PART-B

Program 14

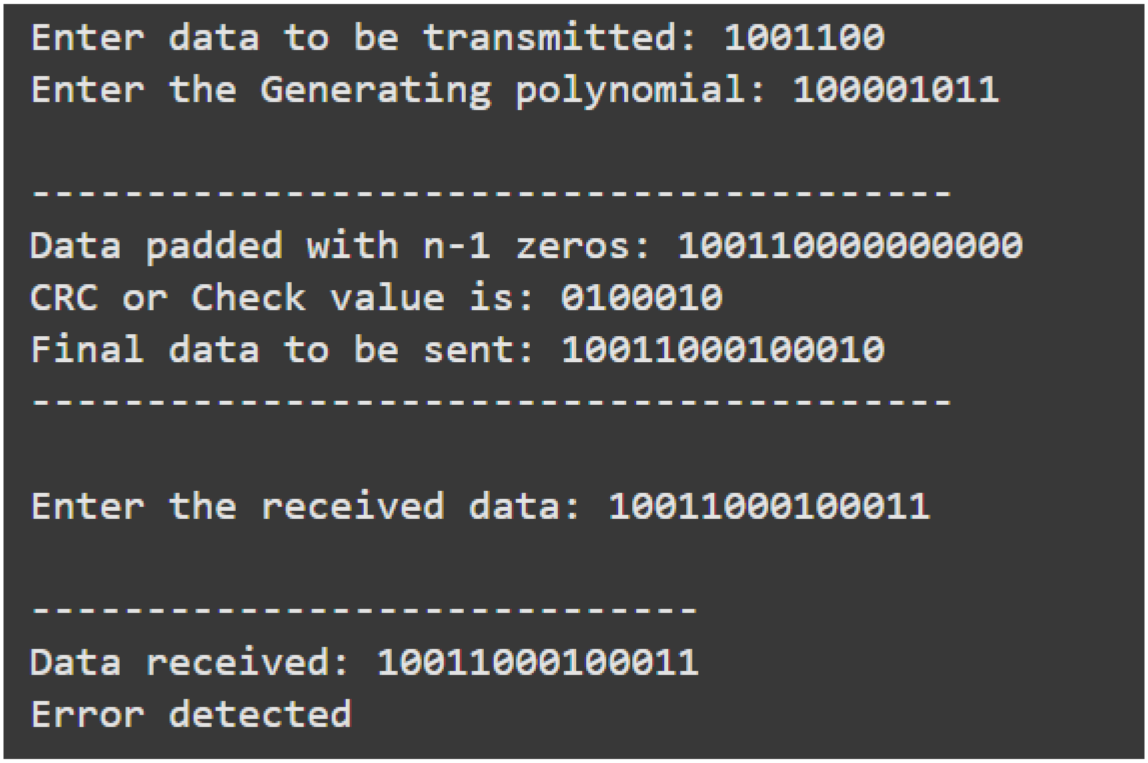
Write a program for error detecting code using CRC-CCITT (16-bits).

Code :

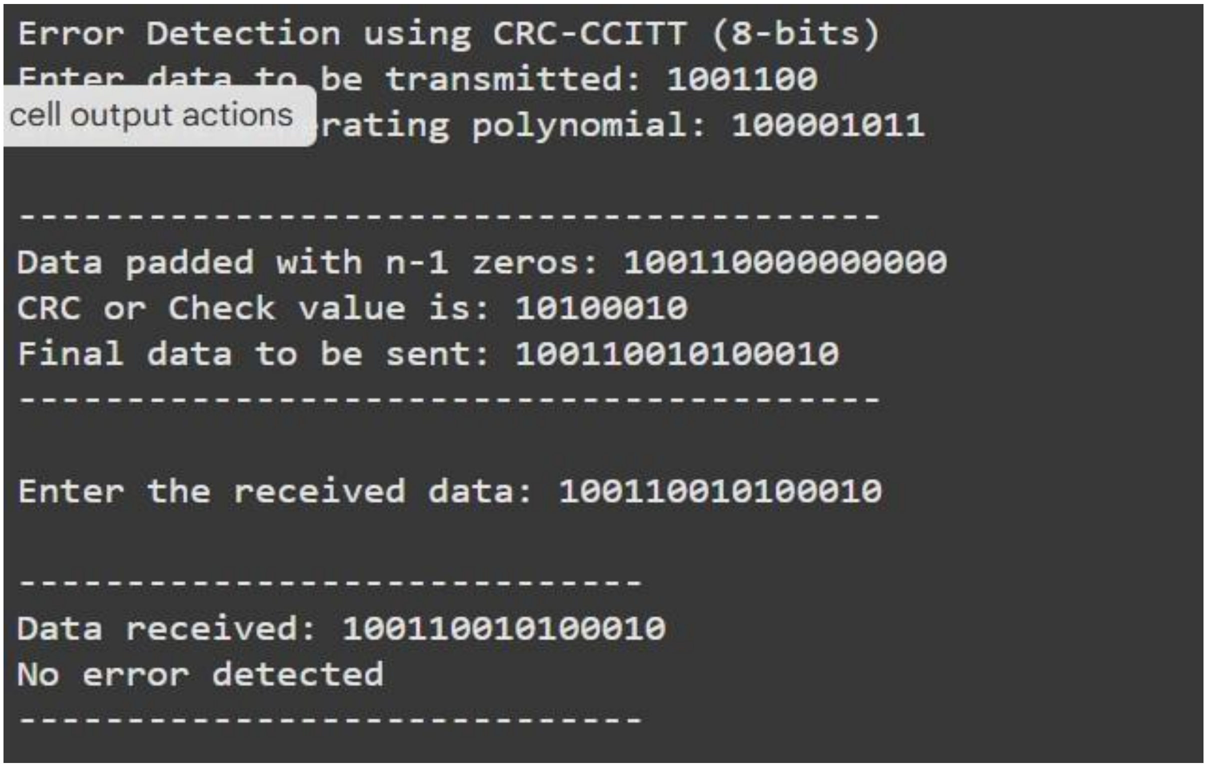
```
def xor(a, b):
    # XOR operation between two binary strings
    result = []
    for i in range(1, len(b)):
        result.append('0' if a[i] == b[i] else '1') return
    ''.join(result)

def mod2div(dividend, divisor): #
    Performs Modulo-2 division
    pick = len(divisor)
    tmp = dividend[: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

    # For the last set of bits if
    tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)

    return tmp

def encode_data(data, key): #
    Encode data with CRC
    l_key = len(key)
    padded_data = data + '0' * (l_key - 1)
    remainder = mod2div(padded_data, key)
    codeword = data + remainder
    return codeword, remainder

def check_data(received_data, key): #
    Check received data for errors
    remainder = mod2div(received_data, key)
    return '0' * (len(key) - 1) == remainder

# Main program
```

```
    if __name__== "__main__":
      print("Error Detection using CRC-CCITT (8-bits)")

# Transmitter
      data = input("Enter data to be transmitted: ").strip()
      key = input("Enter the Generating polynomial: ").strip()

      print("\n-----------------------------------------      ")
      padded_data = data + '0' * (len(key) - 1) print("Data
      padded with n-1 zeros:", padded_data)

      encoded_data, crc = encode_data(data, key)
      print("CRC or Check value is:", crc)
      print("Final data to be sent:", encoded_data)
      print("-----------------------------------------      ")

      # Receiver
      received_data = input("\nEnter the received data: ").strip()
      print("\n-----------------------------     ")
      print("Data received:", received_data)

      if check_data(received_data, key): print("No
        error detected")
      else:
        print("Error detected")
      print("-----------------------------     ")
```

