TASK 1.

UDPPingerServer:

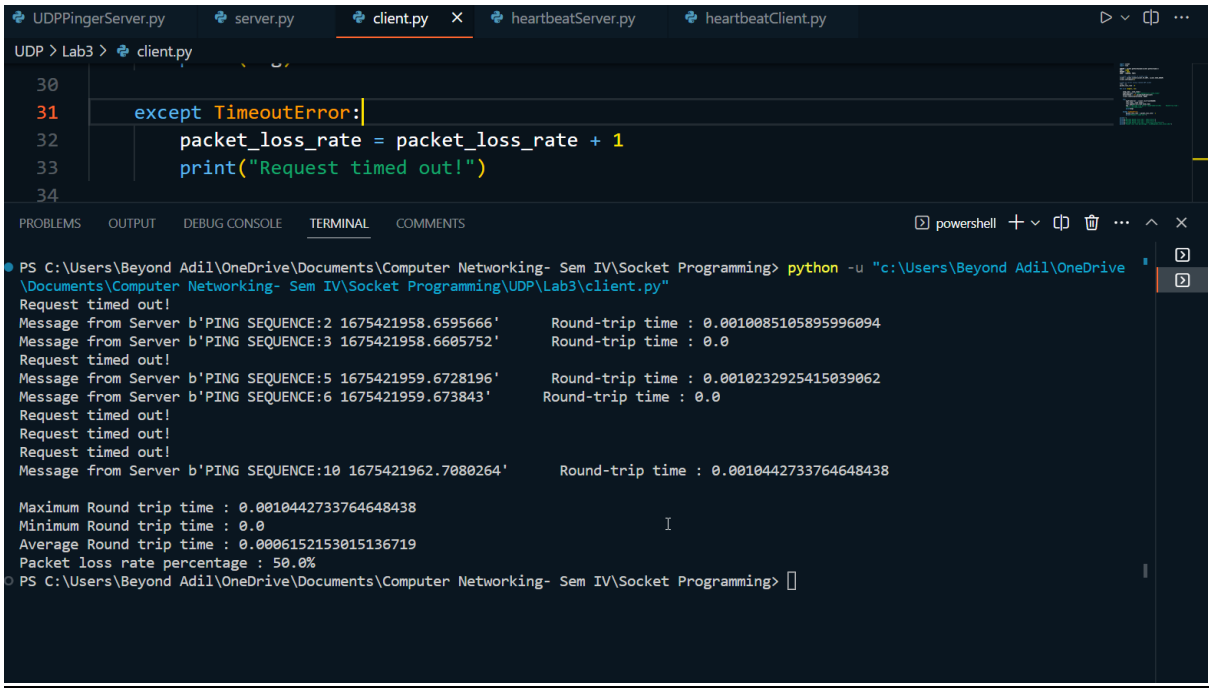
```
# UDPPingerServer.py
# We will need the following module to generate randomized
lost packets
import random
import socket

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
SERVER = socket.gethostbyname(socket.gethostname())
PORT = 12000
HEADER = 1024
ADDR = (SERVER, PORT)
serverSocket = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)

# Assign IP address and port number to socket

serverSocket.bind(ADDR)
print("UDP server up and listening")


while True:
    # Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)

    # Receive the client packet along with the address it is
coming from
    message, address = serverSocket.recvfrom(HEADER)
    print(f"Message from Client :
{message}        rand_value : {rand}")

    # Capitalize the message from the client
    message = message.upper()

    # If rand is less is than 4, we consider the packet lost
and do not respond
```

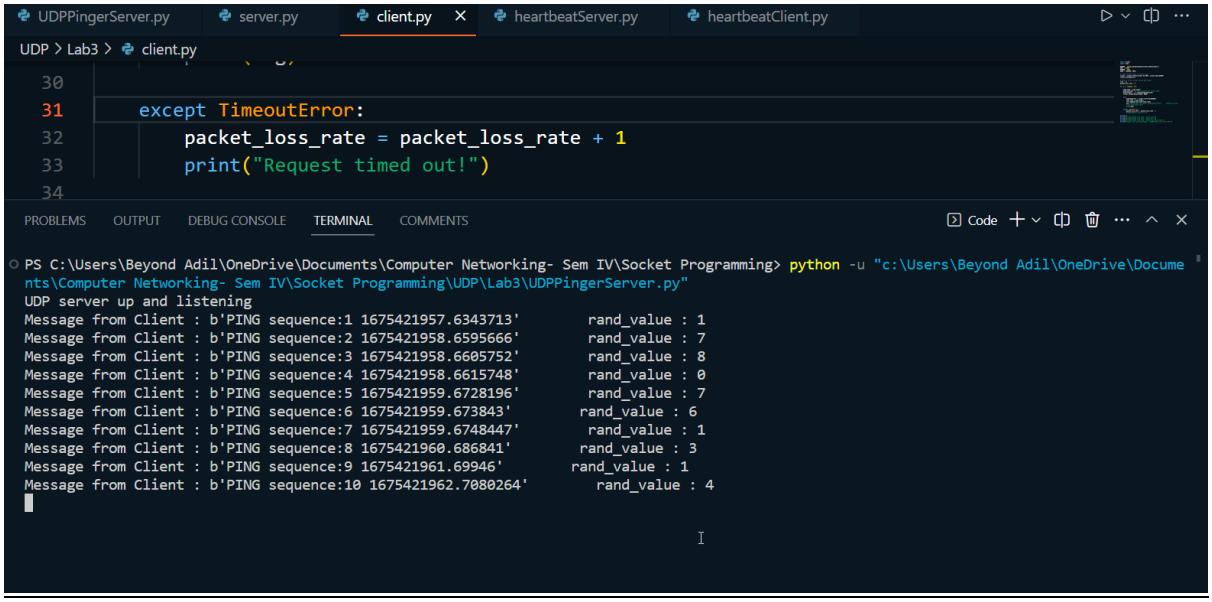
```
    if rand < 4:
        continue

    # Otherwise, the server responds
    serverSocket.sendto(message, address)
```

UDPclient:

```
import socket
import time

SERVER = socket.gethostbyname(socket.gethostname())
PORT = 12000
HEADER = 1024
ADDR = (SERVER, PORT)

# Create a UDP socket at client side
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.settimeout(1)

# Send to server using created UDP socket
rtts = []
packet_loss_rate = 0

for i in range(1, 11):

    send_time = time.time()
    msgFromClient = f"PING sequence:{i} {send_time}"
    bytesToSend = str.encode(msgFromClient)
    client.sendto(bytesToSend, ADDR)

    try:
        msgFromServer = client.recvfrom(HEADER)
        recv_time = time.time()
        rtts.append(recv_time-send_time)
```

```
        msg = f"Message from Server
{msgFromServer[0]}      Round-trip time : {recv_time-
send_time}"
        print(msg)

    except TimeoutError:
        packet_loss_rate = packet_loss_rate + 1
        print("Request timed out!")

print("")
print(f"Maximum Round trip time : {max(rtts)}")
print(f"Minimum Round trip time : {min(rtts)}")
print(f"Average Round trip time : {sum(rtts)/len(rtts)}")
print(f"Packet loss rate percentage :
{(100*packet_loss_rate)/10}%")
```





TASK 2:

Server:

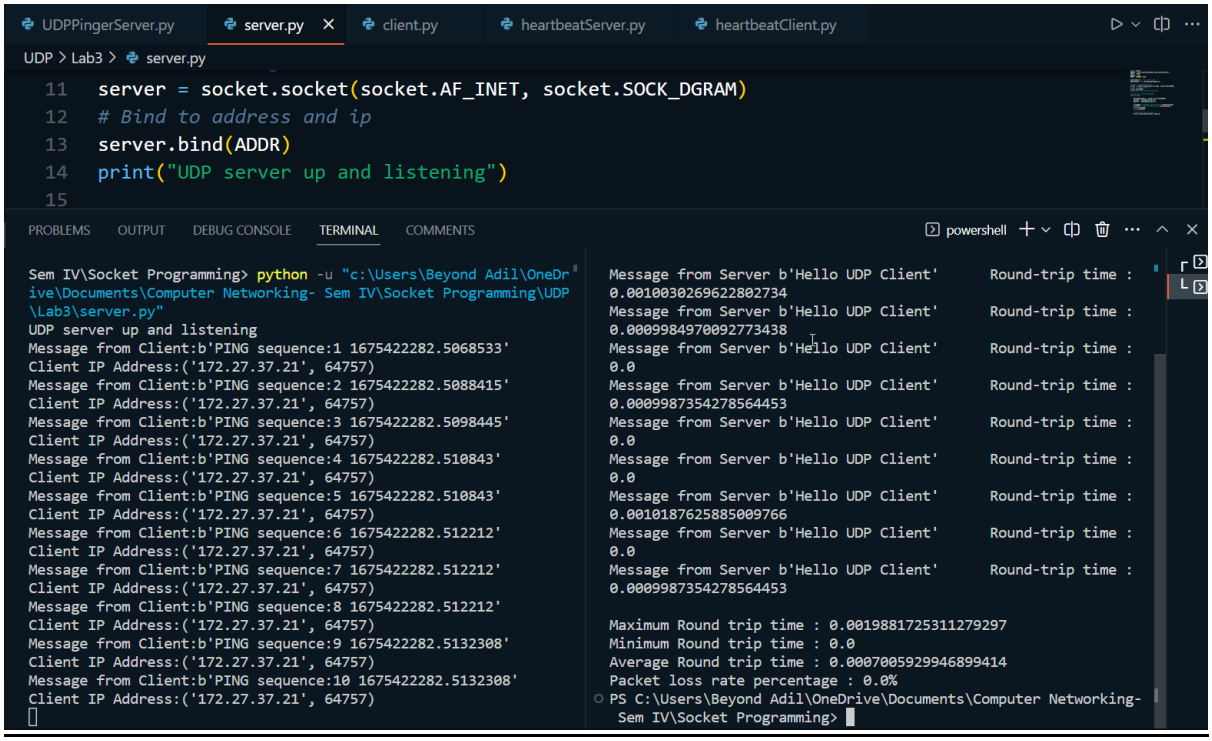
```
import socket
SERVER = socket.gethostbyname(socket.gethostname())
PORT = 12000
HEADER = 1024
ADDR = (SERVER, PORT)

msgFromServer = "Hello UDP Client"
bytesToSend = str.encode(msgFromServer)

# Create a datagram socket
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Bind to address and ip
server.bind(ADDR)
print("UDP server up and listening")

# Listen for incoming datagrams
while (True):

    bytesAddressPair = server.recvfrom(HEADER)
    message = bytesAddressPair[0]
```

```
    address = bytesAddressPair[1]

    clientMsg = "Message from Client:{}".format(message)
    clientIP = "Client IP Address:{}".format(address)
    print(clientMsg)
    print(clientIP)

    # Sending a reply to client
    server.sendto(bytesToSend, address)
```

Client:

```
import socket
import time

SERVER = socket.gethostbyname(socket.gethostname())
PORT = 12000
HEADER = 1024
ADDR = (SERVER, PORT)

# Create a UDP socket at client side
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.settimeout(1)

# Send to server using created UDP socket
rtts = []
packet_loss_rate = 0

for i in range(1, 11):

    send_time = time.time()
    msgFromClient = f"PING sequence:{i} {send_time}"
    bytesToSend = str.encode(msgFromClient)
    client.sendto(bytesToSend, ADDR)

    try:
```

```
        msgFromServer = client.recvfrom(HEADER)
        recv_time = time.time()
        rtts.append(recv_time-send_time)
        msg = f"Message from Server
{msgFromServer[0]}      Round-trip time : {recv_time-
send_time}"
        print(msg)

    except TimeoutError:
        packet_loss_rate = packet_loss_rate + 1
        print("Request timed out!")

print("")
print(f"Maximum Round trip time : {max(rtts)}")
print(f"Minimum Round trip time : {min(rtts)}")
print(f"Average Round trip time : {sum(rtts)/len(rtts)}")
print(f"Packet loss rate percentage :
{(100*packet_loss_rate)/10}%")
```



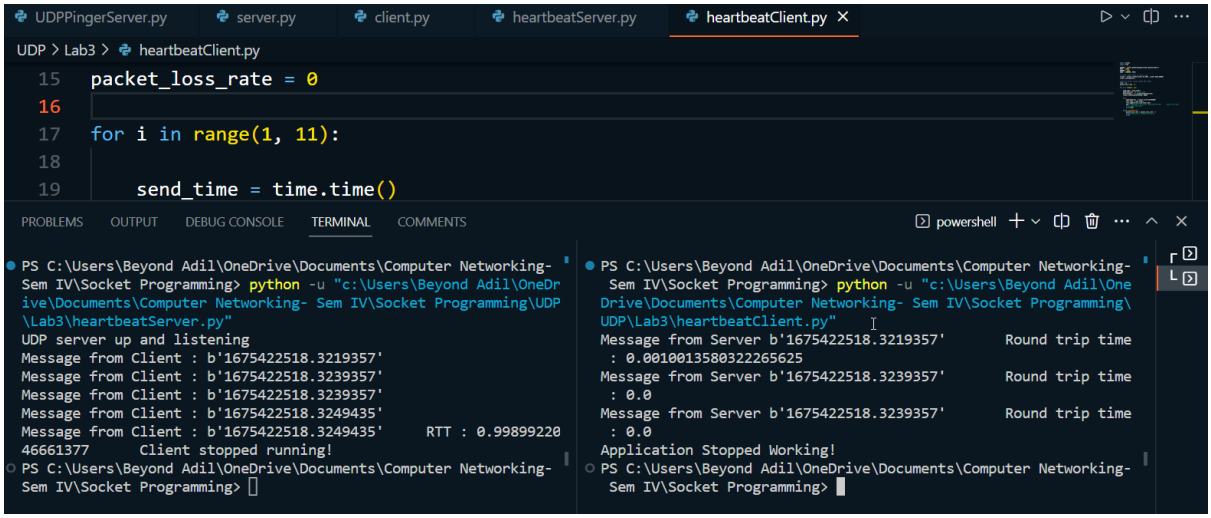
TASK 3:

heartbeatServer:

```
import random
import time
import socket

# Create a UDP socket
SERVER = socket.gethostbyname(socket.gethostname())
PORT = 12000
HEADER = 1024
ADDR = (SERVER, PORT)
serverSocket = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
serverSocket.bind(ADDR)
print("UDP server up and listening")


while True:
    rand = random.randint(0, 2)
    recv_time = time.time()
    message, address = serverSocket.recvfrom(HEADER)
    print(f"Message from Client : {message}")

    send_time = float(message)
    time_diff = (recv_time - send_time) + rand

    if time_diff > 1:
        print(f"Message from Client : {message}     RTT :
{time_diff-1}      Client stopped running!")
        break

    serverSocket.sendto(message, address)
```

heartbeatClient:

```
import socket
import time

SERVER = socket.gethostbyname(socket.gethostname())
PORT = 12000
HEADER = 1024
ADDR = (SERVER, PORT)

# Create a UDP socket at client side
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.settimeout(1)

# Send to server using created UDP socket
rtts = []
packet_loss_rate = 0

for i in range(1, 11):

    send_time = time.time()
    msgFromClient = f"{send_time}"
    bytesToSend = str.encode(msgFromClient)
    client.sendto(bytesToSend, ADDR)

    try:
        msgFromServer = client.recvfrom(HEADER)
        recv_time = time.time()
        rtts.append(recv_time-send_time)
        msg = f"Message from Server
{msgFromServer[0]}       Round trip time : {recv_time-
send_time}"
        print(msg)

    except TimeoutError:
        packet_loss_rate = packet_loss_rate + 1
        print("Application Stopped Working!")
        break
```


UDPPingerServer.py
server.py
client.py
heartbeatServer.py
heartbeatClient.py
UDP > Lab3 > heartbeatClient.py
15 packet_loss_rate = 0
16
17 for i in range(1, 11):
18
19 send_time = time.time()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
COMMENTS
powershell
PS C:\Users\Beyond Adil\OneDrive\Documents\Computer Networking- Sem IV\Socket Programming> python -u "c:\Users\Beyond Adil\OneDrive\Documents\Computer Networking- Sem IV\Socket Programming\UDP\Lab3\heartbeatServer.py"
UDP server up and listening
Message from Client : b'1675422518.3219357'
Message from Client : b'1675422518.3239357'
Message from Client : b'1675422518.3239357'
Message from Client : b'1675422518.3249435'
Message from Client : b'1675422518.3249435' RTT : 0.9989922046661377 Client stopped running!
PS C:\Users\Beyond Adil\OneDrive\Documents\Computer Networking- Sem IV\Socket Programming>
PS C:\Users\Beyond Adil\OneDrive\Documents\Computer Networking- Sem IV\Socket Programming> python -u "c:\Users\Beyond Adil\OneDrive\Documents\Computer Networking- Sem IV\Socket Programming\UDP\Lab3\heartbeatClient.py"
Message from Server b'1675422518.3219357' Round trip time : 0.0010013580322265625
Message from Server b'1675422518.3239357' Round trip time : 0.0
Message from Server b'1675422518.3239357' Round trip time : 0.0
Application Stopped Working!
PS C:\Users\Beyond Adil\OneDrive\Documents\Computer Networking- Sem IV\Socket Programming>