    Next Hop
A        8       C
B        7       C
C        4       C
E        2       E
D        0       D

Routing table for E:
Dest     Cost    Next Hop
A        10      B
B        9       B
C        6       D
D        2       D
E        0       E
```

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

```python
import math
# For INF
def dijkstra(graph, n, src):
    distance = [math.inf] * n
    distance[src] = 0
    final_selected = [(src, distance[src])]
    curr_vertex = src

    while len(final_selected) < n:
        min_vertex, min_dist = -1, math.inf
        for neighbor in graph[curr_vertex]:
            vertex, weight = neighbor
            distance[vertex] = min(
                distance[curr_vertex] + weight, distance[vertex])
```

```
        for vertex in range(n):
            if distance[vertex] <= min_dist and (vertex, distance[vertex])
not in final_selected:
                min_vertex, min_dist = vertex, distance[vertex]

        final_selected.append((min_vertex, min_dist))
        curr_vertex = min_vertex

    print('Vertex\tDistance')
    [print(f'{v}\t{d}') for v, d in final_selected]
if __name__ == "__main__":
    n = int(input("Enter no of vertices: "))
    e = int(input("Enter no of edges: "))
    graph_dict = {}
    print("Enter the edges as follows: [start] [end] [weight]")
    for i in range(e):
        start, end, weight = [int(j) for j in input().split()]
        if not graph_dict.get(start):
            graph_dict[start] = [(end, weight)]
        else:
            graph_dict[start].append((end, weight))

        if not graph_dict.get(end):
            graph_dict[end] = [(start, weight)]
        else:
            graph_dict[end].append((start, weight))
    for i in range(n):
        print(f'Source {i}: ')
        dijkstra(graph_dict, n, i)
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ python3 LAB3/dijkstra.py
Enter no of vertices: 3
Enter no of edges: 4
Enter the edges as follows: [start] [end] [weight]
0 1 3
1 2 3
0 2 1
1 2 4
Source 0:
Vertex  Distance
0       0
2       1
1       3
Source 1:
Vertex  Distance
1       0
2       3
0       3
Source 2:
Vertex  Distance
2       0
0       1
1       3
```

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

```
#include<bits/stdc++.h>
#include<unistd.h>
using namespace std;
#define bucketSize 500
void bucketInput(int a,int b)
{
if(a > bucketSize)
cout<<"\n\t\tBucket overflow";
else{
sleep(5);
while(a > b){
cout<<"\n\t\t"<<b<<" bytes outputted.";
a-=b;
sleep(5);
}
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;
for(int i=1;i<=5;i++)
{
sleep(rand()%10);
pktSize=rand()%700;
cout<<"\nPacket no "<<i<<"\tPacket size = "<<pktSize;
bucketInput(pktSize,op);
}
cout<<endl;
return 0;
}
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ g++ LAB4/leaky_bucket.cpp
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ ./a.out
Enter output rate : 100

Packet no 1     Packet size = 186
                100 bytes outputted.
                Last 86 bytes sent
                Bucket output successful
Packet no 2     Packet size = 215
                100 bytes outputted.
                100 bytes outputted.
                Last 15 bytes sent
                Bucket output successful
Packet no 3     Packet size = 535
                Bucket overflow
Packet no 4     Packet size = 492
                100 bytes outputted.
                100 bytes outputted.
                100 bytes outputted.
                100 bytes outputted.
                Last 92 bytes sent
                Bucket output successful
Packet no 5     Packet size = 521
                Bucket overflow
```

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

Client.py

```
import socket
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 65432
print('\033[32m======== CLIENT ========\033[0m')
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.connect((SERVER_HOST, SERVER_PORT))
    while True:
        filename = input('Enter file name: ')
        if not filename:
            break
        sock.sendall(bytes(filename, 'utf-8'))
        print(f'Sent: {filename}')

        data = sock.recv(1024)
        contents = data.decode('utf-8')
        print(f'Received: {contents}')
        print()
```

Server.py

```
import socket
HOST = '127.0.0.1'
PORT = 65432
print('\033[36m======== SERVER ========\033[0m')
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.bind((HOST, PORT))
    sock.listen(1)
    conn, addr = sock.accept()
    with conn:
        print(f'Connected by: {addr}')
        while True:
            data = conn.recv(1024)
            if not data:
                break
            filename = data.decode('utf-8')
            print(f'Received Filename: {filename}')
            try:
                with open(filename, 'r') as f:
                    data = f.read()
                data = bytes(data, 'utf-8')
            except:
                data = bytes(f'File {filename} not found', 'utf-8')
            conn.sendall(data)
            print(f'Sent: {data}')
            print()
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ python3 LAB5/
server.py
======== SERVER ========
Connected by: ('127.0.0.1', 48234)
Received Filename: LAB5/testfile.txt
Sent: b'this is a sample text.\n\n'
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ python3 LAB5/
client.py
======== CLIENT ========
Enter file name: LAB5/testfile.txt
Sent: LAB5/testfile.txt
Received: this is a sample text.
```

6. 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

```python
import socket

HOST = '127.0.0.1'
PORT = 65432

print('\033[32m======== CLIENT ========\033[0m')

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
    sock.connect((HOST, PORT))
    while True:
        filename = input('Enter file to request from server: ')

        if not filename:
            break

        sock.sendall(bytes(filename, 'utf-8'))
        print(f'Sent: {filename}')

        data = sock.recv(1024).decode('utf-8')
        print(f'Received: {data}')
        print()
```

Server.py

```python
import socket
HOST = '127.0.0.1'
PORT = 65432
print('\033[36m======== SERVER ========\033[0m')
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:
    sock.bind((HOST, PORT))
    while True:
        data, addr = sock.recvfrom(1024)
        if not data:
            break

        filename = data.decode('utf-8')
        print(f'Received Filename: {filename} From: {addr}')

        try:
            with open(filename, 'r') as f:
                data = f.read()
            data = bytes(data, 'utf-8')
        except:
            data = bytes(f'File {filename} not found', 'utf-8')

        sock.sendto(data, addr)
        print(f'Sent: {data} To: {addr}')
        print()
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ python3 LAB6/
server.py
======== SERVER ========
Received Filename: LAB6/testfile.txt From: ('127.0.0.1', 42515)
Sent: b'this is a sample text.\n' To: ('127.0.0.1', 42515)
```

```
bugdigger123@bugdigger123-ASUS-TUF-Gaming-F15-FX506LI-FX506LI:~/CN-LAB$ python3 LAB6/
client.py
======== CLIENT ========
Enter file to request from server: LAB6/testfile.txt
Sent: LAB6/testfile.txt
Received: this is a sample text.


Enter file to request from server:
```