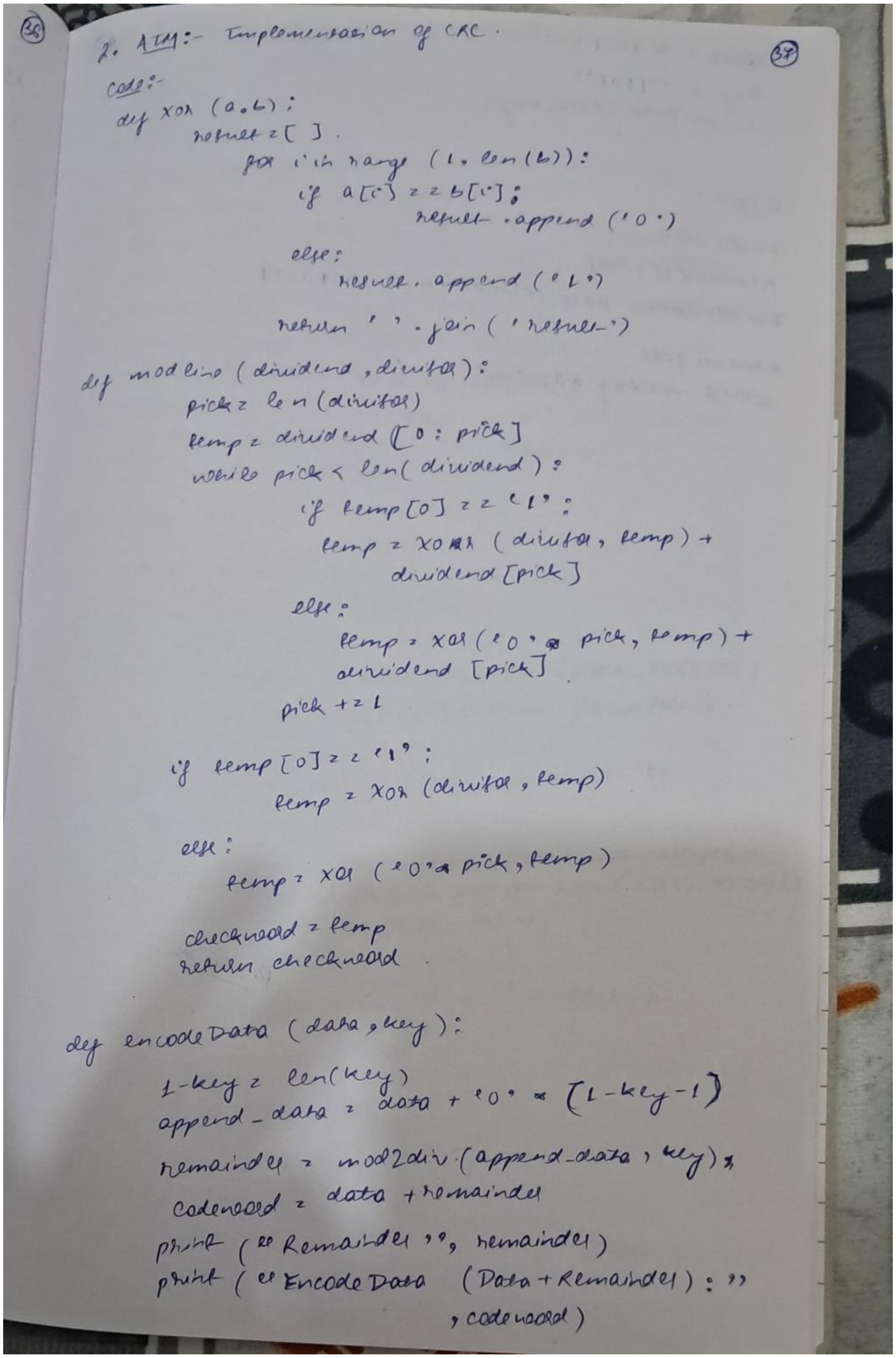
Output



```
Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011


----------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010
----------------------------------------


Enter the received data: 10011000100011


-----------------------------
Data received: 10011000100011
Error detected
```



```
Error Detection using CRC-CCITT (8-bits)
Enter data to be transmitted: 1001100
                rating polynomial: 100001011


----------------------------------------
Data padded with n-1 zeros: 100110000000000
CRC or Check value is: 10100010
Final data to be sent: 100110010100010
----------------------------------------


Enter the received data: 100110010100010


-----------------------------
Data received: 100110010100010
No error detected
-----------------------------
```



