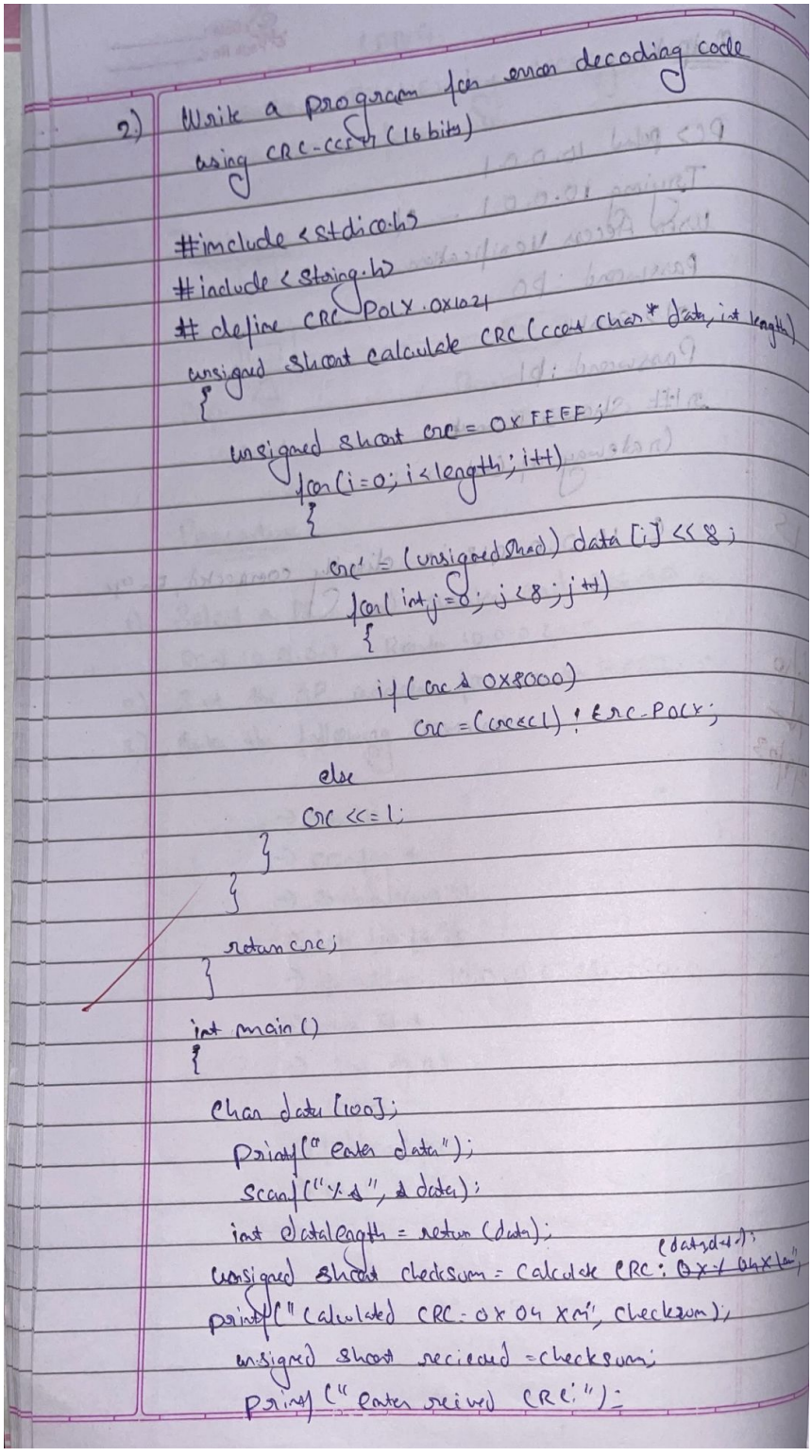
EXPERIMENT-12

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



2) Write a program for error decoding code using CRC-CCITT (16 bits)

```
#include <stdio.h>
#include <string.h>
#define CRC_POLY 0x1021
unsigned short calculate_CRC(const char *data, int length)
{
    unsigned short crc = 0xFFFF;
    for(i=0; i<length; i++)
    {
        crc = (unsigned short) data[i] << 8;
        for(int j=0; j<8; j++)
        {
            if(crc & 0x8000)
                crc = (crc<<1) ^ CRC_POLY;
            else
                crc <<= 1;
        }
    }
    return crc;
}
int main()
{
    char data[100];
    printf("Enter data");
    scanf("%s", data);
    int dataLength = strlen(data);
    unsigned short checksum = calculate_CRC(data, dataLength);
    printf("Calculated CRC = 0x%04X", checksum);
    unsigned short recieved = checksum;
    Print("Enter recived CRC:");
```

```
sscanf("%x", &recieved_checksum);
if(recieved_checksum == checksum);
    printf("Data is error-free\n"); else
    printf("Data contains error\n");
}
}
```

Output:-

Enter frame bits: 1011
Message after appending 16 zeros: 1011 0000 0000 0000 0000
Generator: 10001000000100001
Quotient: 1011
Transmitted frame: 1011 1011 0001 0110 1011
Enter route 1011 1011 0001 0110 1011
Checksum: 0000 0000 0000 0000
Data is error-free

