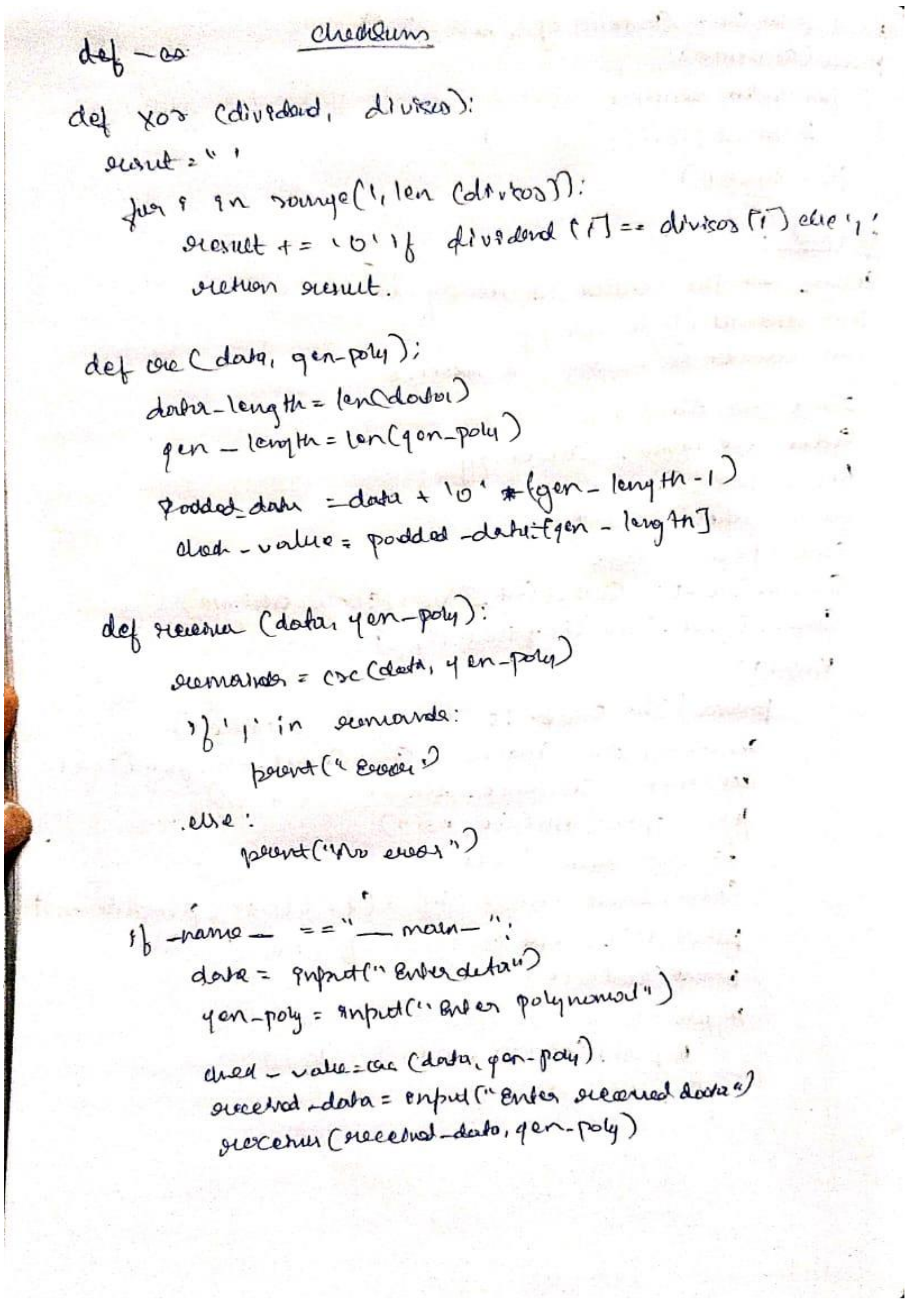
CYCLE-2

PROGRAM1: Write a program for error detecting code using CRCCCITT (16-bits)
OBSERVATION:

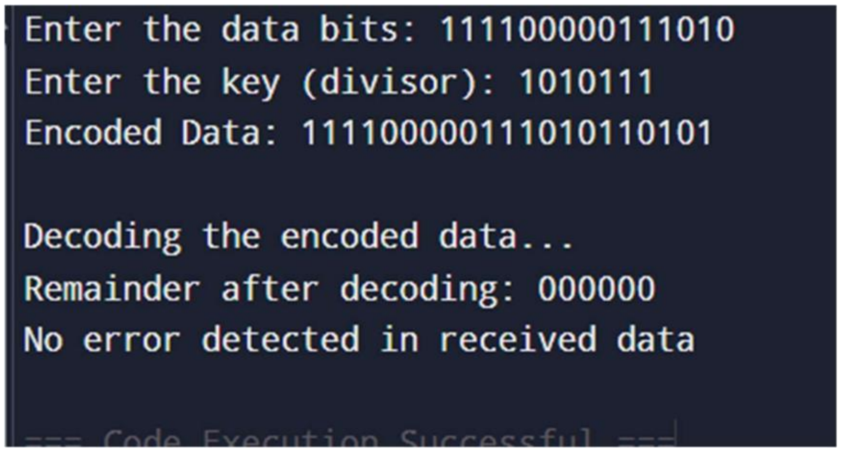


checksum

```
def -cs

def xor (dividend, divisor):
  result = ''
   for i in range(1, len (divisor)):
       result += '0' if dividend [i] == divisor [i] else '1'
       return result.

 def crc (data, gen-poly);
    data_length = len(data)
     gen _ length = len(gen_poly)
     padded_data  = data + '0' *(gen_ length-1)
      check - value = padded -data[gen - length]

  def receiver (data, gen-poly):
     remainder = crc (data, gen-poly)
       if '1' in remainder:
          print(" Error")
      else:
         print("No error")

    if _name_  == "__main__":
       data = input(" Enter data")
       gen-poly = input(" Enter polynomial")
       check - value = crc (data, gen-poly)
       received -data = input("Enter received data")
       receiver (received-data, gen-poly)
```