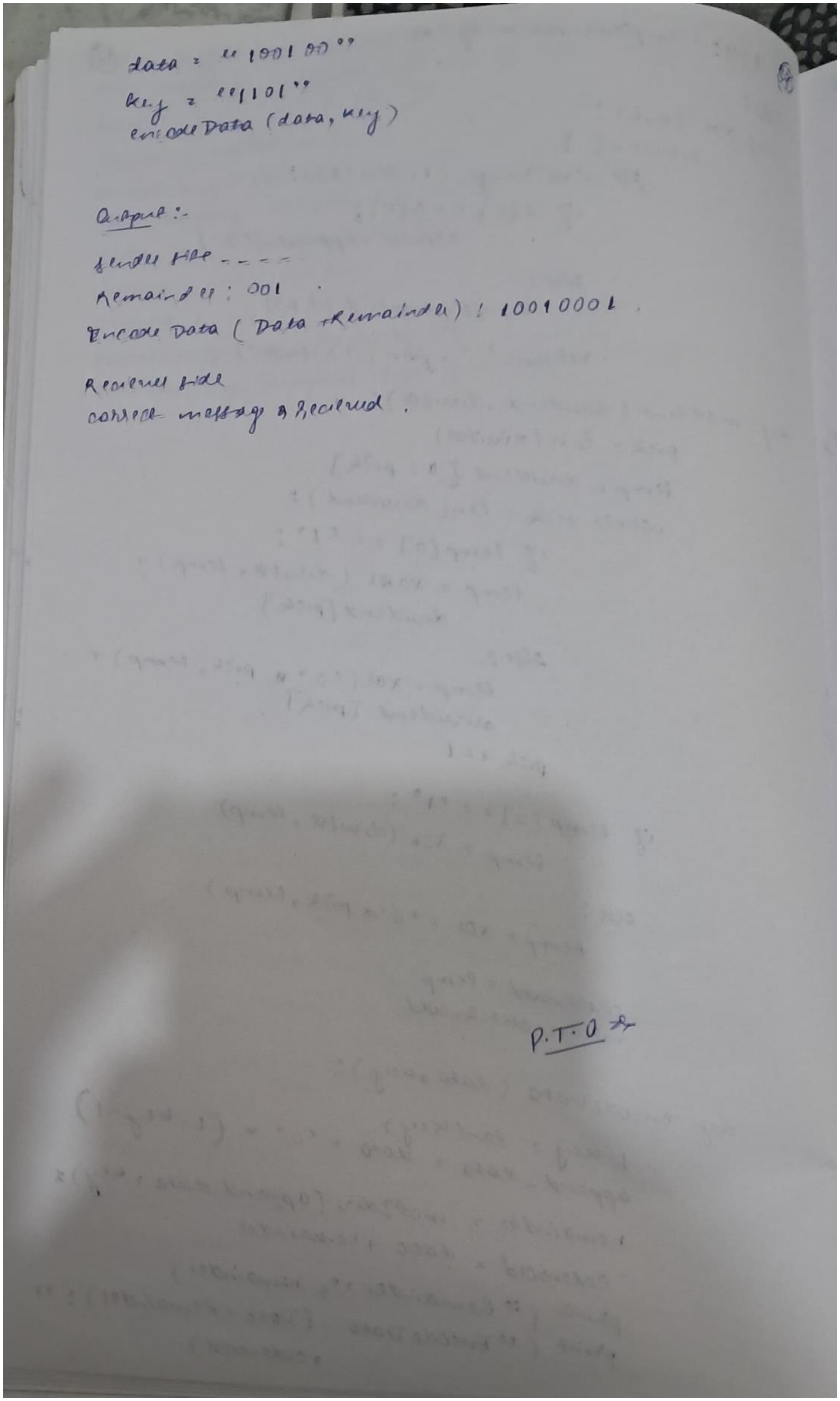
36

2. AIM:- Implementation of CRC.

37

Code:-

```
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)
        temp = dividend [0 : pick]
        while pick < len( dividend ):
                if temp[0] == '1':
                    temp = xor (divisor, temp) +
                        dividend [pick]
                else:
                    temp = xor ('0' * pick, temp) +
                    dividend [pick]
                pick += 1

        if temp [0] == '1':
                temp = xor (divisor, temp)

        else:
            temp = xor ('0'* pick, temp)

        checkword = temp
        return checkword

def encodeData (data, key):

        l-key = len(key)
        append_data = data + '0' * (l-key-1)
        remainder = mod2div (append_data, key);
        codeword = data + remainder
        print ("Remainder", remainder)
        print ("Encode Data (Data + Remainder) : "
                                    , codeword)
```



```
data = "100100"
key = "1101"
encodeData (data, key)
```

Output :-

```
Sender side ----
Remainder : 001 .
Encode Data ( Data + Remainder) : 10010001 .

Receiver side
correct message is received .
```

