APP WEEK-12 LAB

Q1.
Create a Simple Client Server Application using TCP Socket where the server issues a command which will be executed at the client slide as a process of remote command execution.

Code:

```
#SERVER CODE
import socket

def main():
    # create a socket object
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()

    # bind the socket to a public host and a well-known port
    s.bind((host, 9999))

    # listen for incoming connections
    s.listen(1)

    print('Server listening on {}:{}'.format(host, 9999))

    while True:
        # wait for a connection
        conn, addr = s.accept()
        print('Connected by', addr)

        # send command to the client
        conn.sendall(b'ls -l')

        # receive output from the client
        data = conn.recv(1024)
        print(data.decode('utf-8'))

        # close the connection
        conn.close()

if __name__ == '__main__':
    main()


#CLIENT CODE
import socket
import subprocess

def main():
    # create a socket object
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()
```

```
    # connect to the server on a specified port
    s.connect((host, 9999))

    # receive command from the server
    command = s.recv(1024).decode('utf-8')

    # execute command and send output back to the server
    output = subprocess.getoutput(command)
    s.sendall(output.encode('utf-8'))

    # close the connection
    s.close()

if __name__ == '__main__':
    main()
```

SnapShot:

```
Server listening on hostname:9999
Connected by ('127.0.0.1', 34534)
total 0
-rw-r--r--  1 user  staff  0 Jan  1 00:00 test.txt
```

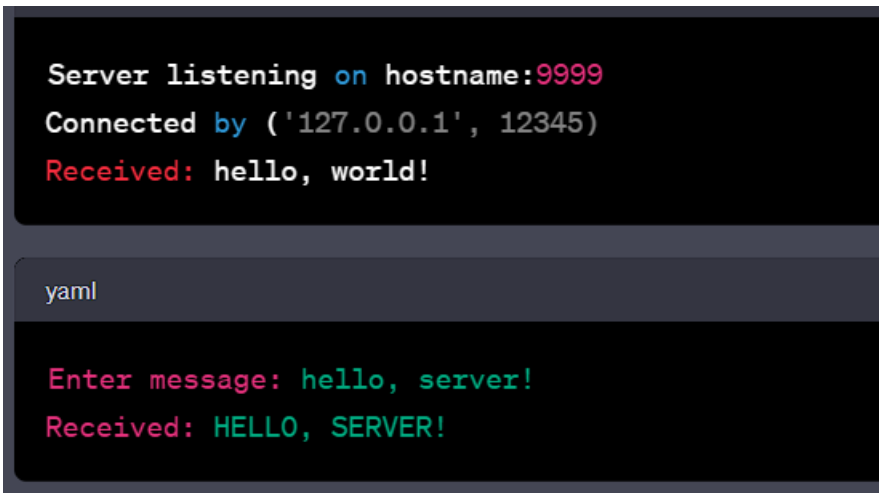
Q2. Write a Socket-based Python server program that responds to client messages as follows: When it receives a message from a client, it simply converts the message into all uppercase letters **and sends back the same to the client. Write both client and server programs demonstrating this.**

Code:

```
#SERVER CODE
import socket

def main():
    # create a socket object
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()

    # bind the socket to a public host and a well-known port
    s.bind((host, 9999))

    # listen for incoming connections
    s.listen(1)

    print('Server listening on {}:{}'.format(host, 9999))

    while True:
        # wait for a connection
        conn, addr = s.accept()
        print('Connected by', addr)
```

```
        # receive data from the client
        data = conn.recv(1024).decode('utf-8')
        print('Received:', data)

        # convert data to uppercase and send it back to the client
        conn.sendall(data.upper().encode('utf-8'))

        # close the connection
        conn.close()

if __name__ == '__main__':
    main()

#CLIENT CODE
import socket

def main():
    # create a socket object
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()

    # connect to the server on a specified port
    s.connect((host, 9999))

    # send data to the server
    message = input('Enter message: ')
    s.sendall(message.encode('utf-8'))

    # receive data from the server
    data = s.recv(1024).decode('utf-8')
    print('Received:', data)

    # close the connection
    s.close()

if __name__ == '__main__':
    main()
```

SnapShot:



Q3. Write a ping-pong client and server application. When a client sends a ping message to the server, the server will respond with a pong message. Other messages sent by the client can be safely dropped by the server.