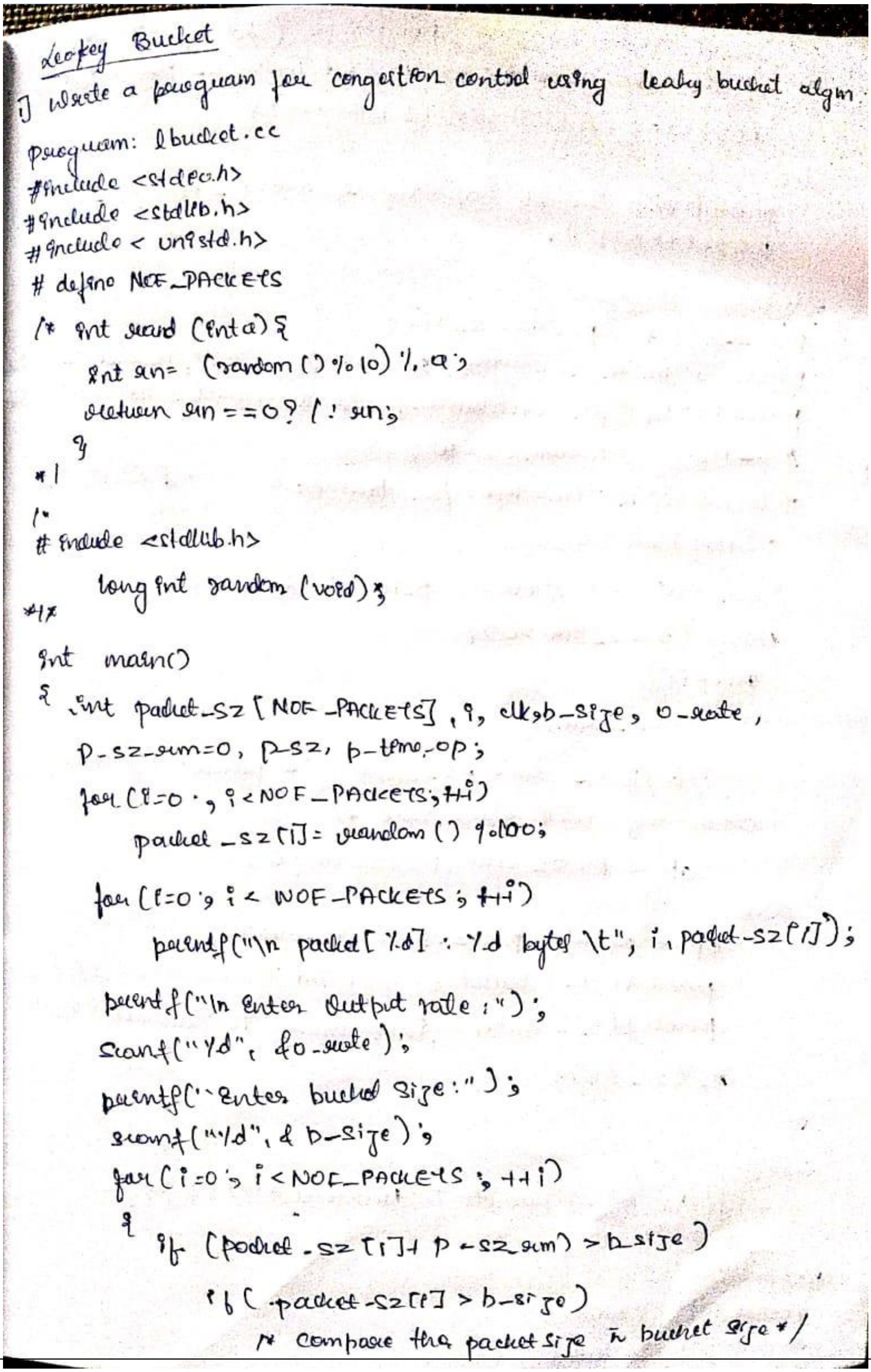
```
        pick += 1

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

    checkword = tmp
return checkword


def encode(data, key):     key_len = len(key)
appended_data = data + '0' * (key_len - 1)
remainder = mod2div(appended_data, key)
codeword = data + remainder
print(f"Encoded Data: {codeword}")
return codeword


def decode(data, key):
    remainder = mod2div(data, key)
print(f"Remainder after decoding: {remainder}")
if '1' not in remainder:
        print("No error detected in received data")
else:
        print("Error detected in received data")


# Main function
if __name__ == "__main__":
    data = input("Enter the data bits: ")
    key = input("Enter the key (divisor): ")

    # Encoding
    encoded_data = encode(data, key)

    # Decoding
    print("\nDecoding the encoded data...")
decode(encoded_data, key)
```

OUTPUT:



```
Enter the data bits: 111100000111010
Enter the key (divisor): 1010111
Encoded Data: 111100000111010110101

Decoding the encoded data...
Remainder after decoding: 000000
No error detected in received data


=== Code Execution Successful ===
```