8/10

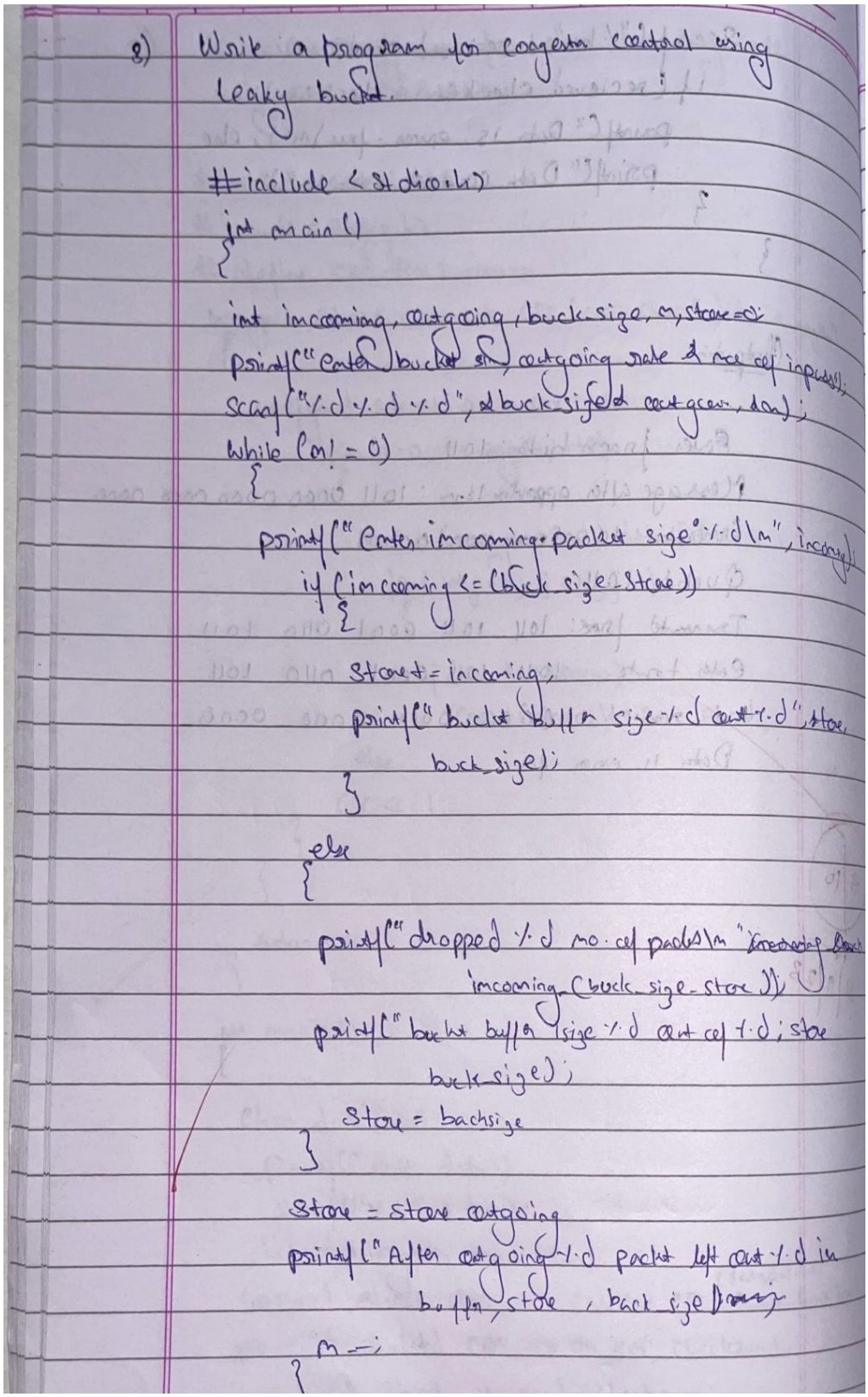
2/2/23

OUTPUT

```
Enter the frame bits:1011
Message after appending 16 zeros:1011000000000000000
generator:10001000000100001


quotient:1011
transmitted frame:10111011000101101011
Enter transmitted frame:10111011000101101011
CRC checking


last remainder:0000000000000000

Received frame is correct
Process returned 0 (0x0)   execution time : 14.468 s
Press any key to continue.
```

EXPERIMENT-14

Q) Write a program for congestion control using Leaky bucket algorithm



8) Write a program for congestion control using leaky bucket.

```
#include <stdio.h>
int main()
{
int incoming, outgoing, buck_size, n, store=0;
printf("enter bucket size, outgoing rate & no of inputs");
scanf("%d %d %d", &buck_size, &outgoing, &n);
while (n!=0)
{
printf("Enter incoming packet size %d\n", incoming);
if (incoming <= (buck_size-store))
{
store+=incoming;
printf("bucket buffer size %d out %d", store,
buck_size);
}
else
{
printf("dropped %d no of packets\n"
incoming-(buck_size-store));
printf("bucket buffer size %d out of %d; store
buck_size);
store = bachsize
}
store = store outgoing
printf("After outgoing %d packet left out %d in
buffer, store, back size
n--;
}
```

Date :
Page No :

Output1:-

Enter bucket size, outgoing rate and no of inputs
20 10 2

Enter incoming packet size = 30
incoming packet = 30
Dropped 10 no of packets
bucket buffer size 0 out of 20
After outgoing 10 packet left 20 in buffer

