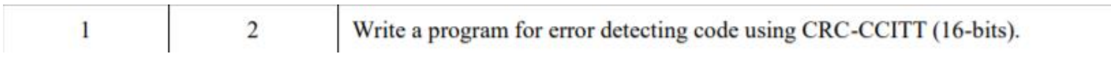
NAME: NEHA B CHADAGA

USN: 1BM19CS098

SEMESTER: 5 ‘B’

SUBJECT: COMPUTER NETWORKS LAB

RECORD OF CYCLE 2 PROGRAMS



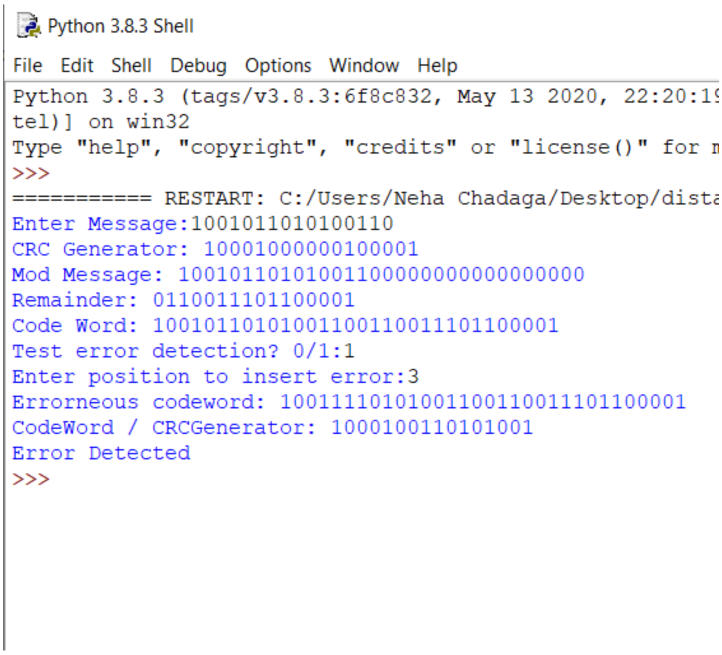
1	2	Write a program for error detecting code using CRC-CCITT (16-bits).

PROGRAM:

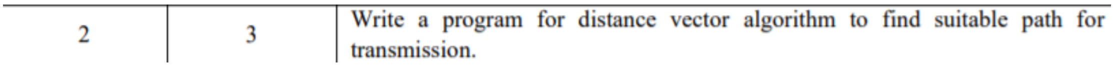
```
def xor(a, b):
    result = []

    # If bits are same XOR is 0, else 1
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')

    return ''.join(result)


def binaryDiv(genlen, msg, gen):
    pick = genlen
    tmp = msg[0:pick]

    while pick < len(msg):
        if tmp[0] == "1":
            tmp = xor(gen, tmp) + msg[pick]
        else:
            tmp = xor('0'*pick, tmp) + msg[pick]

        pick += 1

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

    return tmp

#
#
# Main
#
#


message = input("Enter Message:")

crcGenerator = "10001000000100001"
print("CRC Generator:", crcGenerator)

# get length of generator n
crcGenLength = len(crcGenerator)

# add trailing n-1 zeroes to the message
modMessage = str(int(message) * (10**(crcGenLength-1)))
print("Mod Message:", modMessage)

# rem = int(modMessage) / int(crcGenerator)
```

```
rem = binaryDiv(crcGenLength, modMessage, crcGenerator)
print("Remainder:", rem)

# generate codeword using remainder
codeword = str(int(modMessage) + int(rem))
print("Code Word:", codeword)

ch = int(input("Test error detection? 0/1:"))
if ch == 1:
    pos = int(input("Enter position to insert error:"))

    codeword = list(codeword)
    if codeword[pos+1] == '1':
        codeword[pos+1] = '0'
    else:
        codeword[pos+1] = '1'

    codeword = ''.join(codeword)

    print("Errorneous codeword:", codeword)

    # test = codeword / crcgenerator
    test = binaryDiv(crcGenLength, codeword, crcGenerator)
    print("CodeWord / CRCGenerator:", test)

    # if test = 0 => no error
    if int(test) == 0:
        print("No Error")
    else:
        print("Error Detected")

else:
    print("Skipping error insertion")
```