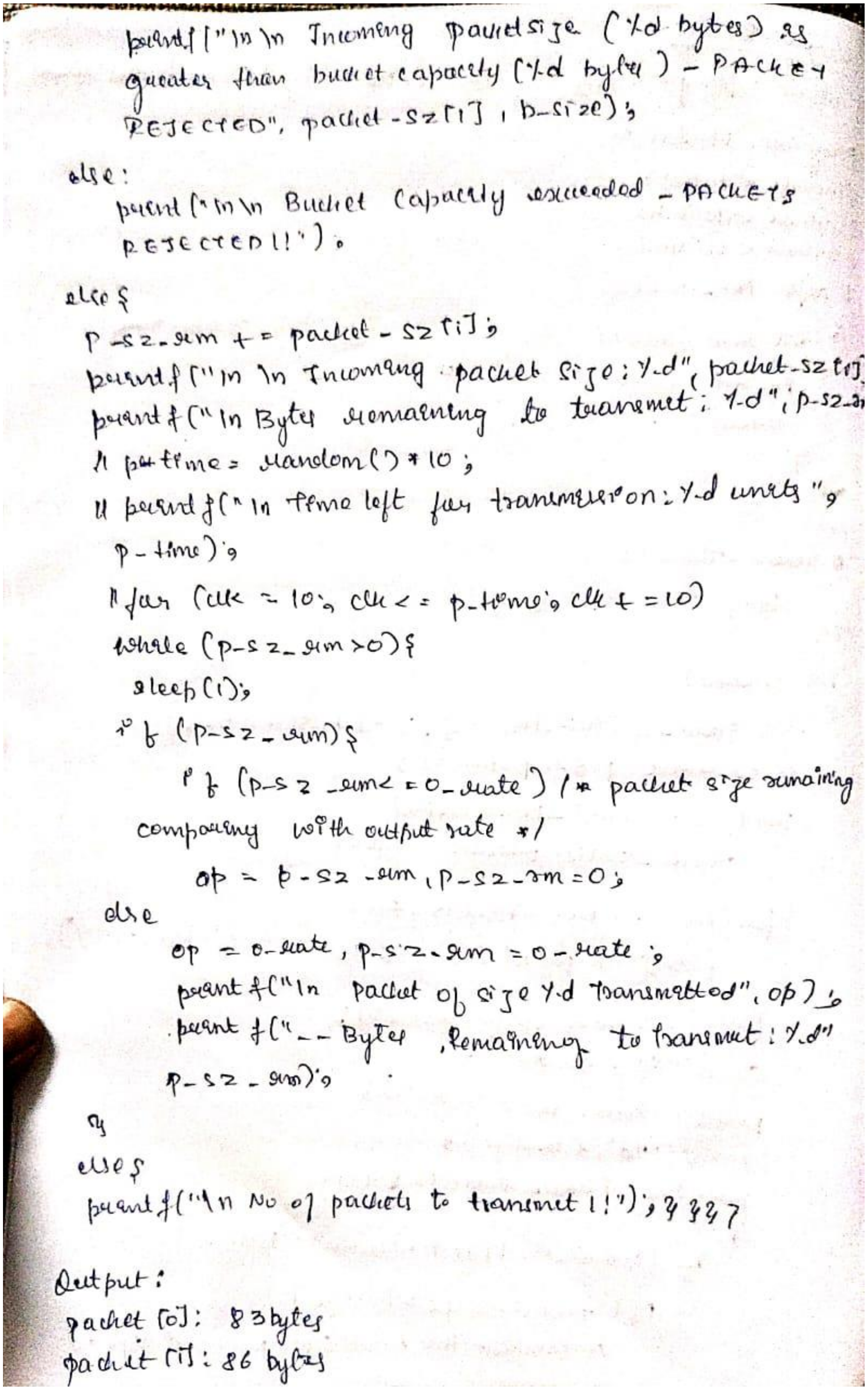
PROGRAM2: Write a program for congestion control using Leaky bucket algorithm.
OBSERVATION:



Leakey Bucket

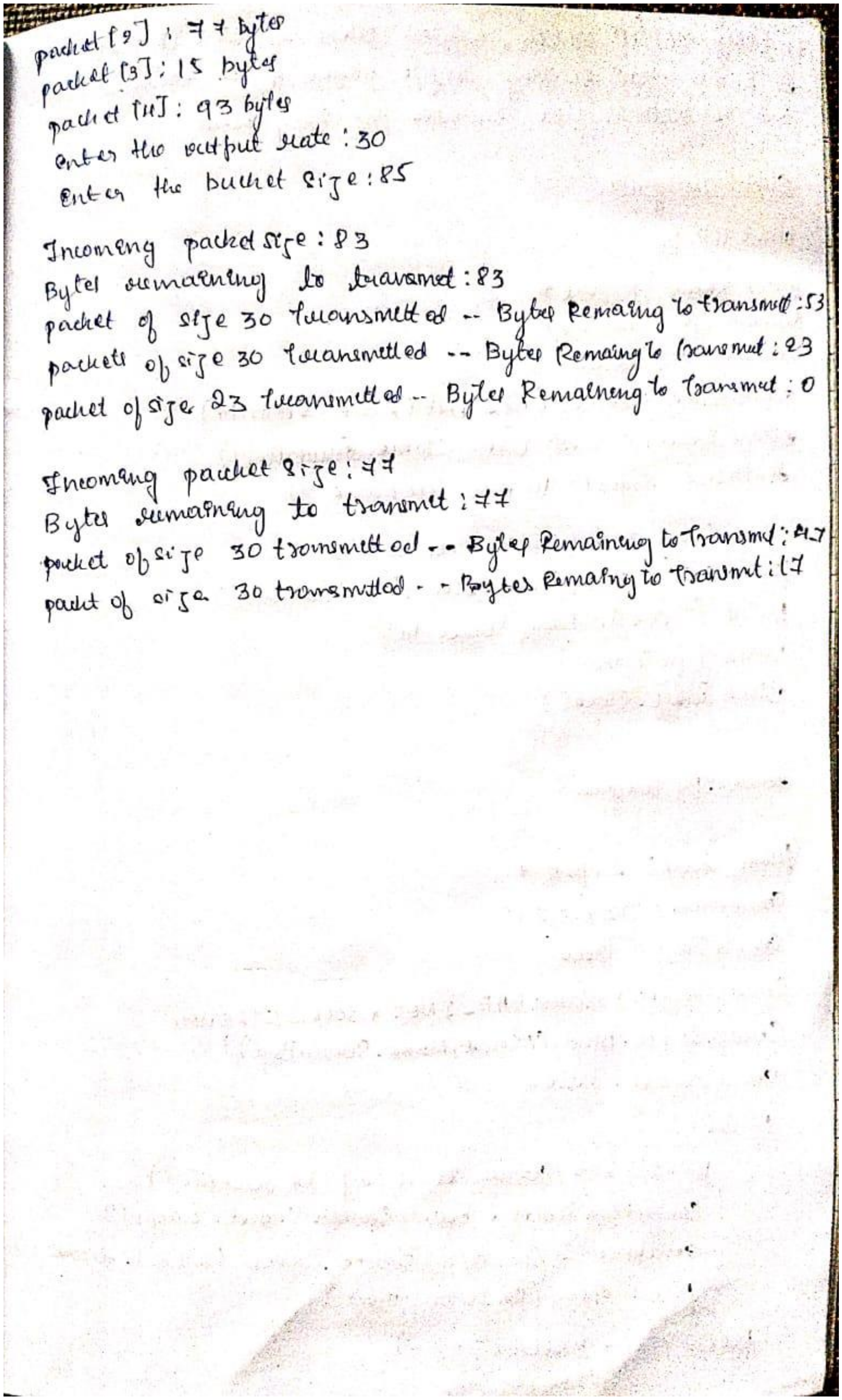
1) Write a program for congestion control using leaky bucket algm.

```
program: lbucket.cc
#include <stdio.h>
#include <stdlib.h>
#include < unistd.h>
# define NOF_PACKETS
/* int rand (int a) {
     int rn= (random () % 10) % a;
     return rn ==0 ? 1 : rn;
   }
 */
/*
# include <stdlib.h>
      long int random (void);
*/
int main()
{ int packet_sz [NOF_PACKETS] , i, clk,b_size, o_rate ,
   p_sz_rm=0, p_sz, p_time,op;
   for (i=0 ; i<NOF_PACKETS; ++i)
        packet_sz [i]= random () %100;
   for (i=0 ; i< NOF_PACKETS ; ++i)
        printf("\n packet [ %d] : %d bytes \t", i, packet_sz[i]);
   printf("\n Enter Output rate :");
   scanf("%d", &o_rate );
   printf("Enter bucket size :" );
   scanf("%d", & b_size );
   for ( i=0 ; i<NOF_PACKETS ; ++i)
   {   if (packet_sz [i]+ p_sz_rm) > b_size )
          if ( packet_sz[i] > b_size )
              /* compare the packet size n bucket size */
```



```
    printf("\n\n Incoming packet size (%d bytes) is
    greater than bucket capacity (%d bytes) - PACKET
    REJECTED", packet-sz[i], b-size);

else:
    print("\n\n Bucket Capacity exceeded - PACKETS
    REJECTED!!");

else {
  p-sz-rm += packet-sz[i];
  printf("\n \n Incoming packet size: %d", packet-sz[i]
  printf("\n Bytes remaining to transmit: %d", p-sz-r
  // p-time = random() * 10;
  // printf("\n time left for transmission: %d units",
   p-time);
  // for (clk = 10; clk <= p-time; clk += 10)
  while (p-sz-rm > 0) {
   sleep(1);
   if (p-sz-rm) {
       if (p-sz-rm < = o-rate) /* packet size remaining
    comparing with output rate */
         op = p-sz-rm, p-sz-rm = 0;
  else
       op = o-rate, p-sz-rm = o-rate;
       printf("\n Packet of size %d Transmitted", op);
       printf("-- Bytes Remaining to transmit: %d"
       p-sz-rm);

 }
 else {
 printf("\n No of packets to transmit!!"); }}}}
```