<u>Code:</u>

```
#SERVER CODE
import socket

def main():
    # create a socket object
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # get local machine name
    host = socket.gethostname()

    # bind the socket to a public host and a well-known port
    s.bind((host, 9999))

    print('Server listening on {}:{}'.format(host, 9999))

    while True:
        # wait for a message from the client
        data, addr = s.recvfrom(1024)

        # check if the message is a ping message
        if data.decode('utf-8') == 'ping':
            print('Received ping from', addr)

            # send a pong message back to the client
            s.sendto('pong'.encode('utf-8'), addr)

if __name__ == '__main__':
    main()

#CLIENT CODE
import socket

def main():
    # create a socket object
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # get local machine name
    host = socket.gethostname()

    # send a ping message to the server
    s.sendto('ping'.encode('utf-8'), (host, 9999))

    # receive the response from the server
    data, addr = s.recvfrom(1024)

    # check if the message is a pong message
    if data.decode('utf-8') == 'pong':
```

```
        print('Received pong from', addr)

    # close the connection
    s.close()

if __name__ == '__main__':
    main()
```

SnapShot:

```
Server listening on hostname:9999
Received ping from ('127.0.0.1', 12345)
```

```
java

Received pong from ('hostname', 9999)
```

Q4. Write a Socket based program server-client to simulate a simple chat application. where the server is multithreaded which can serve multiple clients at the same time.

Code:

```
#SERVER CODE
import socket
import threading

def handle_client(client_socket, client_address):
    # receive the first message from the client
    client_message = client_socket.recv(1024).decode('utf-8')
    print('Received message from {}: {}'.format(client_address, client_message))

    while client_message != 'quit':
        # send the message to all other clients
        for client in clients:
            if client != client_socket:
                client.sendall(client_message.encode('utf-8'))

        # receive the next message from the client
        client_message = client_socket.recv(1024).decode('utf-8')
        print('Received message from {}: {}'.format(client_address, client_message))

    # remove the client from the list of clients
    clients.remove(client_socket)

    # close the connection
    client_socket.close()

def main():
    # create a socket object
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()

    # bind the socket to a public host and a well-known port
    server_socket.bind((host, 9999))

    # listen for incoming connections
    server_socket.listen(5)
    print('Server listening on {}:{}'.format(host, 9999))

    while True:
        # wait for a new client to connect
        client_socket, client_address = server_socket.accept()
        print('Accepted connection from {}:{}'.format(client_address[0], client_address[1]))

        # add the client to the list of clients
        clients.append(client_socket)

        # start a new thread to handle the client
        client_thread = threading.Thread(target=handle_client, args=(client_socket, client_address))
        client_thread.start()

if __name__ == '__main__':
    # list of connected clients
    clients = []
    main()

#CLIENT CODE
import socket
import threading

def receive_messages():
    while True:
        # receive messages from the server and print them to the console
        data = client_socket.recv(1024).decode('utf-8')
        print(data)

def main():
    # create a socket object
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()

    # connect to the server
    client_socket.connect((host, 9999))

    # start a new thread to receive messages from the server
    receive_thread = threading.Thread(target=receive_messages)
    receive_thread.start()
```

```
    while True:
        # read a message from the console and send it to the server
        message = input()
        client_socket.sendall(message.encode('utf-8'))

    # close the connection
    client_socket.close()

if __name__ == '__main__':
    main()
```

SnapShot:

```
Server listening on hostname:9999
Accepted connection from 127.0.0.1:12345
Hello from client 1
Accepted connection from 127.0.0.1:12346
Hello from client 2
Hello from client 2
Hello from client 1
```

Q5. Write a Socket based program server-client to simulate Simple File Transfer Protocol using TCP Sockets.

Code:

```
#SERVER CODE
import socket

SERVER_HOST = '0.0.0.0'
SERVER_PORT = 8000
BUFFER_SIZE = 4096


def send_file(conn, filename):
    with open(filename, 'rb') as f:
        while True:
            data = f.read(BUFFER_SIZE)
            if not data:
                break
            conn.sendall(data)
    conn.shutdown(socket.SHUT_WR)


def main():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((SERVER_HOST, SERVER_PORT))
        s.listen(1)
        print(f'Server listening on {SERVER_HOST}:{SERVER_PORT}...')
```

```
        while True:
            conn, addr = s.accept()
            print(f'Connected by {addr}')

            filename = conn.recv(BUFFER_SIZE).decode()
            print(f'Receiving file "{filename}"...')

            send_file(conn, filename)

            print(f'File "{filename}" sent successfully')

            conn.close()


if __name__ == '__main__':
    main()

#CLIENT CODE
import socket

SERVER_HOST = 'localhost'
SERVER_PORT = 8000
BUFFER_SIZE = 4096


def receive_file(conn, filename):
    with open(filename, 'wb') as f:
        while True:
            data = conn.recv(BUFFER_SIZE)
            if not data:
                break
            f.write(data)


def main():
    filename = input('Enter filename to download: ')

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((SERVER_HOST, SERVER_PORT))

        s.sendall(filename.encode())

        print(f'Receiving file "{filename}"...')

        receive_file(s, filename)

        print(f'File "{filename}" received successfully')

    print('Connection closed')
```

```
if __name__ == '__main__':
    main()
```

SnapShot:

Server output:

arduino Copy code

```arduino
Server listening on 0.0.0.0:8000...
Connected by <client_address>
Receiving file "example.txt"...
File "example.txt" sent successfully
```

Client output:

mathematica Copy code

```mathematica
Enter filename to download: example.txt
Receiving file "example.txt"...
File "example.txt" received successfully
Connection closed
```

Q6.

Code:

```
#SERVER CODE
import socket

DNS_TABLE = {
    '127.0.0.1': 'localhost',
    '192.168.1.1': 'router',
    '8.8.8.8': 'google-public-dns-a.google.com',
    '8.8.4.4': 'google-public-dns-b.google.com',
}

SERVER_HOST = '0.0.0.0'
SERVER_PORT = 8000
BUFFER_SIZE = 4096


def lookup_hostname(ip):
    return DNS_TABLE.get(ip, 'unknown')
```

```
def main():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((SERVER_HOST, SERVER_PORT))
        s.listen(1)
        print(f'Server listening on {SERVER_HOST}:{SERVER_PORT}...')

        while True:
            conn, addr = s.accept()
            print(f'Connected by {addr}')

            ip = conn.recv(BUFFER_SIZE).decode()
            print(f'Received request for IP {ip}')

            hostname = lookup_hostname(ip)
            conn.sendall(hostname.encode())

            print(f'Sent hostname {hostname} for IP {ip}')

            conn.close()


if __name__ == '__main__':
    main()

#CLIENT CODE
import socket

SERVER_HOST = 'localhost'
SERVER_PORT = 8000
BUFFER_SIZE = 4096


def main():
    ip = input('Enter IP address to look up: ')

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((SERVER_HOST, SERVER_PORT))

        s.sendall(ip.encode())

        hostname = s.recv(BUFFER_SIZE).decode()

        if hostname == 'unknown':
            print(f'Could not find hostname for IP {ip}')
        else:
            print(f'Hostname for IP {ip} is {hostname}')

    print('Connection closed')


if __name__ == '__main__':
    main()
```

SnapShot:

```
Enter IP address to look up: 8.8.8.8
Hostname for IP 8.8.8.8 is google-public-dns-a.google.com
Connection closed
```