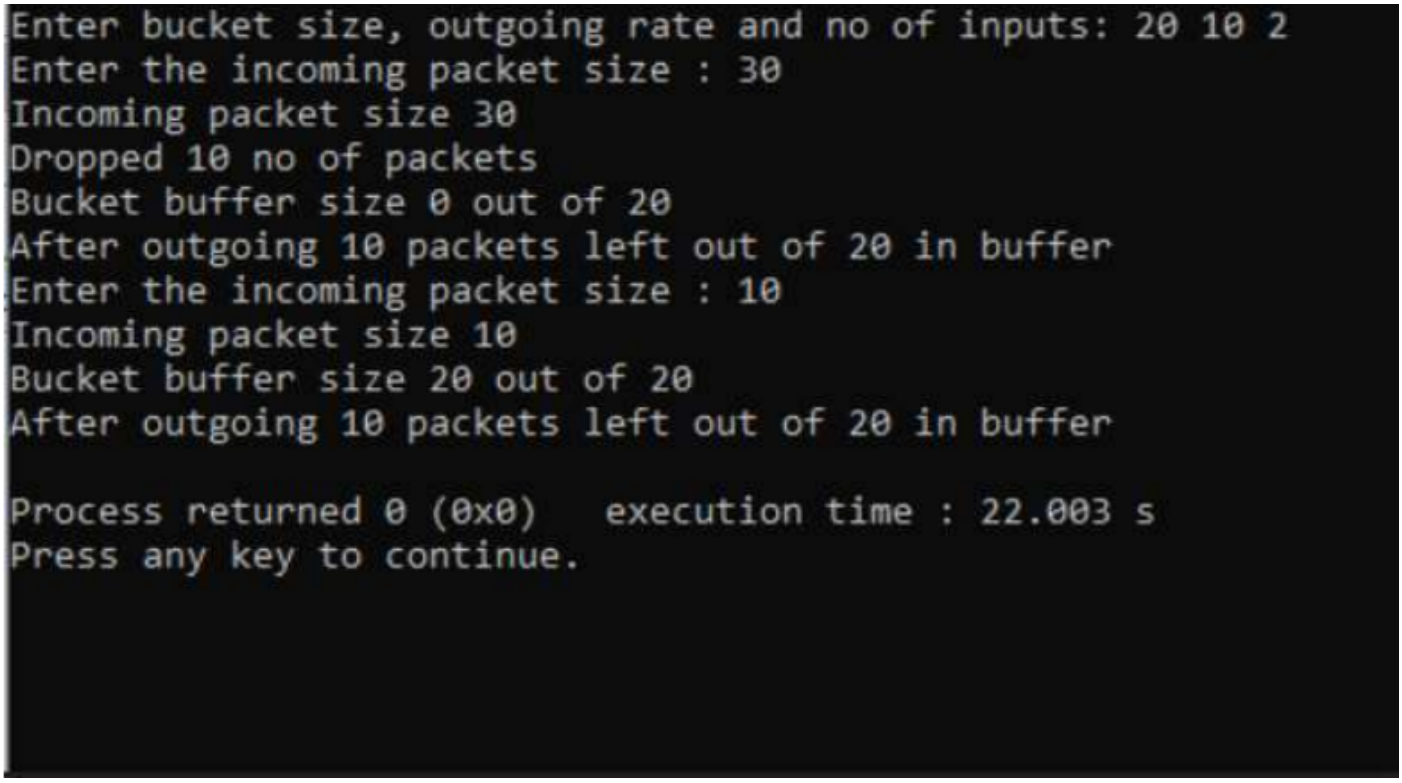
Enter incoming packet size: 10
incoming packet size 10
bucket buffer size 10 out of 20
After outgoing 10 packs left 20 in buffer

Output2: Enter bucket size, outgoing & no of i/ps.