Output:
packet [0]: 83 bytes
packet [1]: 86 bytes

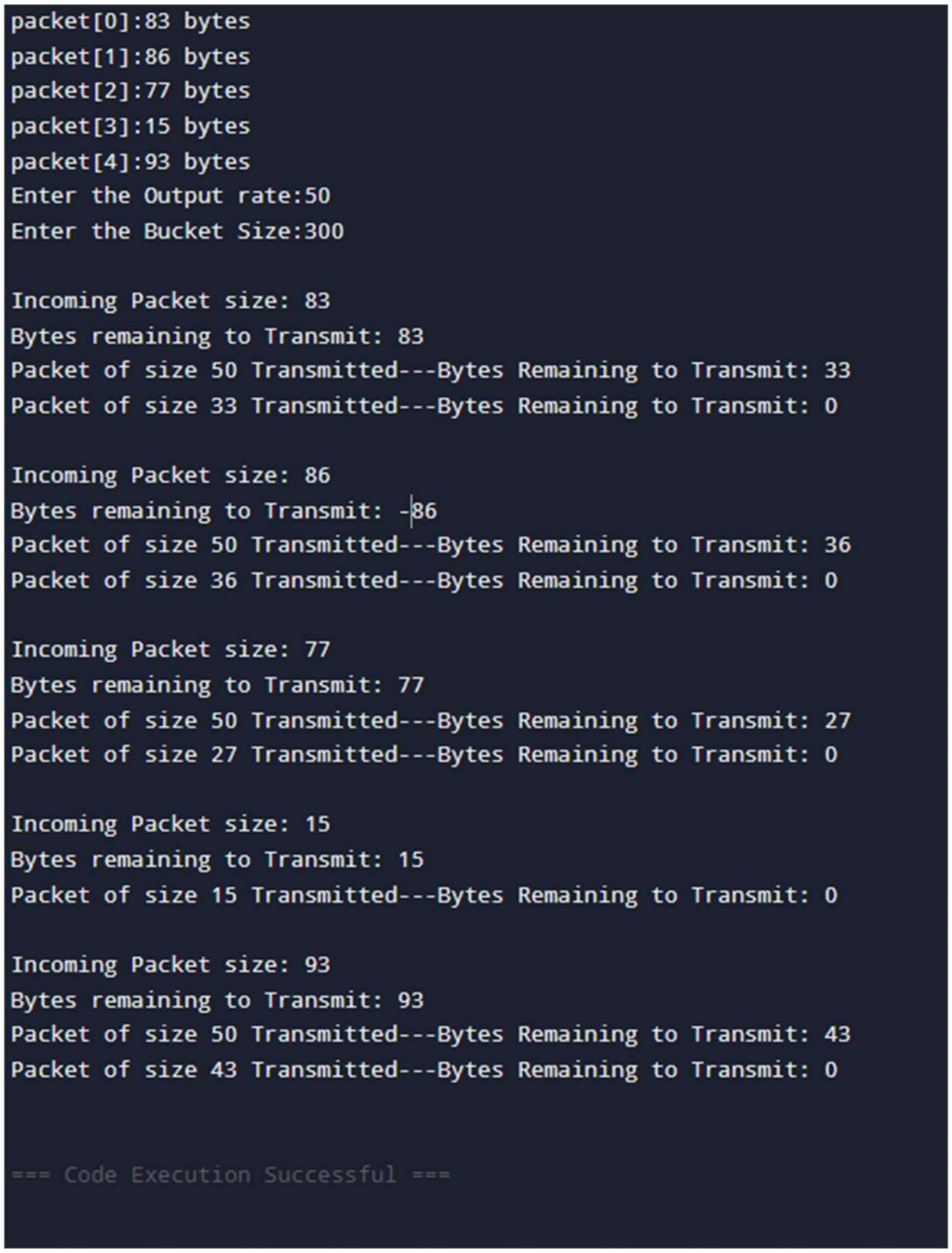


```
packet [2] : 77 bytes
packet [3] : 15 bytes
packet [4] : 93 bytes
enter the output rate : 30
Enter the bucket size : 85

Incoming packet size : 83
Bytes remaining to transmit : 83
packet of size 30 transmitted -- Bytes Remaining to transmit : 53
packets of size 30 transmitted -- Bytes Remaining to transmit : 23
packet of size 23 transmitted -- Bytes Remaining to transmit : 0

Incoming packet size : 77
Bytes remaining to transmit : 77
packet of size 30 transmitted -- Bytes Remaining to transmit : 47
packet of size 30 transmitted -- Bytes Remaining to transmit : 17
```