OUTPUT:



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:1
tel)] on win32
Type "help", "copyright", "credits" or "license()" for m
>>>
=========== RESTART: C:/Users/Neha Chadaga/Desktop/dista
Enter Message:1001011010100110
CRC Generator: 10001000000100001
Mod Message: 10010110101001100000000000000000
Remainder: 0110011101100001
Code Word: 10010110101001100110011101100001
Test error detection? 0/1:1
Enter position to insert error:3
Errorneous codeword: 10011110101001100110011101100001
CodeWord / CRCGenerator: 1000100110101001
Error Detected
>>>
```



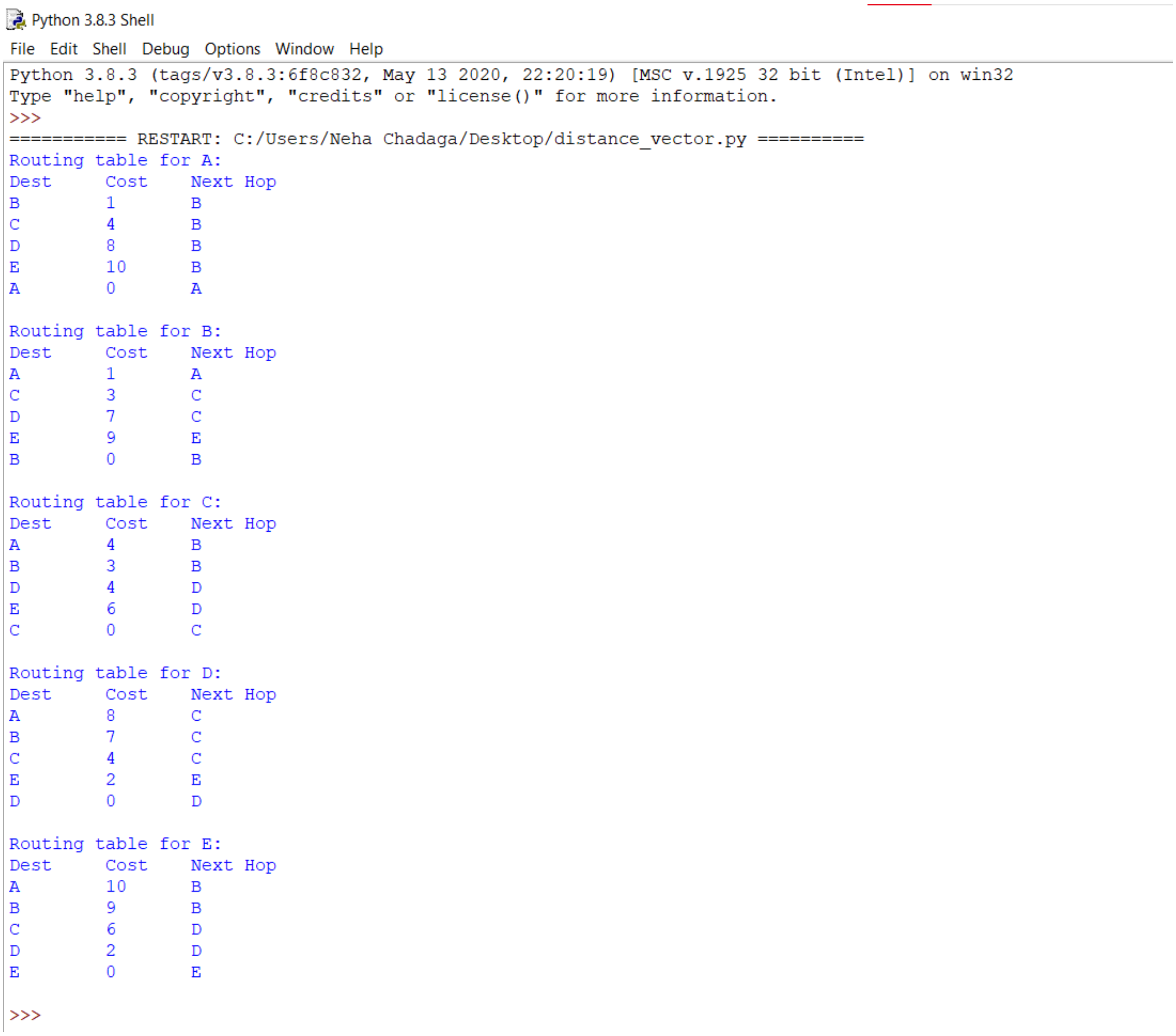
2	3	Write a program for distance vector algorithm to find suitable path for transmission.

PROGRAM:

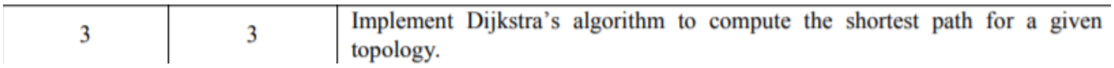
```
class Topology:
    def __init__(self, array_of_points):
        self.nodes = array_of_points
        self.edges = []

