PROGRAM 10 : UDP SOCKET , SERVER / CLIENT

CODE PART 1 – UDP SERVER.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 recieve ")
while 1:
    sentence, addr = serverSocket.recvfrom(2048)
    file = open(sentence, "r")
    l = file.read (2048)
    serverSocket.sendto(bytes(l, "utf-8"), add
    print ("sent back to client", l)
file.close()
```

CODE PART 2 – UDP ~~[illegible]~~ CLIENT.PY

```
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, addr = clientSocket.recvfrom(2048)
print ('From server :', filecontents)
clientSocket.close()
```

```
buffer = Buffer(int(input("enter [illegible]")))
client = Client(int(input("enter the maximum
                    rate of [illegible]")))
data_to_send = str

while truestate :
    data_to_send = input("enter a string")
    count = 0
    if buffer.checkstate():
        for i in range(0, len(data_to_send)):
~~else~~    if i < client.rate :
                Client.data.append(data_to_send)
            else:
                if count < buffer.buffer_size:
                    buffer.buffer.append~~(data_to_send~~
                                  (data_to_send)
                    count = len(buffer.buffer)
                else:
                    print("Data loss" + data_to_send)
    else :
        j = 0
        for i in range(0, len(data_to_send) +
                          len(buffer.buffer)):
            if i < client.rate :
                if len(buffer.buffer):
                    client.data.append(buffer.buffer[0])
                    del buffer.buffer[0]
                else :
                    client.data.append(data_to_send[j])
                    j += 1
            else :
                ~~[illegible]~~
            min_index = V
    return min_index

def digikstra (self, src) :
    dist = [sys.maxsize] * self.V
    dist [src] = 0
    sptSet = [False] * self.V
    for cout in range (self.V) :
        u = self. minDistance (dist, sptSet)
        sptSet [u] = True
        for v in range (self. V) :
            if self. graph [u][v] > 0 and
              sptSet [v] == False and
              dist [v] > dist [u] + self.graph[u][v]:
                  dist [v] = dist [u] + self.graph[u][
    self.printSolution [illegible](dist)
```