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // for sleep function
#define NOF_PACKETS 5
// Function to simulate sending packets void send_packet(int
packet_size, int output_rate) {     while (packet_size > 0) {         int
sent = (packet_size < output_rate) ? packet_size : output_rate;
printf("Packet of size %d Transmitted---", sent);         packet_size -=
sent;         printf("Bytes Remaining to Transmit: %d\n",
packet_size);         sleep(1);  // Simulate time delay between packets
    }
}

int main() {
    int output_rate, bucket_size, incoming_packet_size;
    int i, packet_size[NOF_PACKETS];

    // Input number of packets and their sizes
for(i = 0; i < NOF_PACKETS; i++) {
        packet_size[i] = rand() % 100;  // Random packet size between 0 and 99
printf("packet[%d]:%d bytes\n", i, packet_size[i]);
    }

    printf("Enter the Output rate:");
    scanf("%d", &output_rate);

    printf("Enter the Bucket Size:");
    scanf("%d", &bucket_size);

    for(i = 0; i < NOF_PACKETS; i++) {         printf("\nIncoming Packet size: %d\n",
packet_size[i]);         if(packet_size[i] > bucket_size) {             printf("Incoming packet
size (%dbytes) is Greater than bucket capacity (%dbytes)-
PACKET REJECTED\n", packet_size[i], bucket_size);
continue;
        }

        printf("Bytes remaining to Transmit: %d\n", packet_size[i]);
send_packet(packet_size[i], output_rate);
    }
return 0;
}
```

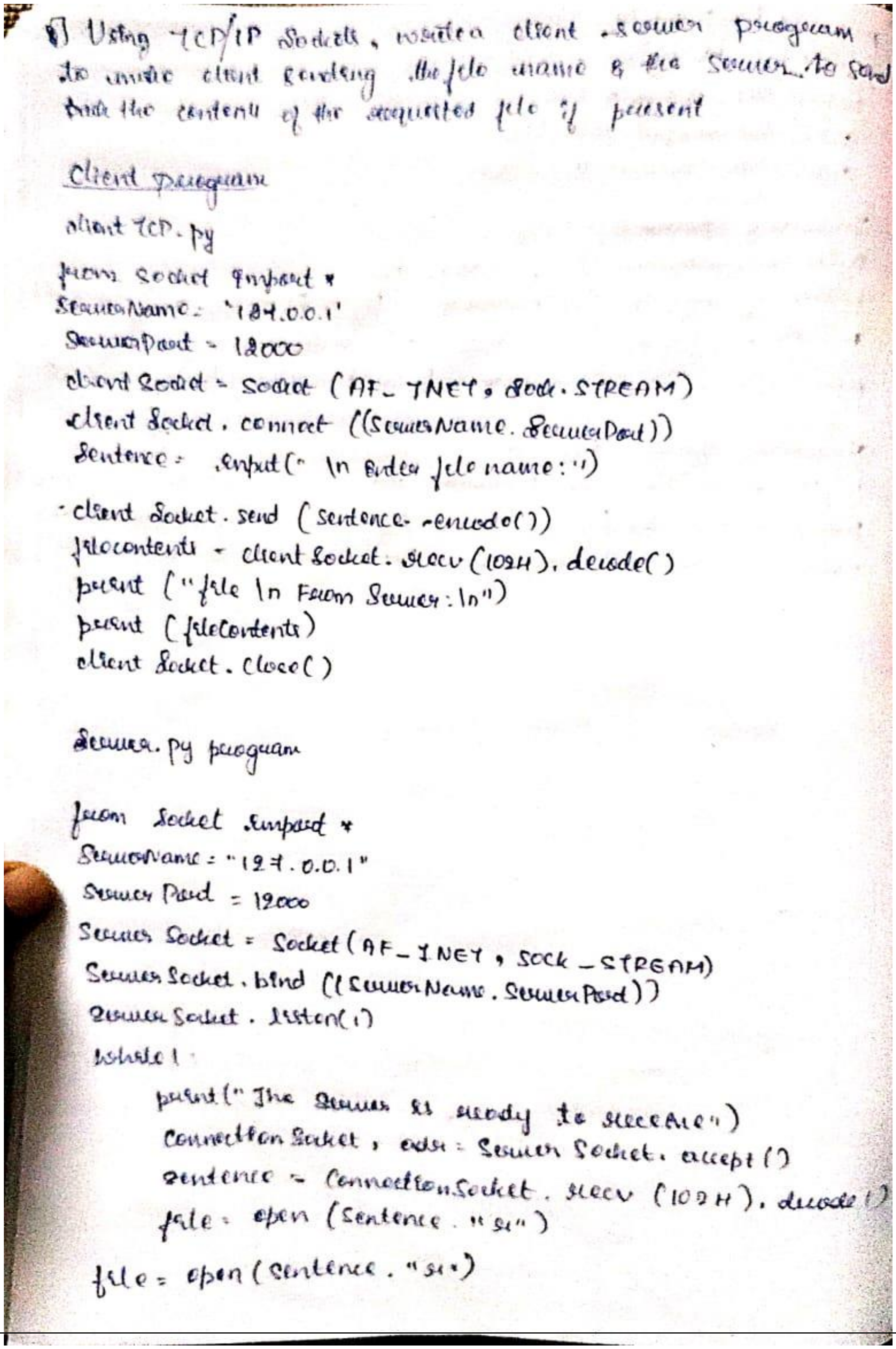
OUTPUT:



```
packet[0]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:50
Enter the Bucket Size:300

Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 33
Packet of size 33 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 86
Bytes remaining to Transmit: -86
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 36
Packet of size 36 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 27
Packet of size 27 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 15
Bytes remaining to Transmit: 15
Packet of size 15 Transmitted---Bytes Remaining to Transmit: 0

Incoming Packet size: 93
Bytes remaining to Transmit: 93
Packet of size 50 Transmitted---Bytes Remaining to Transmit: 43
Packet of size 43 Transmitted---Bytes Remaining to Transmit: 0



=== Code Execution Successful ===
```

PROGRAM3: 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.

OBSERVATION:



8] Using TCP/IP Sockets, write a client-server program to make client sending the file name & the server to send back the contents of the requested file if present

Client program

```
client TCP.py
from socket import *
ServerName = '127.0.0.1'
ServerPort = 12000
clientSocket = socket (AF_INET, Sock.STREAM)
clientSocket.connect ((ServerName. ServerPort))
Sentence = input(" \n Enter file name:")
clientSocket.send (sentence.encode())
filecontents = clientSocket.recv (1024).decode()
print ("file \n From Server:\n")
print (fileContents)
clientSocket.close()
```

Server.py program