8/10

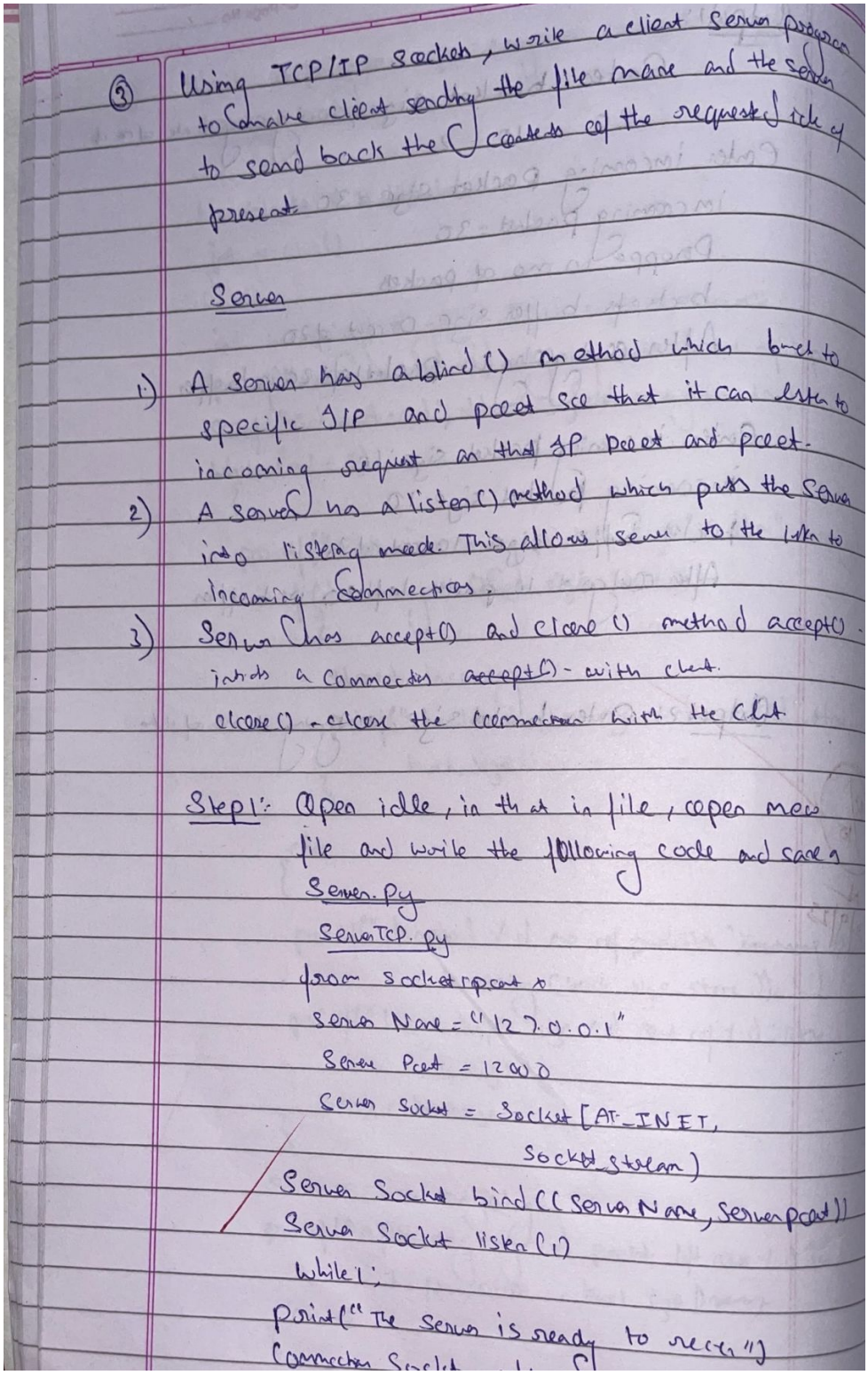
2/9/23

OUTPUT


Enter bucket size, outgoing rate and no of inputs: 20 10 2
Enter the incoming packet size : 30
Incoming packet size 30
Dropped 10 no of packets
Bucket buffer size 0 out of 20
After outgoing 10 packets left out of 20 in buffer
Enter the incoming packet size : 10
Incoming packet size 10
Bucket buffer size 20 out of 20
After outgoing 10 packets left out of 20 in buffer
Process returned 0 (0x0) execution time : 22.003 s
Press any key to continue.

EXPERIMENT-15

Q) 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.



③ 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

Server

1) A server has a bind() method which binds to specific IP and port so that it can listen to incoming request on that IP port and port.
2) A server has a listen() method which puts the server into listening mode. This allows server to the listen to incoming connections.
3) Server has accept() and close() method accept() - accepts a connection ~~accept()~~ - with client.
close() - close the connection with the client

Step1: Open idle, in that in file, open new file and write the following code and save as

Server.py

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 recv")
connectionSocket, addr =
```

file = open (sentence, "r")
l = file.read (1024)
connectionSocket.send (l.encode())
print("\n Sent contents of " + sentence)
file.close()
connectionSocket.close()

Step2: Run the file server.py

O/P ⇒ The server is ready to second. This shows that server is working

Client:

Step1: Make a Socket object

Step2: Establish a connection with server and lastly we will read data from the server and close the connection.

Step3: Open idle and open new file and write the following code and save as "client.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: \n")
```

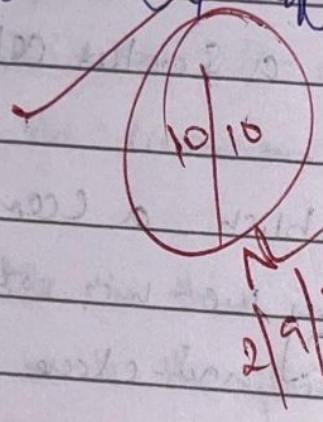
Print (file contents)
Client Socket . Close ()

Run the file client.py

Client O/P => enter file name. server.py

Server O/P => The server is ready to recieve sent
the contents of sam.py.
The server is ready to receive.

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 requested file if present.