    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]}')
# Example 1
# Number of points
array = ['A', 'B', 'C', 'D', 'E']

# Create the network
t = Topology(array)

# Direct connection of each point in the Topology
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()

OUTPUT:



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:/Users/Neha Chadaga/Desktop/distance_vector.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

>>>
```

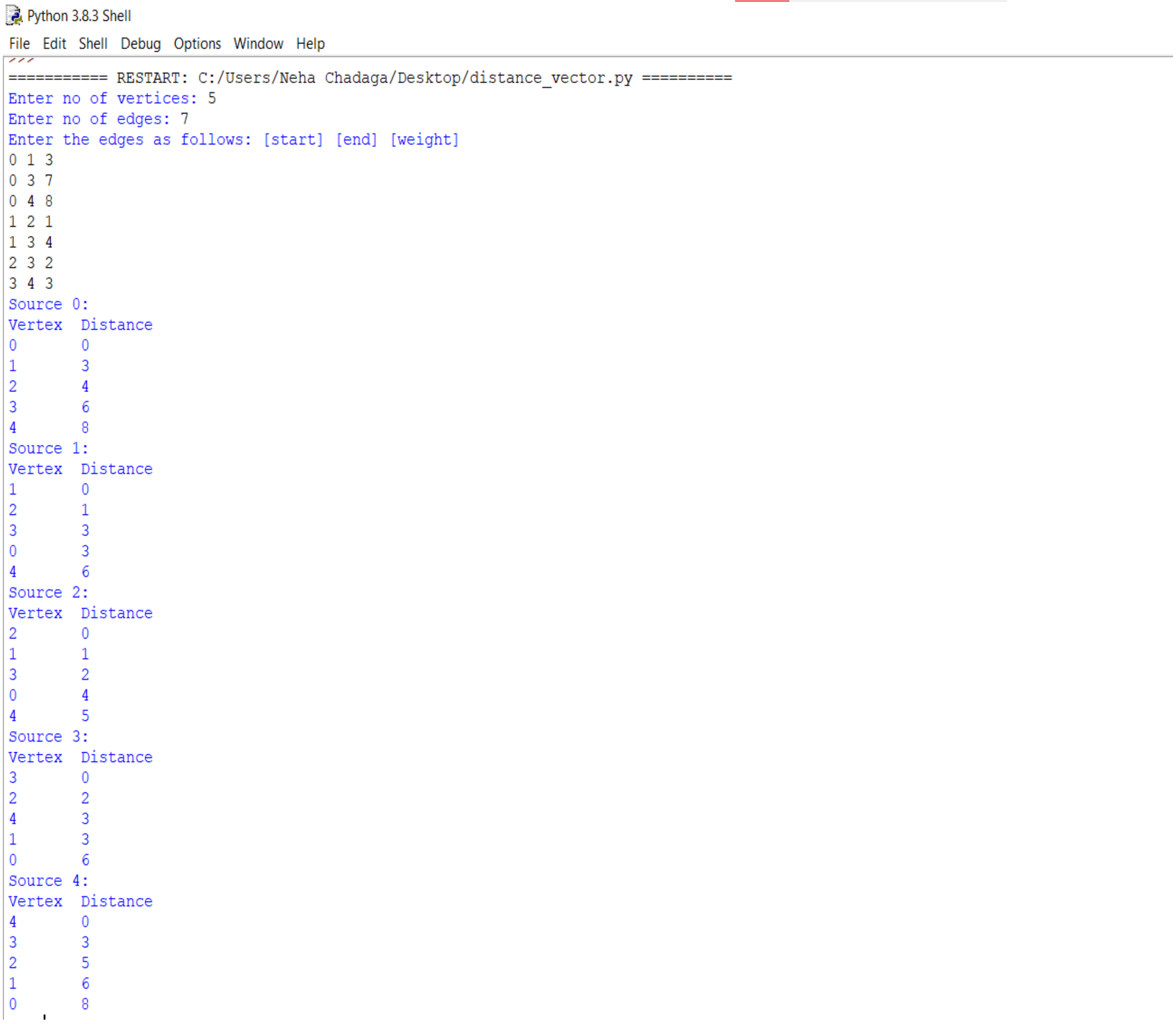


3	3	Implement Dijkstra's algorithm to compute the shortest path for a given topology.

PROGRAM:

```
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)
```

OUTPUT:



```
=========== RESTART: C:/Users/Neha Chadaga/Desktop/distance_vector.py ==========
Enter no of vertices: 5
Enter no of edges: 7
Enter the edges as follows: [start] [end] [weight]
0 1 3
0 3 7
0 4 8
1 2 1
1 3 4
2 3 2
3 4 3
Source 0:
Vertex  Distance
0       0
1       3
2       4
3       6
4       8
Source 1:
Vertex  Distance
1       0
2       1
3       3
0       3
4       6
Source 2:
Vertex  Distance
2       0
1       1
3       2
0       4
4       5
Source 3:
Vertex  Distance
3       0
2       2
4       3
1       3
0       6
Source 4:
Vertex  Distance
4       0
3       3
2       5
1       6
0       8
```

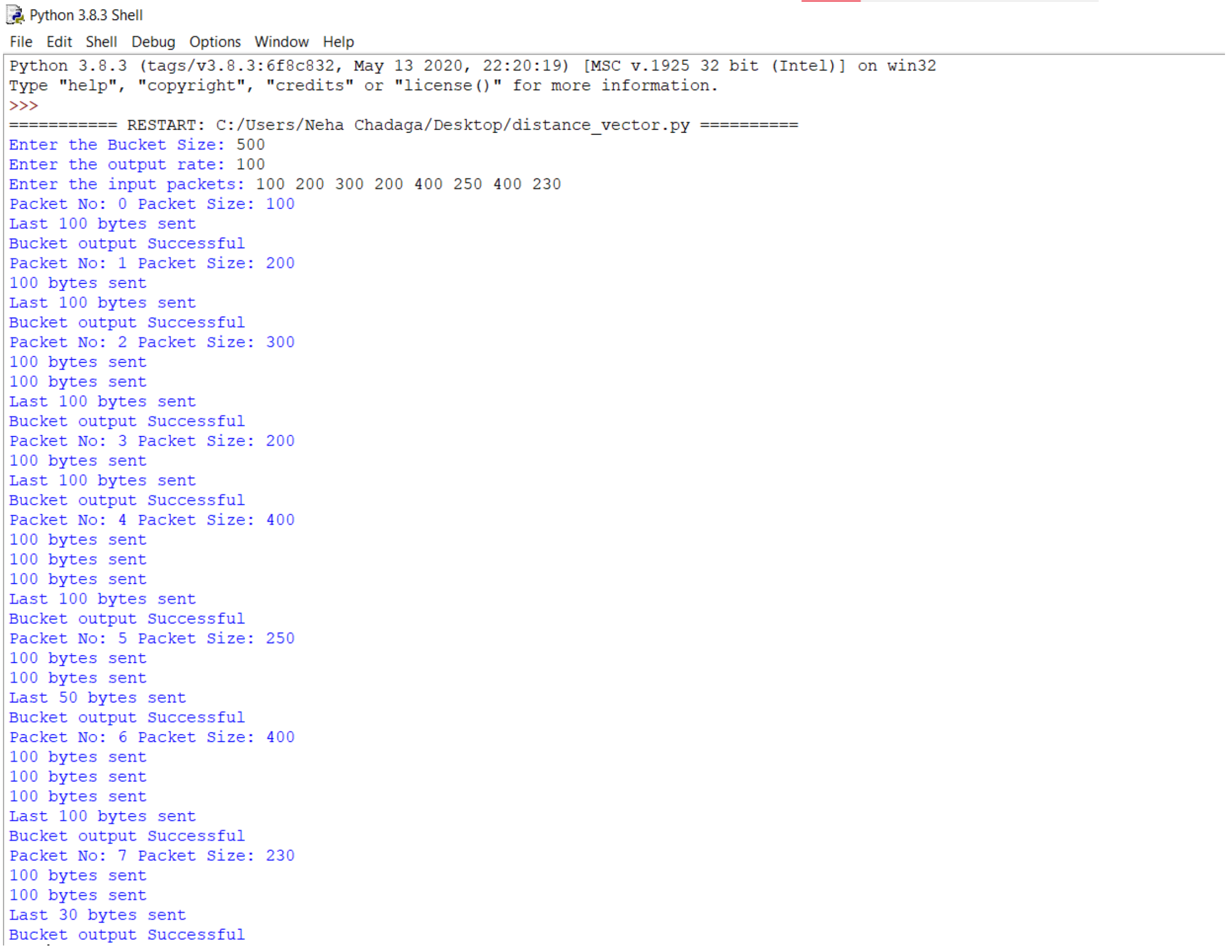
4	3	Write a program for congestion control using Leaky bucket algorithm.

PROGRAM:

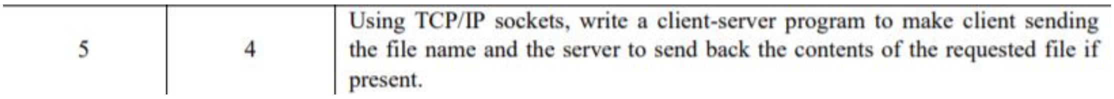
```
class LeakyBucket:
    def __init__(self, bucket_size, output_rate, packets):
        self.bucket_size = bucket_size
        self.output_rate = output_rate
        self.packets = packets

    def traffic_shaping(self):
        for i in range(len(self.packets)):
            packet_size = self.packets[i]
            print(f"Packet No: {i} Packet Size: {packet_size}")
            if packet_size > self.bucket_size:
                print("Bucket Overflow")
            else:
                while packet_size > output_rate:
                    print(f"{output_rate} bytes sent")
                    packet_size -= output_rate

                if packet_size:
```

```
            print(f"Last {packet_size} bytes sent")

        print("Bucket output Successful")


bucket_size = int(input("Enter the Bucket Size: "))
output_rate = int(input("Enter the output rate: "))
packets = [int(x) for x in input("Enter the input packets: ").split()]
lb = LeakyBucket(bucket_size, output_rate, packets)
lb.traffic_shaping()
```

OUTPUT:



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:/Users/Neha Chadaga/Desktop/distance_vector.py ==========
Enter the Bucket Size: 500
Enter the output rate: 100
Enter the input packets: 100 200 300 200 400 250 400 230
Packet No: 0 Packet Size: 100
Last 100 bytes sent
Bucket output Successful
Packet No: 1 Packet Size: 200
100 bytes sent
Last 100 bytes sent
Bucket output Successful
Packet No: 2 Packet Size: 300
100 bytes sent
100 bytes sent
Last 100 bytes sent
Bucket output Successful
Packet No: 3 Packet Size: 200
100 bytes sent
Last 100 bytes sent
Bucket output Successful
Packet No: 4 Packet Size: 400
100 bytes sent
100 bytes sent
100 bytes sent
Last 100 bytes sent
Bucket output Successful
Packet No: 5 Packet Size: 250
100 bytes sent
100 bytes sent
Last 50 bytes sent
Bucket output Successful
Packet No: 6 Packet Size: 400
100 bytes sent
100 bytes sent
100 bytes sent
Last 100 bytes sent
Bucket output Successful
Packet No: 7 Packet Size: 230
100 bytes sent
100 bytes sent
Last 30 bytes sent
Bucket output Successful
```



5	4	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:

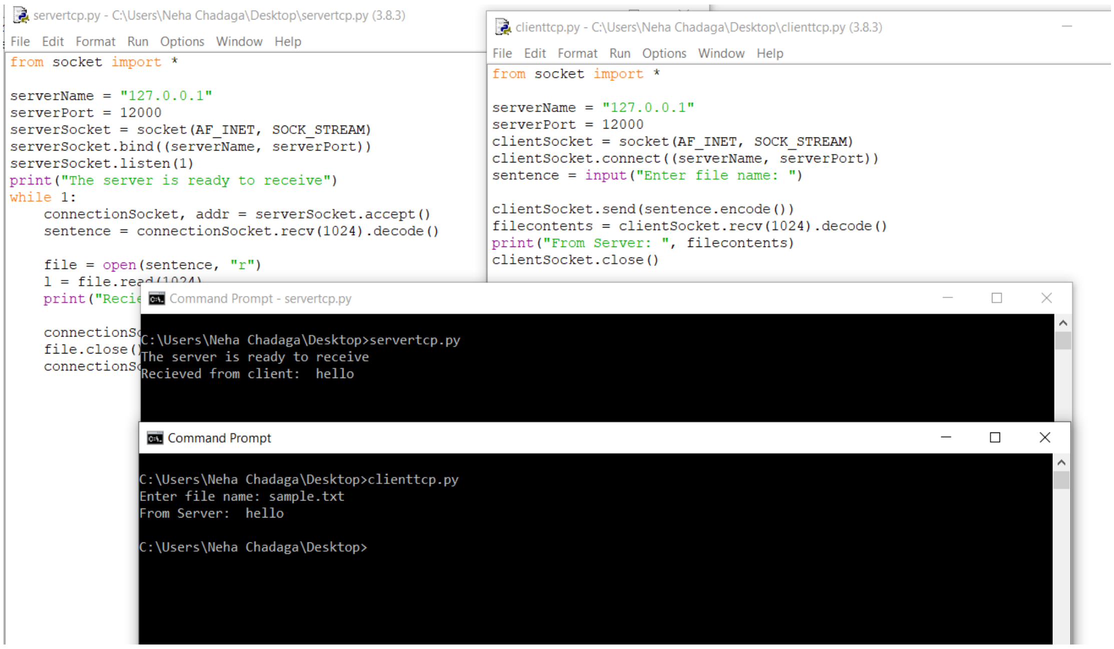
SERVER:

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
  connectionSocket, addr = serverSocket.accept()
  sentence = connectionSocket.recv(1024).decode()

  file = open(sentence, "r")
  l = file.read(1024)
  print("Recieved from client: ", l)

  connectionSocket.send(l.encode())
  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("Enter file name: ")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("From Server: ", filecontents)
clientSocket.close()
```

OUTPUT:


servertcp.py - C:\Users\Neha Chadaga\Desktop\servertcp.py (3.8.3)
File Edit Format Run Options Window Help
clienttcp.py - C:\Users\Neha Chadaga\Desktop\clienttcp.py (3.8.3)
File Edit Format Run Options Window Help
Command Prompt - servertcp.py
Command Prompt

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

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

    connectionS
    file.close(
    connectionS
```

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("Enter file name: ")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("From Server: ", filecontents)
clientSocket.close()
```

```
C:\Users\Neha Chadaga\Desktop>servertcp.py
The server is ready to receive
Recieved from client:  hello
```

```
C:\Users\Neha Chadaga\Desktop>clienttcp.py
Enter file name: sample.txt
From Server:  hello

C:\Users\Neha Chadaga\Desktop>
```

6	4	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:

SERVER:

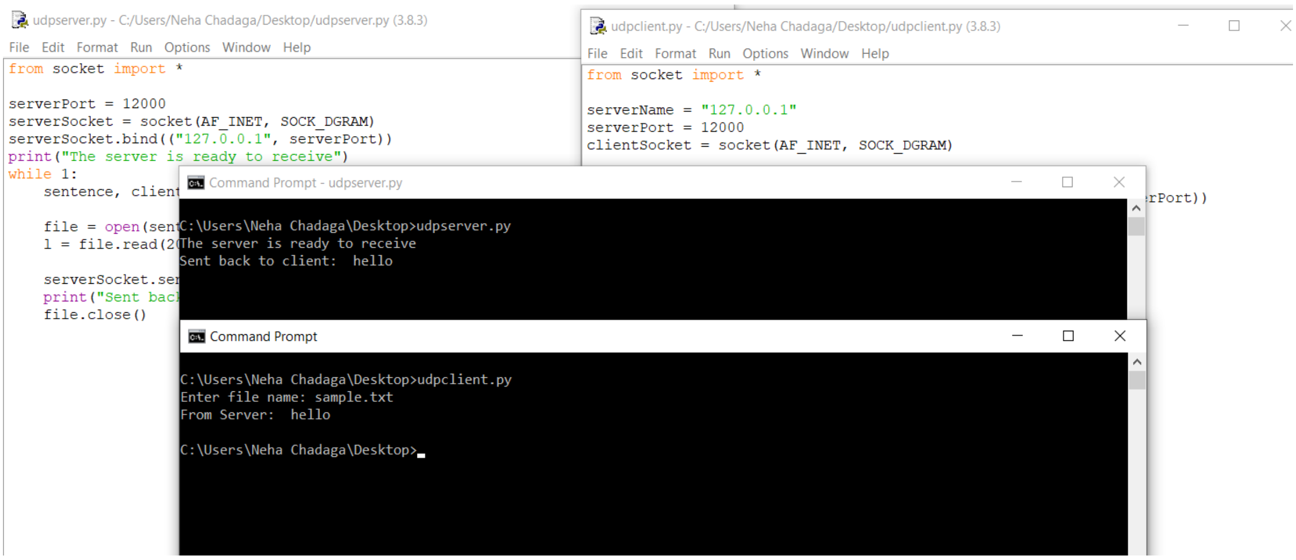
```
from socket import *

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)

   file = open(sentence, "r")
   l = file.read(2048)

   serverSocket.sendto(bytes(l, "utf-8"), clientAddress)
   print("Sent back to client: ", l)
   file.close()
```

CLIENT:

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("Enter file name: ")
```

```
clientSocket.sendto(bytes(sentence, "utf-8"), (serverName, serverPort))
filecontents, serverAddress = clientSocket.recvfrom(2048)
print("From Server: ", filecontents.decode())
clientSocket.close()
```

OUTPUT:


udpserver.py - C:/Users/Neha Chadaga/Desktop/udpserver.py (3.8.3)
File Edit Format Run Options Window Help
udpclient.py - C:/Users/Neha Chadaga/Desktop/udpclient.py (3.8.3)
File Edit Format Run Options Window Help
Command Prompt - udpserver.py
Command Prompt

```
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, client

    file = open(sen
    l = file.read(2

    serverSocket.se
    print("Sent bac
    file.close()
```

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

rPort))
```

```
C:\Users\Neha Chadaga\Desktop>udpserver.py
The server is ready to receive
Sent back to client:  hello
```

```
C:\Users\Neha Chadaga\Desktop>udpclient.py
Enter file name: sample.txt
From Server:  hello

C:\Users\Neha Chadaga\Desktop>
```