```
from socket import *
ServerName = "127.0.0.1"
Server Port = 12000
ServerSocket = Socket(AF_INET, SOCK_STREAM)
ServerSocket.bind ((ServerName. ServerPort))
ServerSocket.listen(1)
while 1:
     print("The server is ready to receive")
     ConnectionSocket, addr = ServerSocket.accept()
     sentence = ConnectionSocket.recv (1024).decode()
     file = open (Sentence. "r")

 file = open(sentence. "r")
```

```
l= file. read(1024)
connectionSocket . send ( l.encode())
print ("\n Sent contents of' + Sentence)
file . close ()
connectionSocket . close ()
```

Output :

```
server -> The server is ready to receive
sent contents of server.py
client -> Enter file name : server. py
```

From Server:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
Server Socket = Socket (AF_INET, SOCK, STREAM)
Server Socket. bind ((Server Name, ServerPort))
Server Socket. listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = ServerSocket.accept()
    sentence = connectionSocket. recv(1024). decode()
    file=open (Sentence, "r")
    l= file. read(1024)
    ConnectionSocket.send ( l . encode())
    print('\n sent contents of' + Sentence)
    file. close()
    ConnectionSocket. close()
```

CODE:

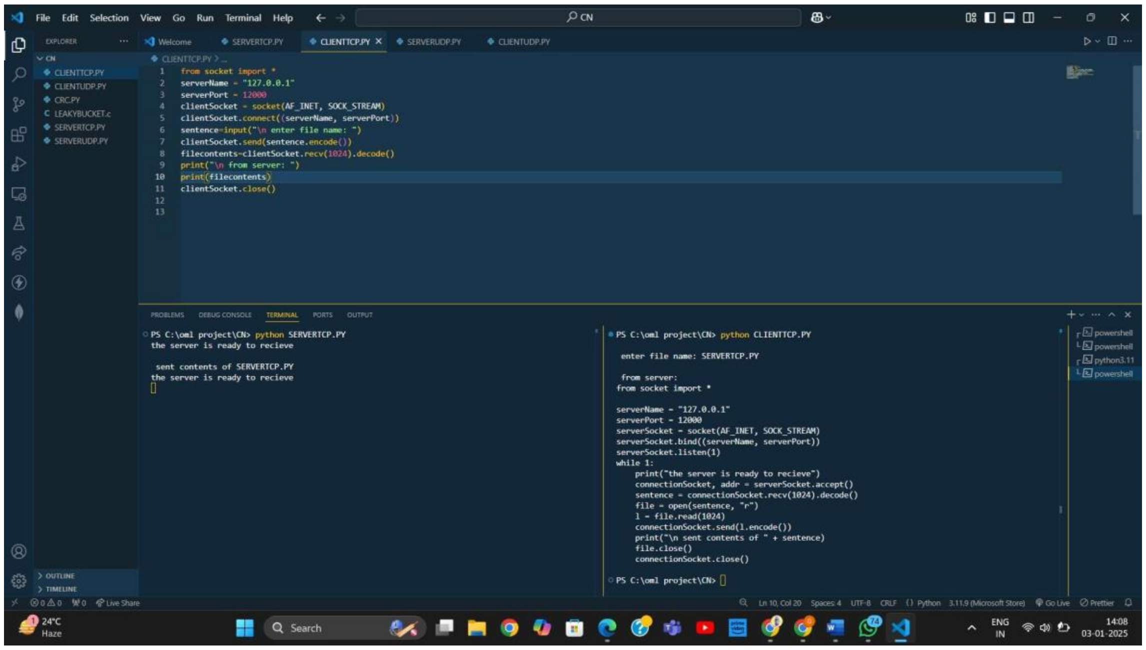
SERVERTCP.PY:

```
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,
SOCK_STREAM) serverSocket.bind((serverName,
serverPort)) serverSocket.listen(1) while 1:
   print("the server is ready to recieve")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
   file = open(sentence, "r")     l =
file.read(1024)
connectionSocket.send(l.encode())
print("\n sent contents of " + sentence)
   file.close()
   connectionSocket.close()
```

CLIENTTCP.PY:

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET,
SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence=input("\n enter file name: ")
clientSocket.send(sentence.encode())
filecontents=clientSocket.recv(1024).decode()
print("\n from server: ")
print(filecontents)
clientSocket.close()
```

OUTPUT:

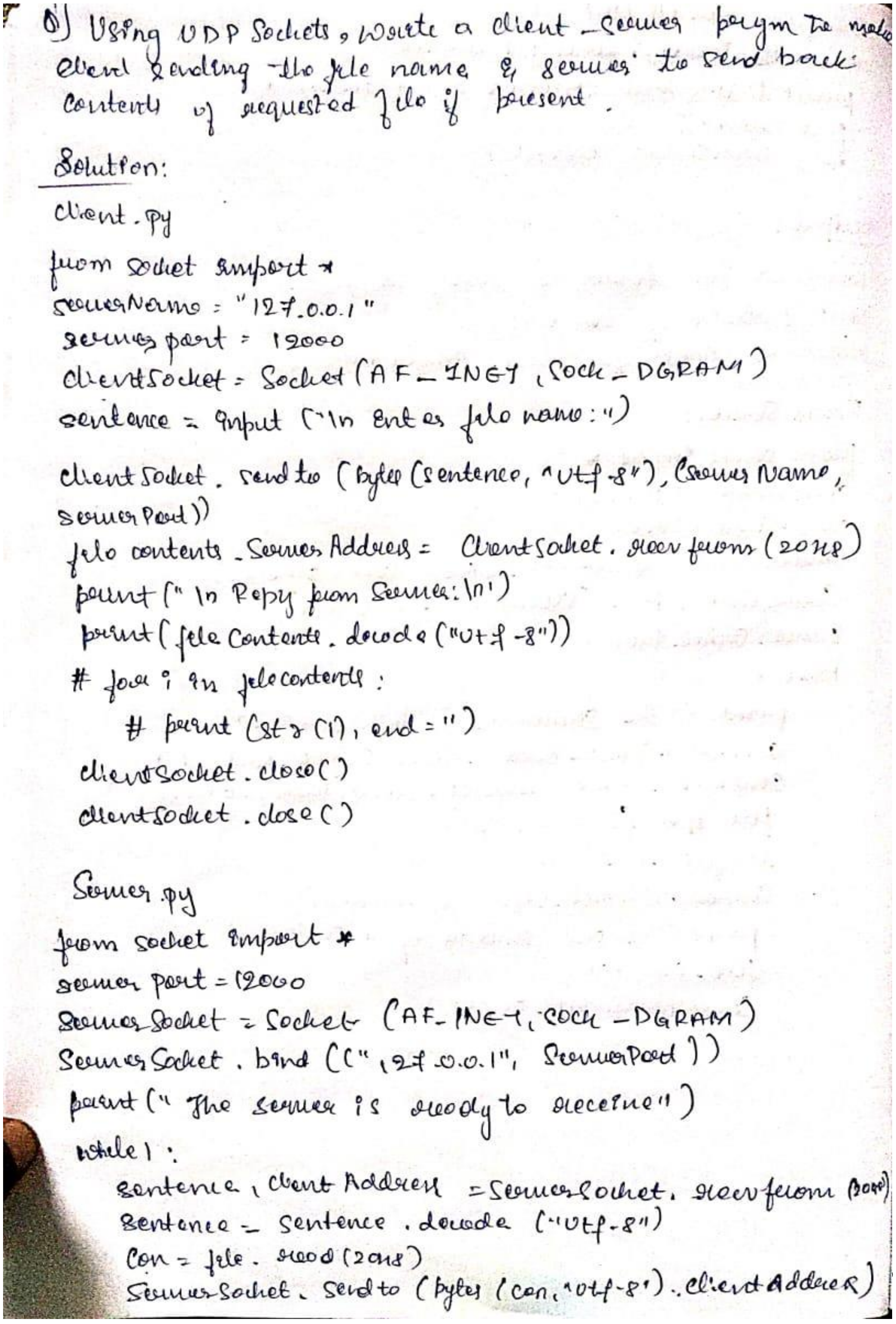


```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence=input("\n enter file name: ")
clientSocket.send(sentence.encode())
filecontents=clientSocket.recv(1024).decode()
print("\n from server: ")
print(filecontents)
clientSocket.close()
```

```
PS C:\oml project\CN> python SERVERTCP.PY
the server is ready to recieve

 sent contents of SERVERTCP.PY
the server is ready to recieve
```

```
PS C:\oml project\CN> python CLIENTTCP.PY

 enter file name: SERVERTCP.PY

 from server:
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("the server is ready to recieve")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l.encode())
    print("\n sent contents of " + sentence)
    file.close()
    connectionSocket.close()

PS C:\oml project\CN>
```

PROGRAM4: 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.
OBSERVATION:



Q) Using UDP Sockets, Write a client_server prgm to make client sending the file name & server to send back contents of requested file if present.

Solution:

Client.py

```
from socket import *
serverName = "127.0.0.1"
server port = 12000
clientSocket = Socket (AF_INET ,Sock_DGRAM)
sentence = input ("\n Enter file name: ")

clientSocket . sendto ( bytes (sentence, "utf-8"), (serverName,
serverPort))
 file contents _ ServerAddress = ClientSocket . recvfrom (2048)
 print (" \n Repy from Server: \n")
  print ( file Contents . decode ("utf-8"))
 # for i in filecontents :
      # print (str (i), end = "")
  clientSocket . close( )
  clientSocket . close ( )
```

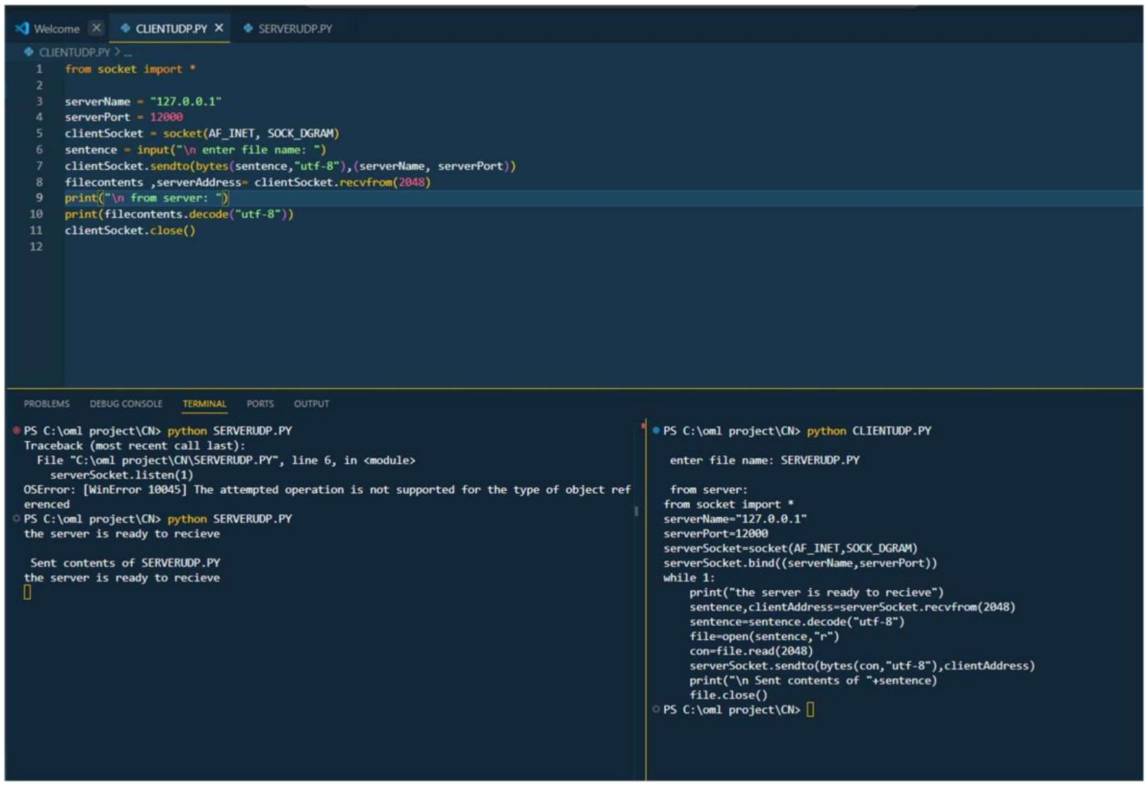
Server.py

```
from socket import *
server port = 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  = ServerSocket. recvfrom (2048)
    Sentence = sentence . decode ("utf-8")
    Con = file . read (2048)
    ServerSocket . Sendto ( bytes ( con, "utf-8") . ClientAddress)
```

```
print ('\n Sent Contents of', end = ' ')
print (sentence)
# for i in sentence:
    # print (str(i), end = '')
file.close()
```

Output:

```
Server -> The server is ready to receive.
Sent contents of server.py
The server is ready to receive

client -> client.py
Enter file name: server.py
Reply from server:
from socket import *
serverport = 12000
ServerSocket = Socket(AF_INET, SOCK_DGRAM)
ServerSocket.bind(("127.0.0.1", ServerPort))
while1:
    print ("The server is ready to receive")
    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)
    # for i in sentence:
        # print (str(i), end = '')
    file.close()
```

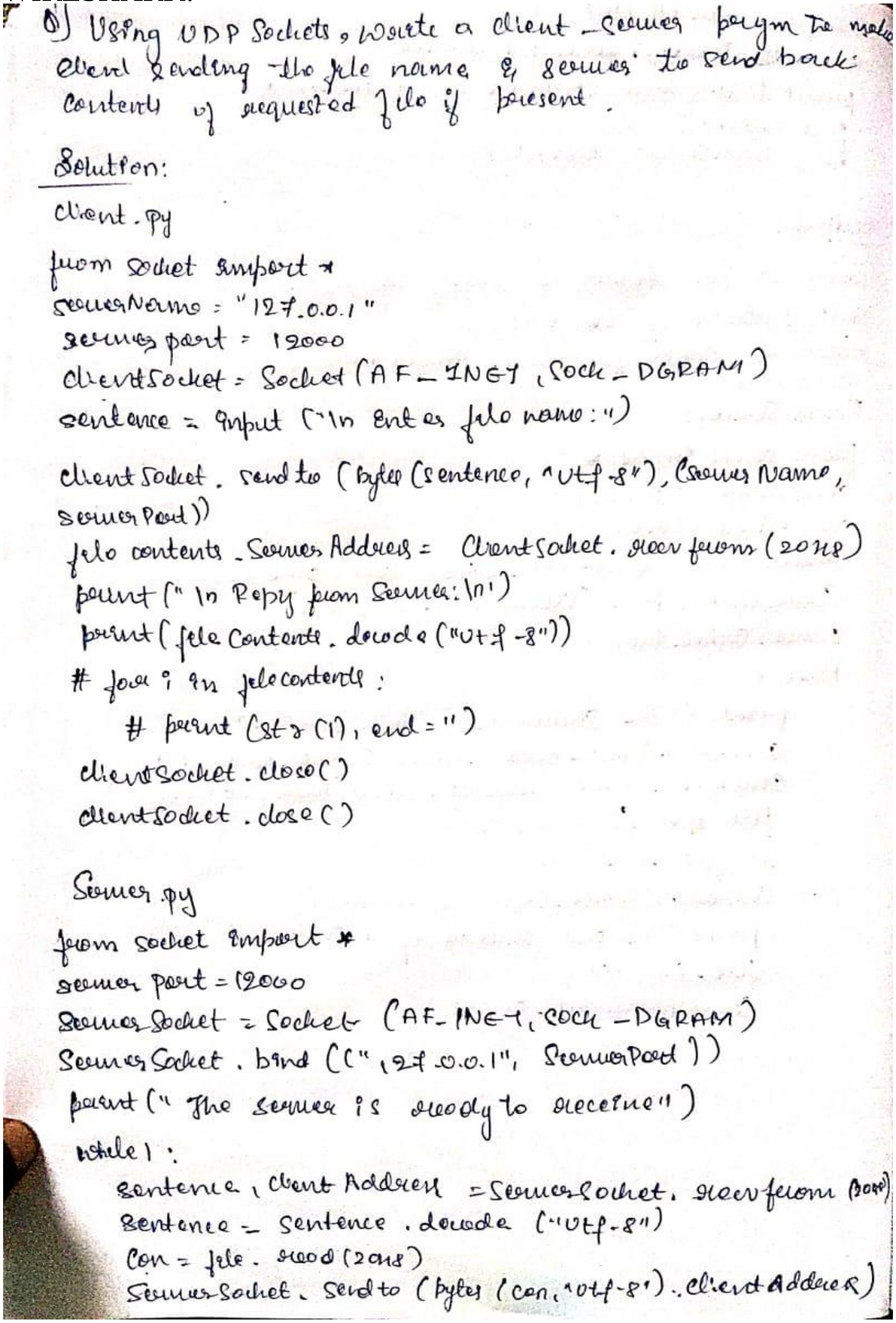


```python
from socket import *

serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\n enter file name: ")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents ,serverAddress= clientSocket.recvfrom(2048)
print("\n from server: ")
print(filecontents.decode("utf-8"))
clientSocket.close()

```

```
PS C:\oml project\CN> python SERVERUDP.PY
Traceback (most recent call last):
  File "C:\oml project\CN\SERVERUDP.PY", line 6, in <module>
    serverSocket.listen(1)
OSError: [WinError 10045] The attempted operation is not supported for the type of object ref
erenced
PS C:\oml project\CN> python SERVERUDP.PY
the server is ready to recieve

 Sent contents of SERVERUDP.PY
the server is ready to recieve
```

```
PS C:\oml project\CN> python CLIENTUDP.PY

 enter file name: SERVERUDP.PY

 from server:
from socket import *
serverName="127.0.0.1"
serverPort=12000
serverSocket=socket(AF_INET,SOCK_DGRAM)
serverSocket.bind((serverName,serverPort))
while 1:
    print("the server is ready to recieve")
    sentence,clientAddress=serverSocket.recvfrom(2048)
    sentence=sentence.decode("utf-8")
    file=open(sentence,"r")
    con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
    print("\n Sent contents of "+sentence)
    file.close()
PS C:\oml project\CN>
```

WIRESHARK:



Q) Using UDP Sockets, write a client – server prgm to make client sending the file name & server to send back contents of requested file if present.

Solution:

client.py

```
from socket import *
serverName = "127.0.0.1"
serverport = 12000
clientSocket = Socket(AF_INET, SOCK_DGRAM)
sentence = input ("\n Enter file name:")

clientSocket. sendto (bytes(sentence, "utf-8"), (serverName,
serverPort))
filecontents, ServerAddress = ClientSocket. recvfrom (2048)
print (" \n Repy from Server:\n")
print(fileContents. decode ("utf-8"))
# for i in filecontents :
      # print (str (i), end = '')
clientSocket. close()
clientSocket . close ()
```

Server.py

```
from socket import *
server port = 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 = ServerSocket. recvfrom (2048)
    sentence = sentence . decode ("utf-8")
    con = file . read (2048)
    ServerSocket . sendto (bytes (con, "utf-8"), clientAddress)
```