CODE-

ClientTCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name:")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ("\nFrom Server:\n")
print(filecontents)
clientSocket.close()
```

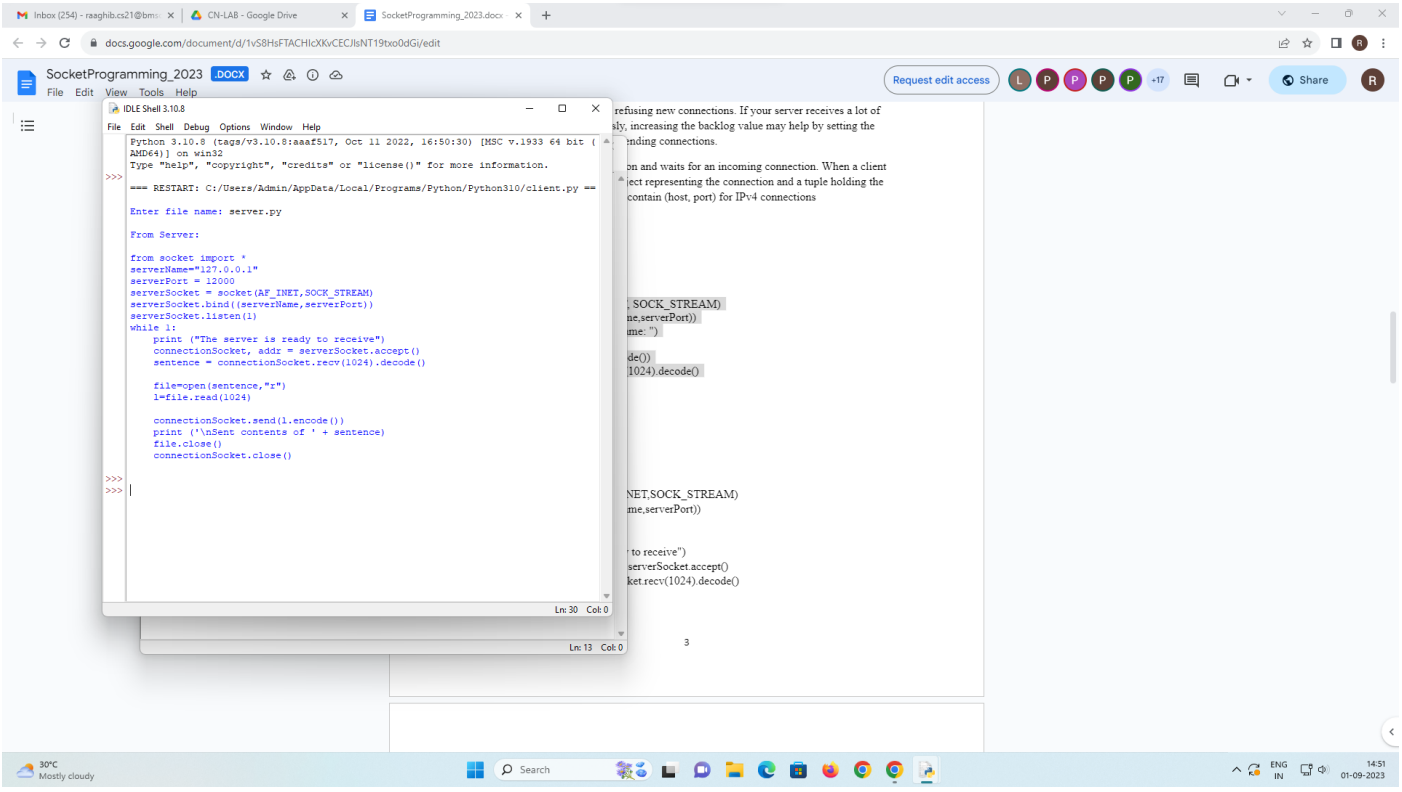
ServerTCP.py

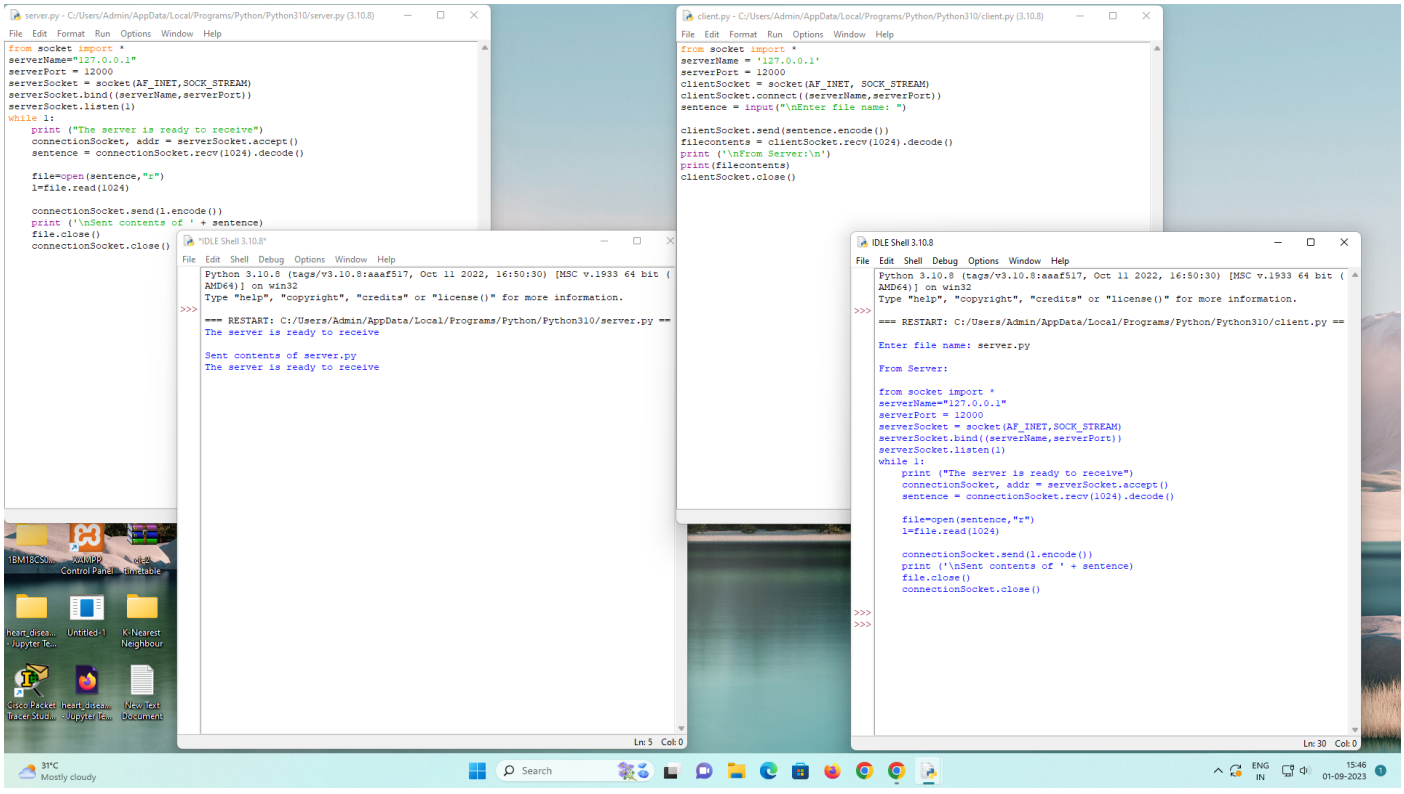
```
from socket import *
serverName='127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
  print("The server is ready to receive")
  connectionSocket, addr = serverSocket.accept()
  sentence = connectionSocket.recv(1024).decode()