P.T.O

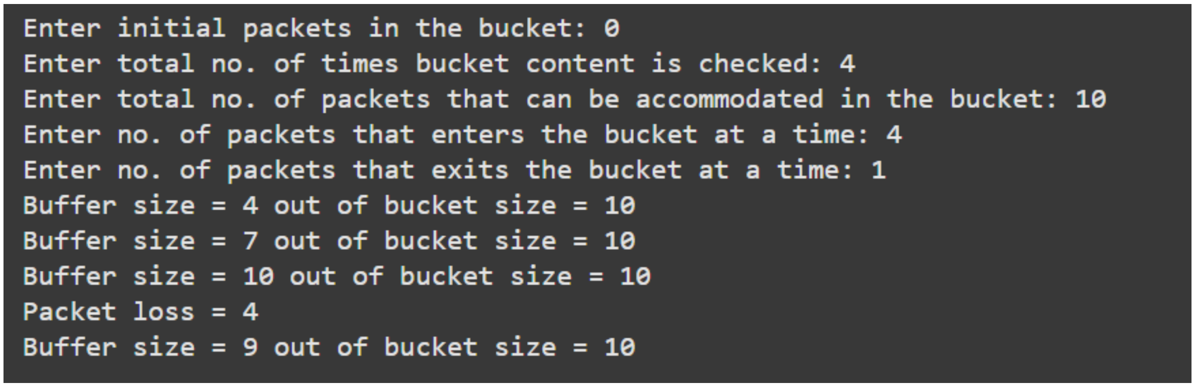
Program 15

Write a program for congestion control using Leaky bucket algorithm.

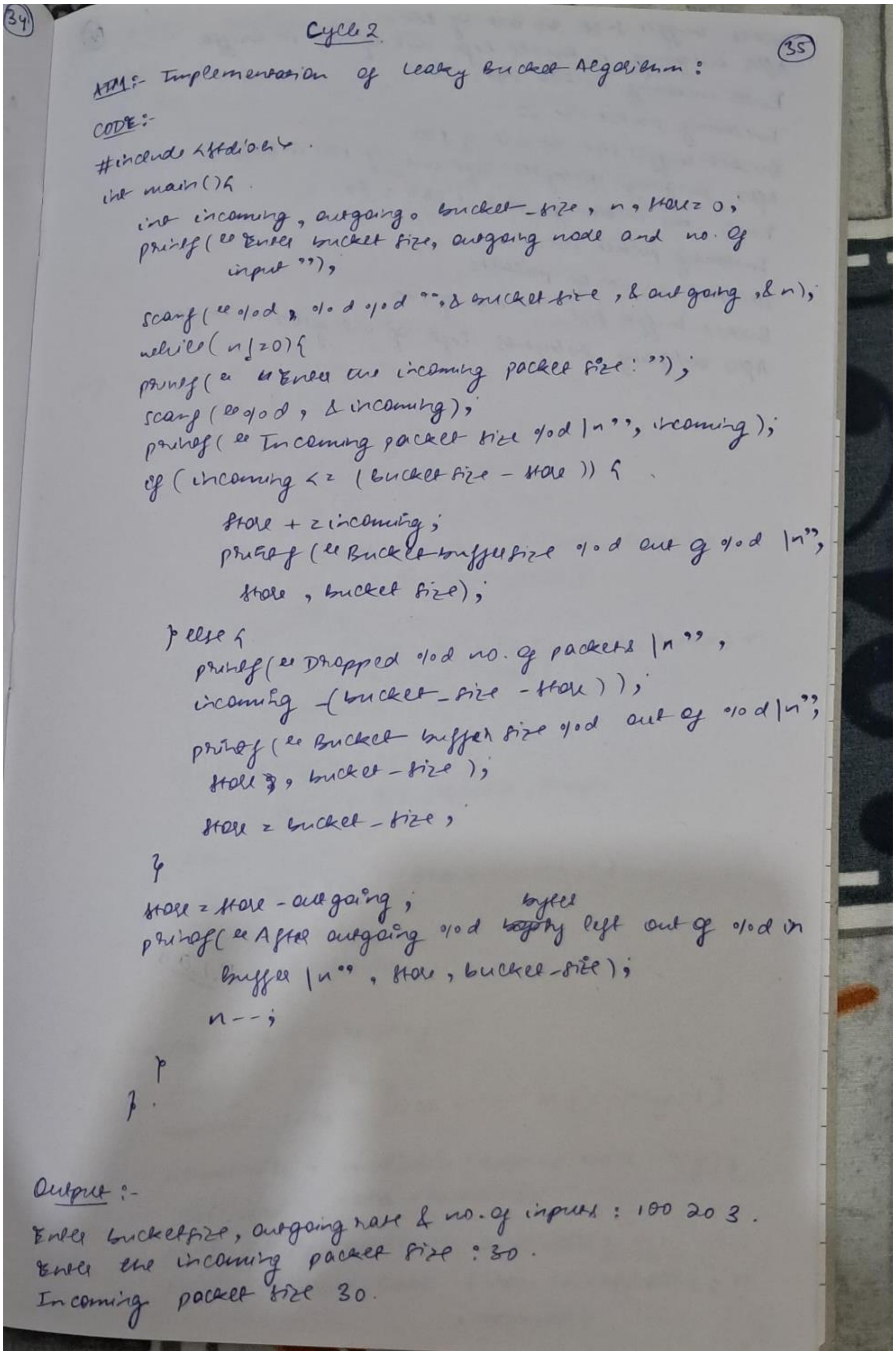
Code :

```
# Getting user inputs
storage = int(input("Enter initial packets in the bucket: "))
no_of_queries = int(input("Enter total no. of times bucket content is checked: ")) bucket_size
= int(input("Enter total no. of packets that can be accommodated in the bucket: "))
input_pkt_size = int(input("Enter no. of packets that enters the bucket at a time: "))
output_pkt_size = int(input("Enter no. of packets that exits the bucket at a time: "))

for i in range(no_of_queries):  # space left
    size_left = bucket_size - storage
    if input_pkt_size <= size_left: #
        update storage
        storage += input_pkt_size
    else:
        print("Packet loss =", input_pkt_size)

    print(f"Buffer size = {storage} out of bucket size = {bucket_size}")

    # as packets are sent out into the network, the size of the storage decreases storage
    -= output_pkt_size
```