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

  connectionSocket.send(l.encode())
  print ("\nSent contents of " + sentence)
```

file.close()

connectionSocket.close()

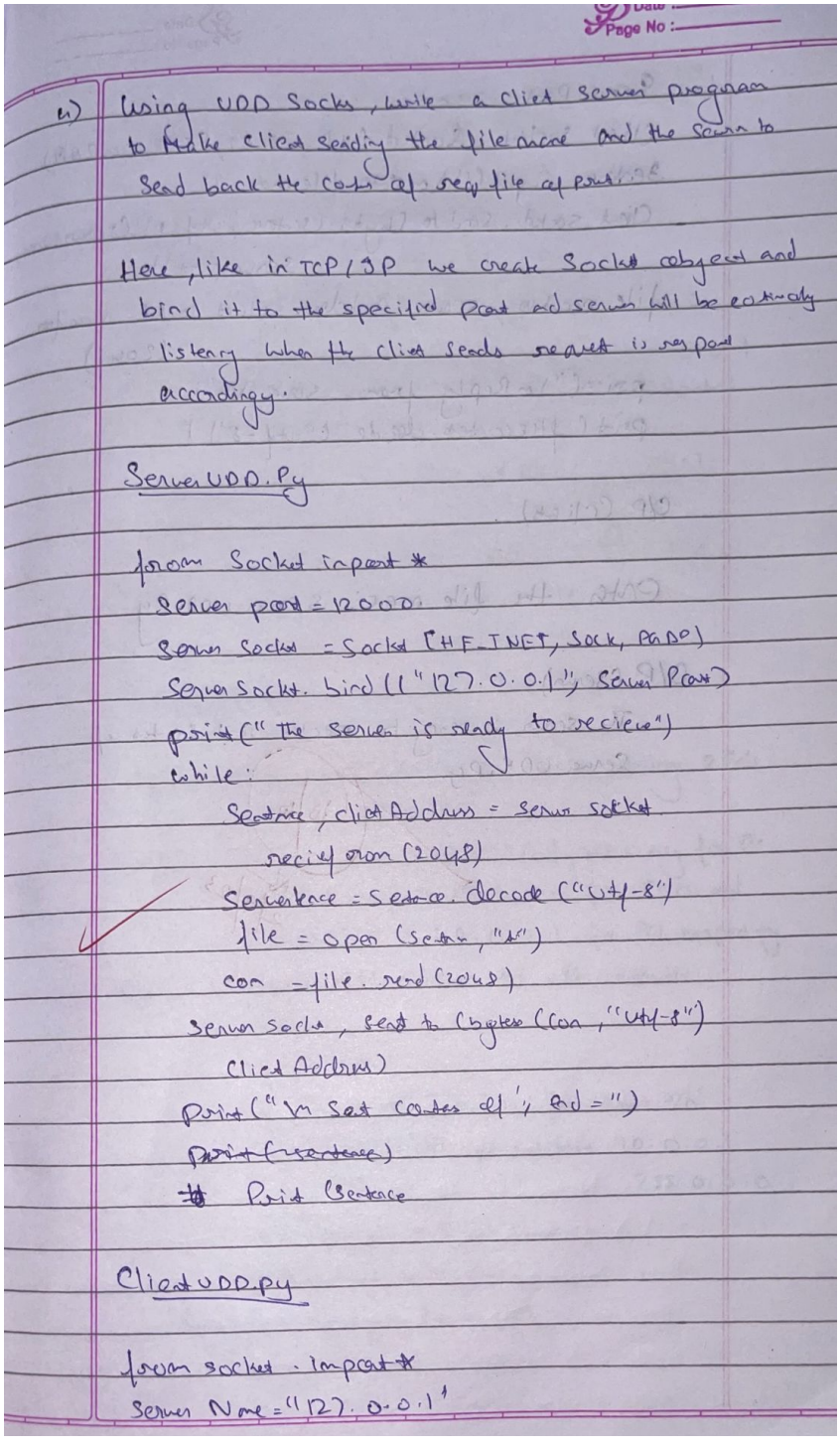
OUTPUT





EXPERIMENT-16

Q) 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



a) Using UDP Socks, write a client server program to make client sending the file name and the server to send back the contents of requested file at port.

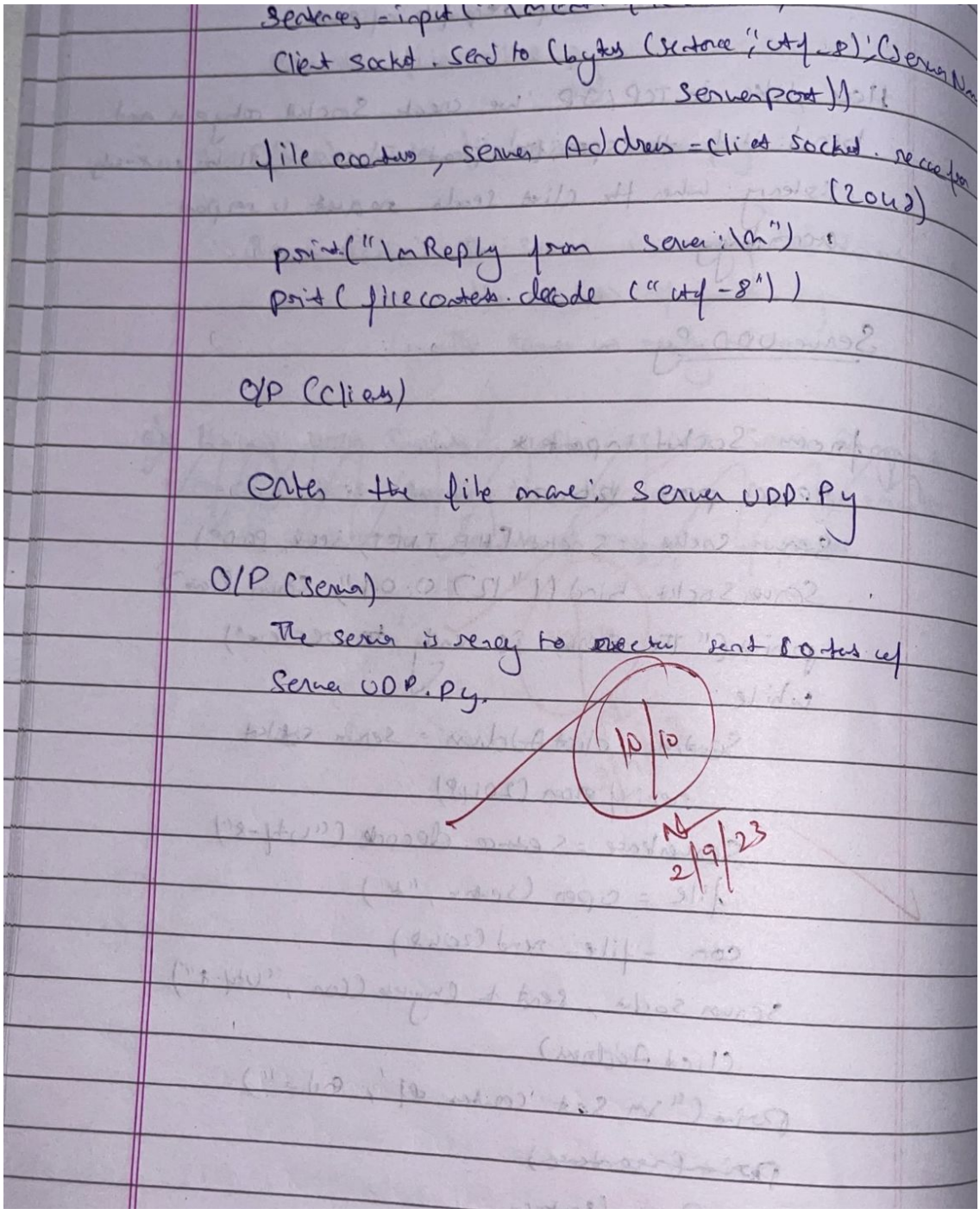
Here, like in TCP/IP we create Socket object and bind it to the specified port and server will be continuously listening when the client sends request is response accordingly.

Server UDP.Py