Output



```
Enter initial packets in the bucket: 0
Enter total no. of times bucket content is checked: 4
Enter total no. of packets that can be accommodated in the bucket: 10
Enter no. of packets that enters the bucket at a time: 4
Enter no. of packets that exits the bucket at a time: 1
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```



(34)

Cycle 2

(35)

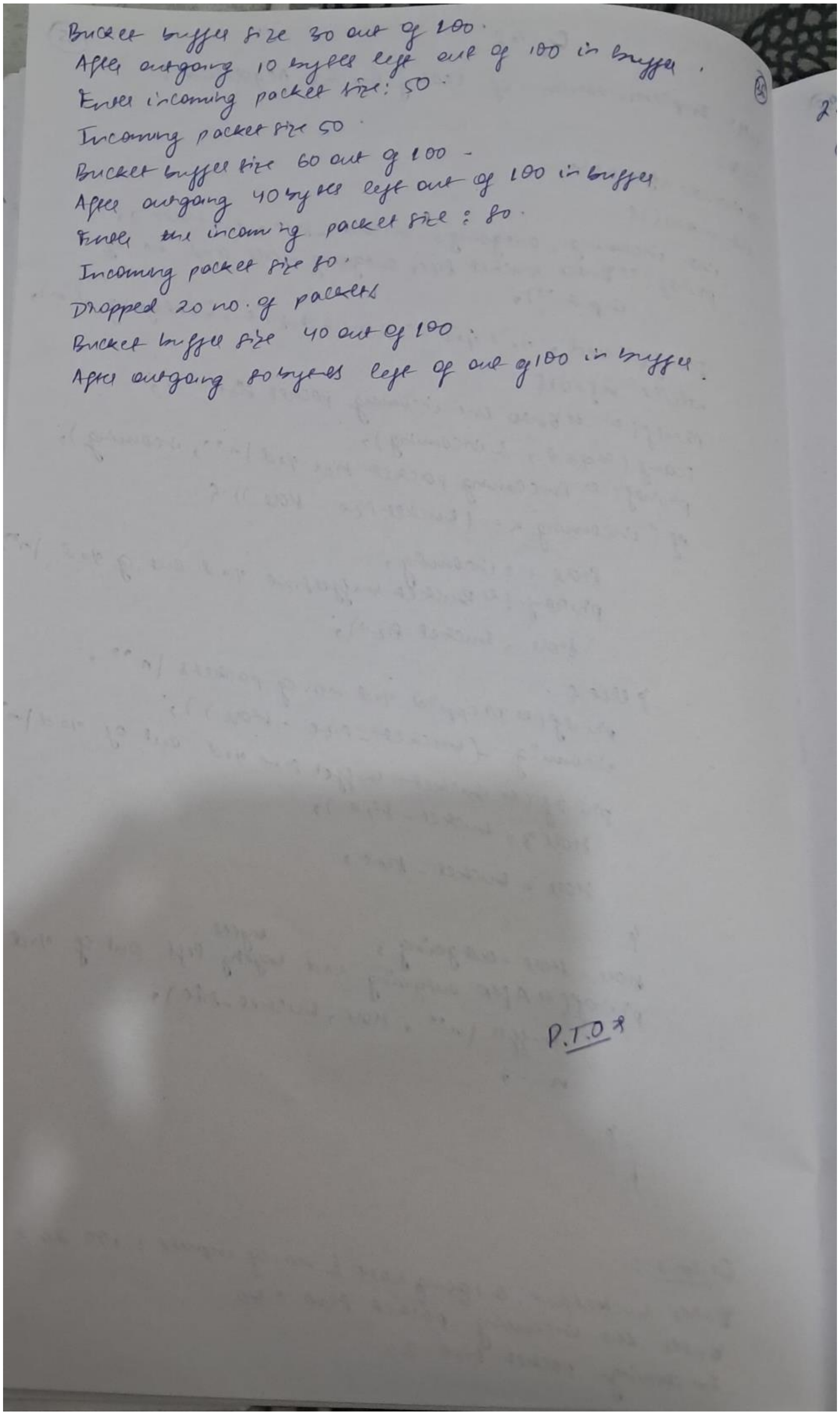
AIM:- Implementation of Leaky Bucket Algorithm :

CODE:-

```
#include <stdio.h>.
int main(){
    int incoming, outgoing, bucket_size, n, store=0;
    printf("Enter bucket size, outgoing rate and no. of
            input"),

    scanf("%d, %d %d", & bucket size, & outgoing, &n);
    while(n!=0){
    printf(" "Enter the incoming packet size:");
    scanf("%d, & incoming),
    printf(" Incoming packet size %d \n", incoming);
    if (incoming <= (bucket size - store)) {

          store += incoming;
          printf("Bucket buffer size %d out of %d \n",
           store, bucket size);

     } else {
       printf(" Dropped %d no. of packets \n",
       incoming -(bucket_size - store));
       printf(" Bucket buffer size %d out of %d\n";
        store, bucket - size);

       store = bucket_size,

    }
    store = store - outgoing;
    printf(" After outgoing %d bytes left out of %d in
          buffer \n", store, bucket-size);
          n--;

    }
}.
```

Output :-

```
Enter bucketsize, outgoing rate & no. of inputs : 100 20 3.
Enter the incoming packet size : 30.
Incoming packet size 30.
```



```
Bucket buffer size 30 out of 100.
After outgoing 10 bytes left out of 100 in buffer.
Enter incoming packet size: 50.
Incoming packet size 50.
Bucket buffer size 60 out of 100.
After outgoing 40 bytes left out of 100 in buffer
Enter the incoming packet size: 80.
Incoming packet size 80.
Dropped 20 no. of packets
Bucket buffer size 40 out of 100.
After outgoing 80 bytes left of out of 100 in buffer.
```

P.T.O

Program 16

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.

Code:

Client.py

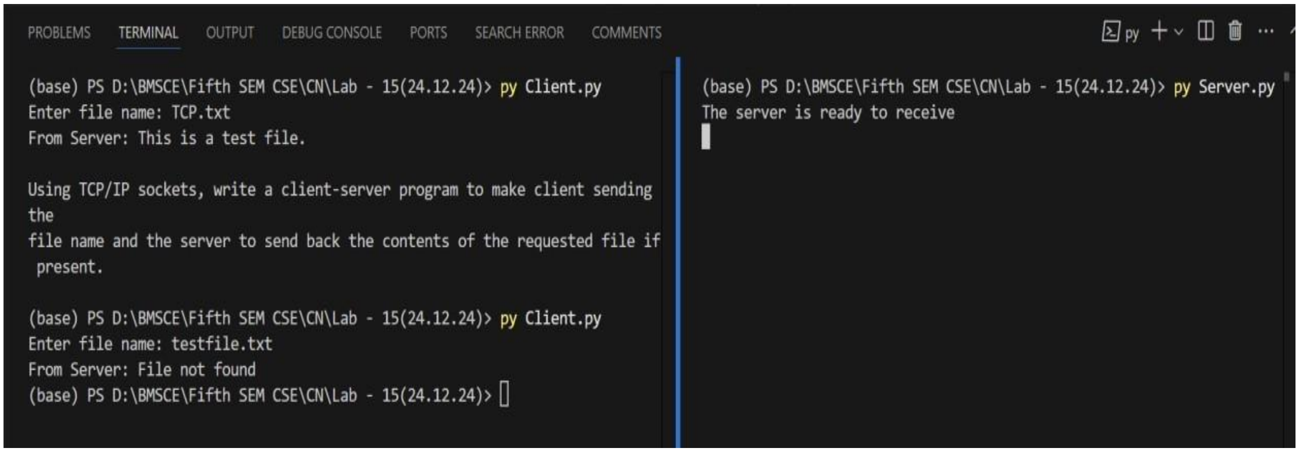
```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number where the server listens

# Create TCP socket
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))  # Connect to server

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send file name to server
clientSocket.send(sentence.encode())

# Receive file contents from server
filecontents = clientSocket.recv(1024).decode()
print('From Server:', filecontents)

# Close the connection
clientSocket.close()
```

Server.py

```
from socket import *
serverName = "127.0.0.1"  # Server address (localhost)
serverPort = 12000  # Port number to listen on

# Create TCP socket
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))  # Bind socket to the address and port
serverSocket.listen(1)  # Listen for 1 connection
print("The server is ready to receive")

while True:
   # Accept a connection
   connectionSocket, addr = serverSocket.accept()

   # Receive the file name from the client
   sentence = connectionSocket.recv(1024).decode()

   # Try opening the file try:
```

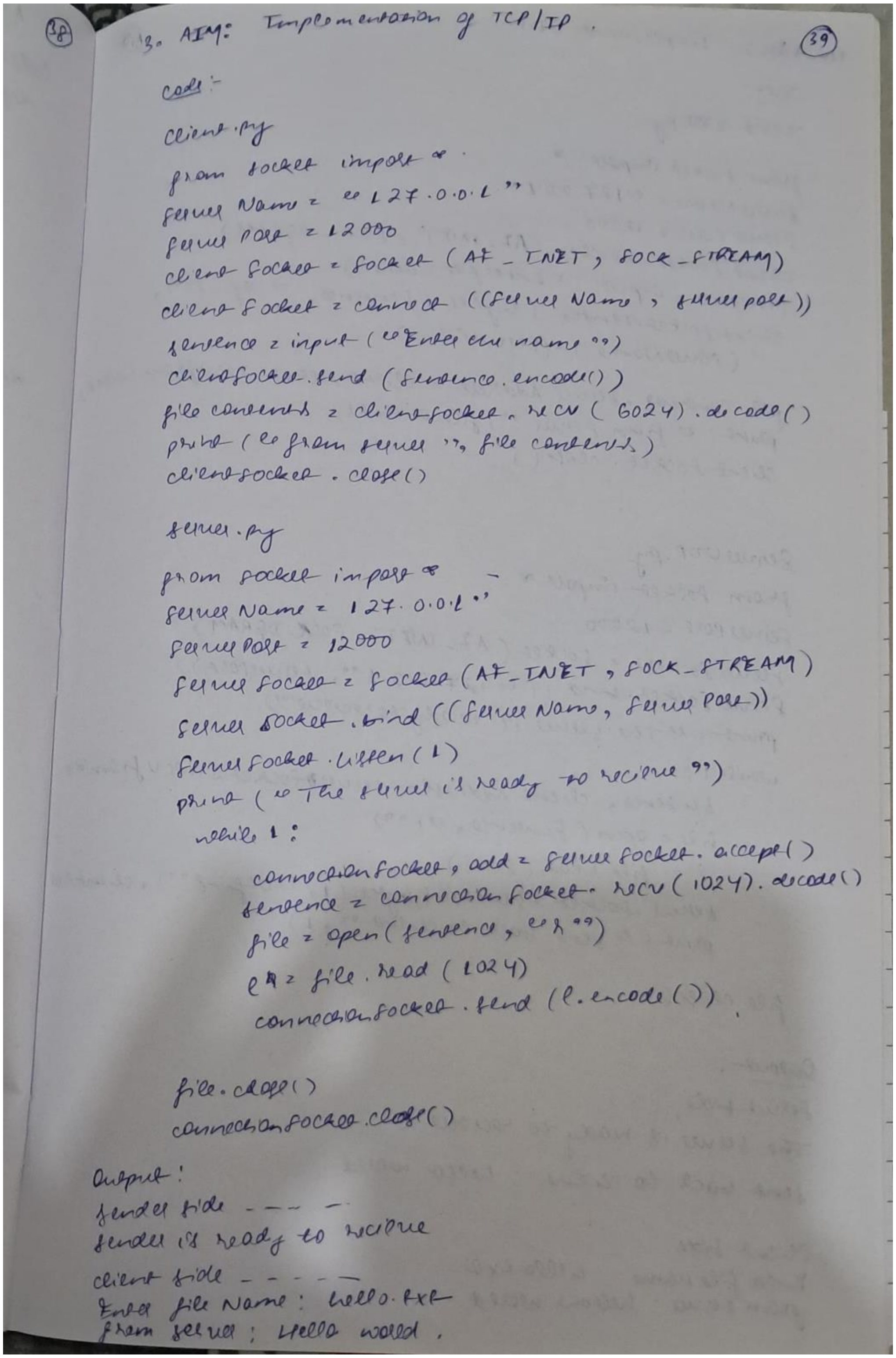
```
file = open(sentence, "r")  # Open file in read mode
            fileContents = file.read(1024)  # Read file content (up to 1024 bytes)
            file.close()
        except FileNotFoundError:
            # Send error message if file not found
            connectionSocket.send("File not found".encode())

        # Close the connection
        connectionSocket.close()
```

Output



```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py Client.py
Enter file name: TCP.txt
From Server: This is a test file.

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.

(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py Client.py
Enter file name: testfile.txt
From Server: File not found
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> 
```

```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py Server.py
The server is ready to receive
```



38

3. AIM: Implementation of TCP/IP.

39

code:-

```
client.py
from socket import *.
serverName = "127.0.0.1"
server port = 12000
client socket = socket (AF_INET, SOCK_STREAM)
client socket = connect ((server Name , server port))
sentence = input ("Enter the name")
clientsocket.send (sentence.encode())
file contents = client socket. recv ( 6024). decode()
print ("from server ", file contents)
clientsocket . close()

server.py
from socket import *
server Name = 127.0.0.1"
server Port = 12000
server socket = socket (AF_INET , SOCK_STREAM)
server socket .bind ((server Name, server Port))
server socket .listen (1)
print ("The server is ready to recieve")
 while 1:
      connectionsocket, addr = server socket. accept()
      sentence = connection socket. recv ( 1024). decode()
      file = open( sentence, "r")
      l = file . read ( 1024)
      connectionsocket . send (l.encode ()) .


   file.close()
   connectionsocket.close()
```

Output:

```
sender side ---- -
sender is ready to recieve

client side -----
Enter file Name : hello.txt
from server ; Hello world .
```

Program 17

Using UDP 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.

Code:

ClientUDP.py

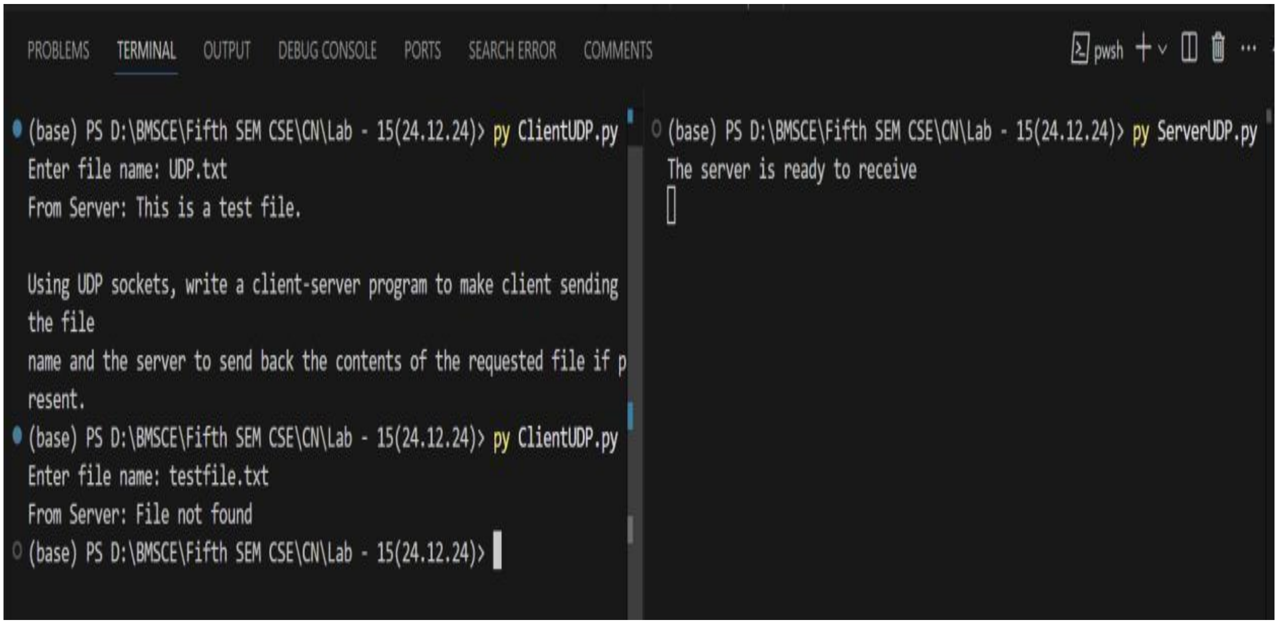
```
from socket import *

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

sentence = input("Enter file name: ")
clientSocket.sendto(sentence.encode(), (serverName, serverPort))

filecontents, serverAddress = clientSocket.recvfrom(2048)
print('From Server:', filecontents.decode())

clientSocket.close()
```

ServerUDP.py

```
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 True:
    sentence, clientAddress = serverSocket.recvfrom(2048)
```

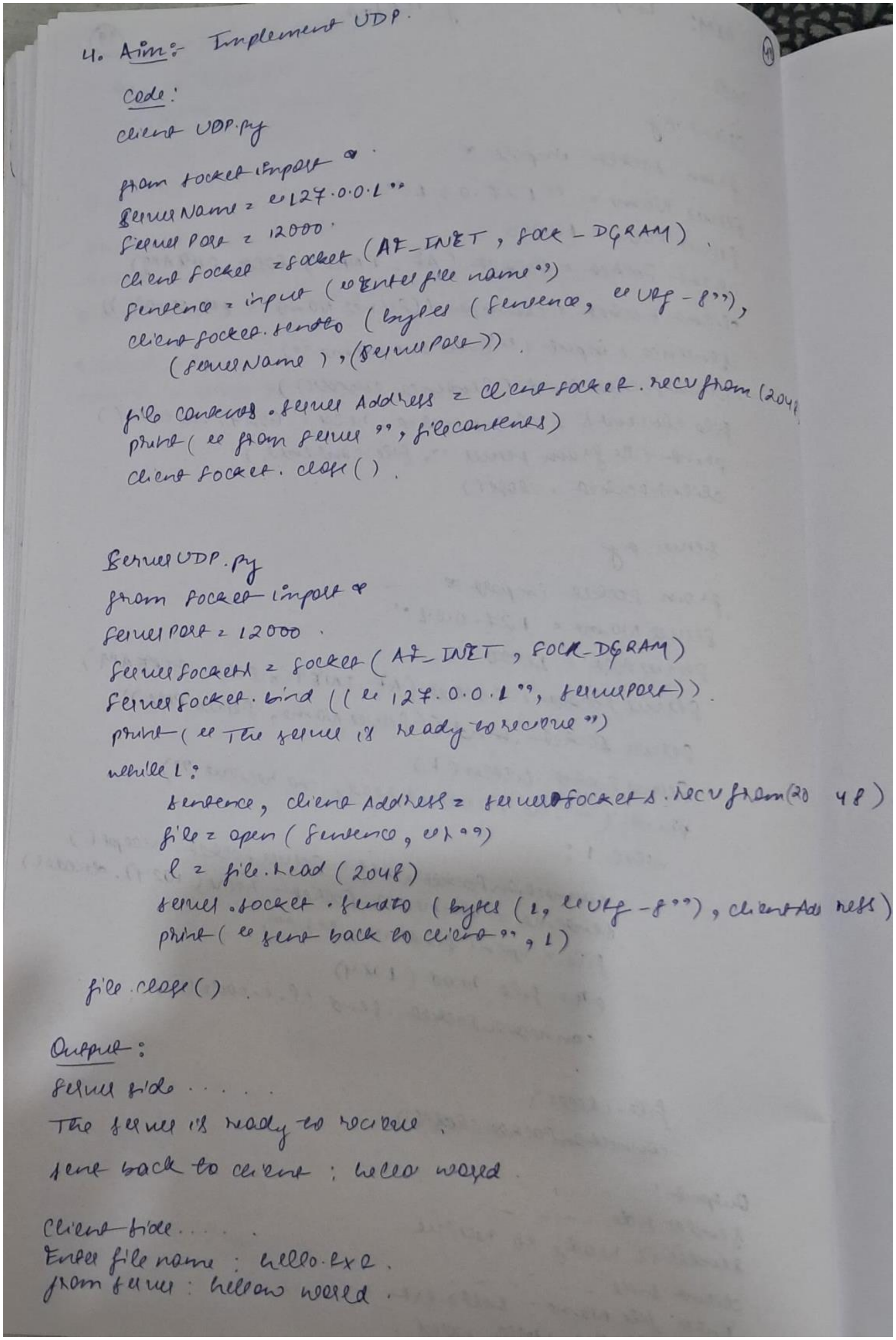
```
try:
    with open(sentence.decode(), "r") as file:
        l = file.read(2048)
        serverSocket.sendto(l.encode(), clientAddress)
        print(f"Sent back to client: {l}")
except FileNotFoundError:
    serverSocket.sendto("File not found.".encode(), clientAddress)
```

Output



```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: UDP.txt
From Server: This is a test file.

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.
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: testfile.txt
From Server: File not found
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)>
```

```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ServerUDP.py
The server is ready to receive
```



4. **Aim:-** Implement UDP.

Code:

client UDP.py

```
from socket import *.
serverName = "127.0.0.1"
serverPort = 12000.
client socket =socket (AF_INET , SOCK_DGRAM) .
sentence = input ("Enter file name")
client socket. sendto (bytes (sentence, "Utf-8"),
    (serverName ),(serverPort)) .

file contents . server Address = client socket. recvfrom (2048
print( "from server ", filecontents)
client socket. close () .


serverUDP.py
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 Address = serverSockets. recvfrom(20 48)
    file = open ( sentence, "r")
    l = file.read ( 2048)
    server .socket . sendto ( bytes (l, "Utf-8"), clientAddress)
    print ( "sent back to client ", l)

file.close () .
```

Output:

```
server side . . . . .
The server is ready to receive .
sent back to client : hello world

Client side . . . .
Enter file name : hello.txe .
from server : hellow world .
```