```
from Socket import *
Server port = 12000
Server Socket = Socket (AF_INET, SOCK_DGRAM)
Server Socket. bind (("127.0.0.1", Server Port))
print ("The server is ready to recieve")
while:
    Sentence, client Address = server socket
      recieve from (2048)
    Sentence = Sentence. decode ("utf-8")
    file = open (sentence, "r")
    con = file. read (2048)
    Server Socket, sent to (bytes (con, "utf-8")
      Client Address)
    Print ("\n Sent contents of ', end = '')
    Print (sentence)
    Print Sentence
```

Client UDP.py

```
from socket . import *
Server Name = "127. 0.0.1"
```



CODE-

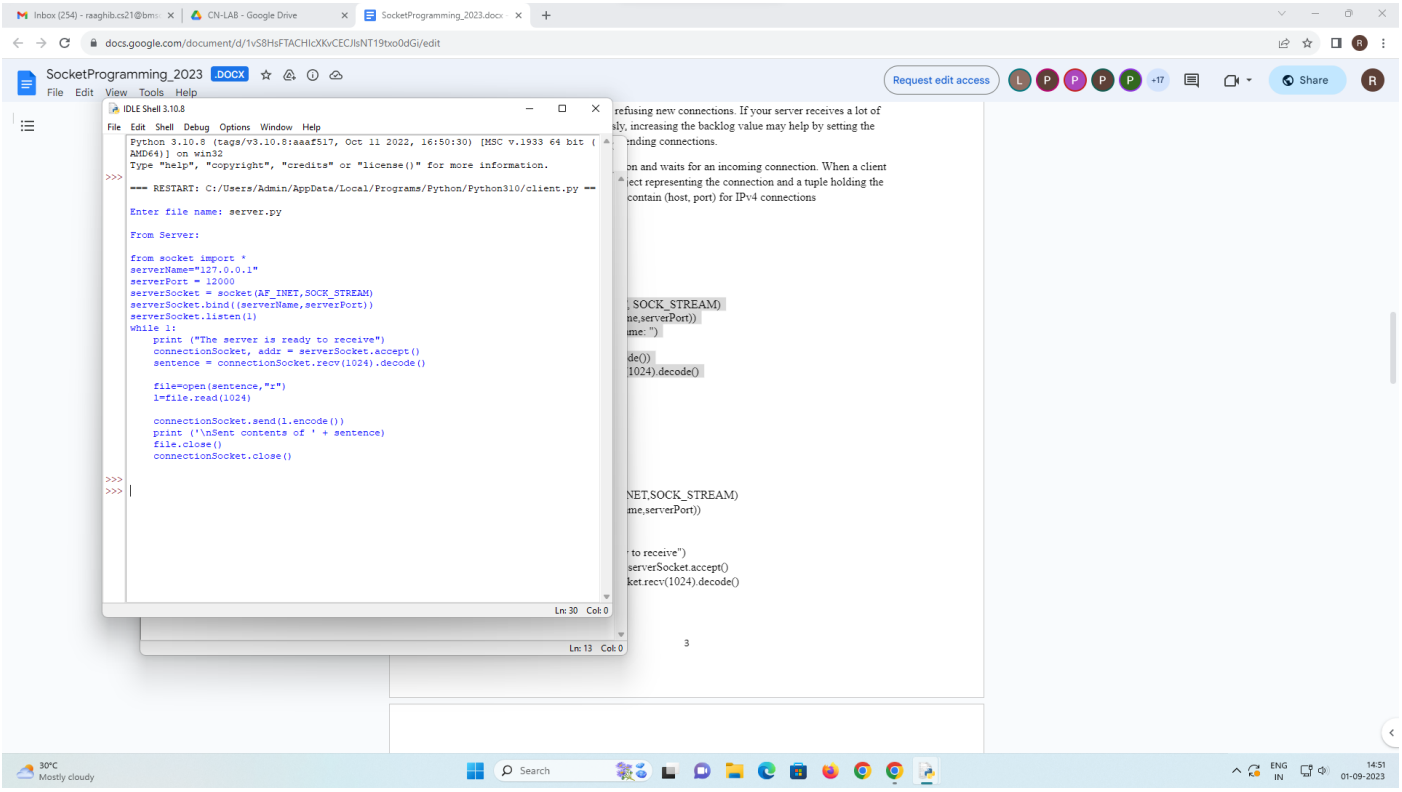
ClientUDP.py

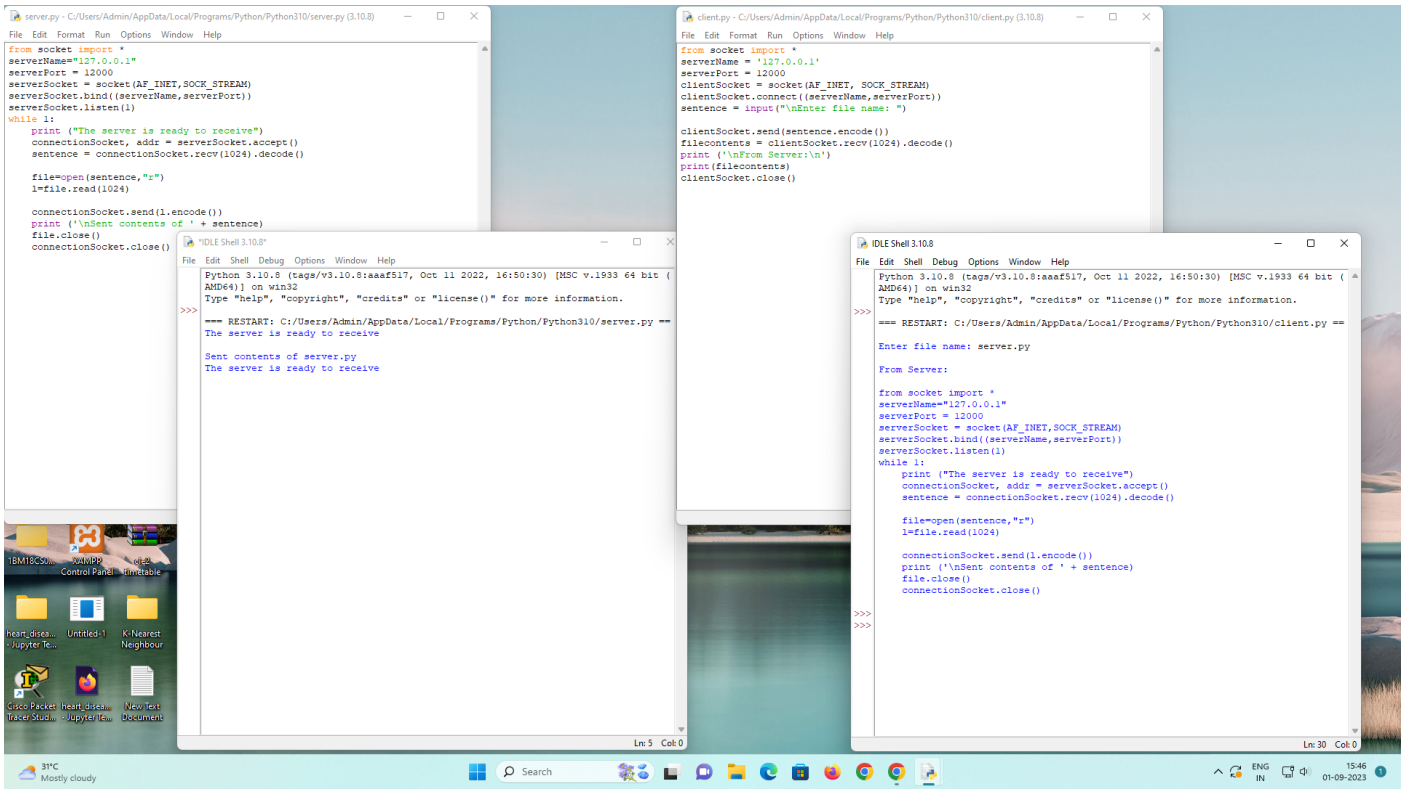
```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\nEnter file name:")


clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ("\nReply from Server:\n")
```

```
print (filecontents.decode("utf-8"))
#for i in filecontents:
  #print(str(i), end ='')
clientSocket.close()
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 1:
    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 ("nSent contents of ", end = "")
    print (sentence)
    # for i in sentence:
      # print (str(i), end = '')
    file.close()
```


SocketProgramming_2023
IDLE Shell 3.10.8
Enter file name: server.py
From Server:
The server is ready to receive


server.py - C:/Users/Admin/AppData/Local/Programs/Python/Python310/server.py (3.10.8)
client.py - C:/Users/Admin/AppData/Local/Programs/Python/Python310/client.py (3.10.8)
IDLE Shell 3.10.8
The server is ready to receive
Sent contents of server.py
The server is ready to receive
Enter file name: server.py
From Server: