Aman Kumar
2020csb1153

**Q1.**

To run the Program:

python3 server.py
python3 client.py

The files which need to be transferred should be kept on the files folder.
change the variable filename to the name of the file to be transferred. Upon running client.py the file will be transferred and stored in the Q1 directory.

```
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4# cd Q1
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q1# python3 server.py
Waiting for a connection...
Connected by ('127.0.0.1', 54261)
Receiving file: text.txt
File received and saved successfully.
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q1#
```

```
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4# cd Q1
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q1# python3 client.py
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q1#
```

**Q2.**

To run the Program:

python3 server.py
python3 client.py

To increase the users, run python3 client.py on new terminals.

Design and explaination:

1. server.py file will be run first.
2. To have n number of users run client.py file n times on different terminals.
3. Upon running the client file the user will be prompted to either register himself or login if he is an existing user.
4. After logging in, he will have 4 options, QUIT – to quit the program, LIST – to list active users, CREATE – to create a chatroom, JOIN – to join a chatroom.
5. If he chooses CREATE, He has to give unique name to chatroom which will be stored in server.
6. If he chooses JOIN, He has to give the exact name of the chatroom.

7. Upon joining, he will be first shown all the previous chats happened in the room and he will be added to the list of users in the chatroom. Then the user can send messages to the chatroom. He can receive other messages only after he has written a new message in the chatroom
8. All the chat history are saved in a txt file created when the chatroom was created. User can type QUIT_CHAT to exit the chatroom. Then he will again be presented with 4 options as in point no. 4.

```
# Define a list of active clients and chat rooms
active_clients = []
passwords = {}
chat_rooms = {}
chat_history = {}
username = ''
```

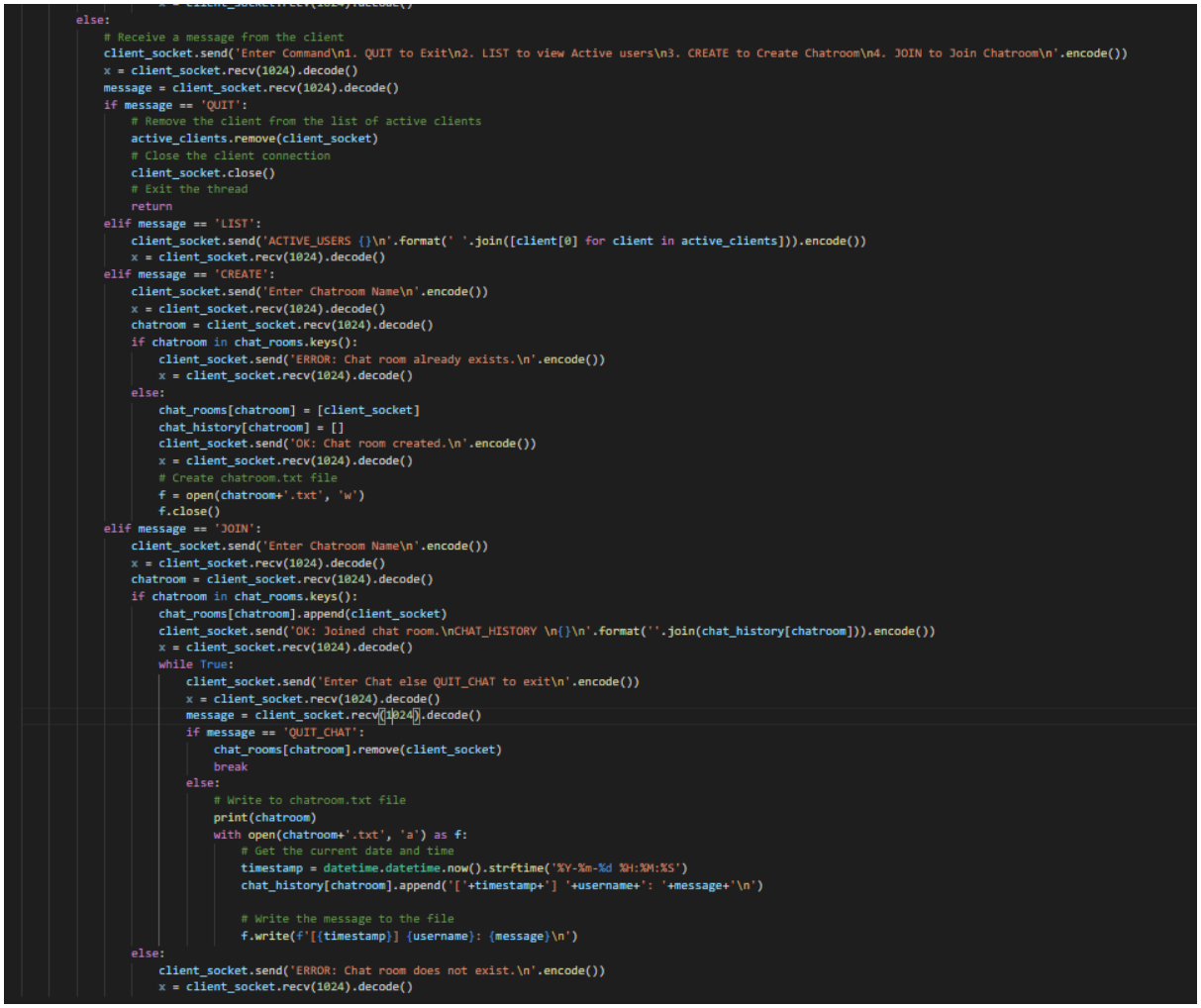
Variables with use same as their name

```
while True:
    if(f == 0):
        client_socket.send('Type 1 to Register and 2 to Login'.encode())
        x = client_socket.recv(1024).decode()
        message = client_socket.recv(1024).decode()
        if message == '1':
            client_socket.send('Enter Username'.encode())
            x = client_socket.recv(1024).decode()
            username = client_socket.recv(1024).decode()
            print(username)
            if len(active_clients)!=0 and username in [client[0] for client in active_clients]:
                client_socket.send('ERROR: Username already taken.'.encode())
                x = client_socket.recv(1024).decode()
            else:
                client_socket.send('Enter Password'.encode())
                x = client_socket.recv(1024).decode()
                password = client_socket.recv(1024).decode()
                passwords[username] = password
                active_clients.append((username, client_socket))
                client_socket.send('OK: Registration successful.\n'.encode())
                x = client_socket.recv(1024).decode()
                f = 1
```

Server Sided Code for Registration

```
        elif message == '2':
            client_socket.send('Enter Username'.encode())
            x = client_socket.recv(1024).decode()
            username = client_socket.recv(1024).decode()
            client_socket.send('Enter Password'.encode())
            x = client_socket.recv(1024).decode()
            password = client_socket.recv(1024).decode()
            if username in [client[0] for client in active_clients] and passwords[username] == password:
                client_socket.send('OK: Login successful.\nACTIVE_USERS {}\n'.format(' '.join([client[0] for client in active_clients])))
                x = client_socket.recv(1024).decode()
                f = 1
            else:
                client_socket.send('ERROR: Incorrect Credentials.\n'.encode())
                x = client_socket.recv(1024).decode()
```

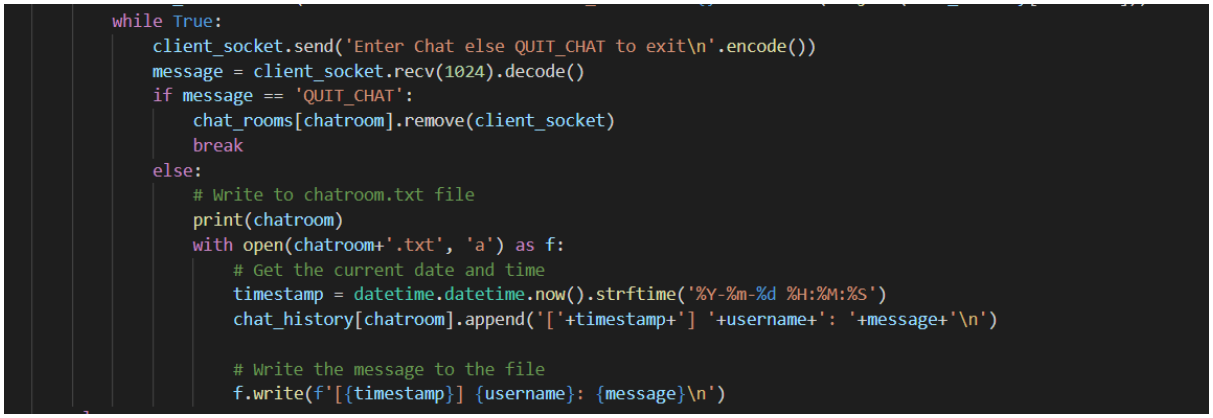
Server Sided Code for Login



```
else:
    # Receive a message from the client
    client_socket.send('Enter Command\n1. QUIT to Exit\n2. LIST to view Active users\n3. CREATE to Create Chatroom\n4. JOIN to Join Chatroom\n'.encode())
    x = client_socket.recv(1024).decode()
    message = client_socket.recv(1024).decode()
    if message == 'QUIT':
        # Remove the client from the list of active clients
        active_clients.remove(client_socket)
        # Close the client connection
        client_socket.close()
        # Exit the thread
        return
    elif message == 'LIST':
        client_socket.send('ACTIVE_USERS {}\n'.format(' '.join([client[0] for client in active_clients])).encode())
        x = client_socket.recv(1024).decode()
    elif message == 'CREATE':
        client_socket.send('Enter Chatroom Name\n'.encode())
        x = client_socket.recv(1024).decode()
        chatroom = client_socket.recv(1024).decode()
        if chatroom in chat_rooms.keys():
            client_socket.send('ERROR: Chat room already exists.\n'.encode())
            x = client_socket.recv(1024).decode()
        else:
            chat_rooms[chatroom] = [client_socket]
            chat_history[chatroom] = []
            client_socket.send('OK: Chat room created.\n'.encode())
            x = client_socket.recv(1024).decode()
            # Create chatroom.txt file
            f = open(chatroom+'.txt', 'w')
            f.close()
    elif message == 'JOIN':
        client_socket.send('Enter Chatroom Name\n'.encode())
        x = client_socket.recv(1024).decode()
        chatroom = client_socket.recv(1024).decode()
        if chatroom in chat_rooms.keys():
            chat_rooms[chatroom].append(client_socket)
            client_socket.send('OK: Joined chat room.\nCHAT_HISTORY \n{}\n'.format(''.join(chat_history[chatroom])).encode())
            x = client_socket.recv(1024).decode()
            while True:
                client_socket.send('Enter Chat else QUIT_CHAT to exit\n'.encode())
                x = client_socket.recv(1024).decode()
                message = client_socket.recv(1024).decode()
                if message == 'QUIT_CHAT':
                    chat_rooms[chatroom].remove(client_socket)
                    break
                else:
                    # Write to chatroom.txt file
                    print(chatroom)
                    with open(chatroom+'.txt', 'a') as f:
                        # Get the current date and time
                        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                        chat_history[chatroom].append('['+timestamp+'] '+username+': '+message+'\n')

                        # Write the message to the file
                        f.write(f'[{timestamp}] {username}: {message}\n')
        else:
            client_socket.send('ERROR: Chat room does not exist.\n'.encode())
            x = client_socket.recv(1024).decode()
```

Server Sided code for the 4 operations as mentioned in Point 4 in Design



```
while True:
    client_socket.send('Enter Chat else QUIT_CHAT to exit\n'.encode())
    message = client_socket.recv(1024).decode()
    if message == 'QUIT_CHAT':
        chat_rooms[chatroom].remove(client_socket)
        break
    else:
        # Write to chatroom.txt file
        print(chatroom)
        with open(chatroom+'.txt', 'a') as f:
            # Get the current date and time
            timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            chat_history[chatroom].append('['+timestamp+'] '+username+': '+message+'\n')

            # Write the message to the file
            f.write(f'[{timestamp}] {username}: {message}\n')
```

Server Sided code for chatting and saving chat history

```
while True:
    message = sock.recv(1024).decode()
    sock.sendall('1'.encode())
    print(message)
    if message == 'Type 1 to Register and 2 to Login':
        inp = input()
        sock.sendall(inp.encode())
        if inp == '1':
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
            inp = input()
            sock.sendall(inp.encode())
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
            if message == 'ERROR: Username already taken.':
                continue
            inp = input()
            sock.sendall(inp.encode())
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
        elif inp == '2':
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
            inp = input()
            sock.sendall(inp.encode())
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
```

Client Sided Code for Registration and Login

```
    else:
        inp = input()
        sock.sendall(inp.encode())
        if inp == 'QUIT':
            break
        elif inp == 'LIST':
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
        elif inp == 'CREATE':
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
            inp = input()
            sock.sendall(inp.encode())
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
        elif inp == 'JOIN':
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
            inp = input()
            sock.sendall(inp.encode())
            message = sock.recv(1024).decode()
            sock.sendall('1'.encode())
            print(message)
            if message.startswith('OK: Joined chat room.\n'):
                len = 0
                with open(inp+'.txt') as f:
                    for i, l in enumerate(f):
                        len = i
                filename = inp
                while True:
                    message = sock.recv(1024).decode()
                    sock.sendall('1'.encode())
                    with open(filename+'.txt') as f:
                        for i, l in enumerate(f):
                            if i > len:
                                print(l)
                                len = i
                    print(message)
                    inp = input()
                    sock.sendall(inp.encode())
                    if inp == 'QUIT_CHAT':
                        break
```

Client Sided code for 4 operations in point 4 and sending messages

```
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4# cd Q2
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q2# python3 client.py
Welcome to the chat server!
Type 1 to Register and 2 to Login
1
Enter Username
Aman
Enter Password
Aman
OK: Registration successful.

Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom

LIST
ACTIVE_USERS Aman Amank

Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom
```

Registration and listing active users

```
Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom

CREATE
Enter Chatroom Name

CHAT
OK: Chat room created.

Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom
```

Chatroom Created

```
Type 1 to Register and 2 to Login
2
Enter Username
Aman
Enter Password
Aman
OK: Login successful.
ACTIVE_USERS Aman Amank

Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom

JOIN
Enter Chatroom Name

CHAT
OK: Joined chat room.
CHAT_HISTORY


Enter Chat else QUIT_CHAT to exit

Hi
Enter Chat else QUIT_CHAT to exit

Hello
[2023-04-24 21:01:54] Aman: Hello

Enter Chat else QUIT_CHAT to exit
```

Logging in and joining chatroom

```
Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom

JOIN
Enter Chatroom Name

CHAT
OK: Joined chat room.
CHAT_HISTORY
[2023-04-24 21:01:53] Aman: Hi
[2023-04-24 21:01:54] Aman: Hello


Enter Chat else QUIT_CHAT to exit

Hi
[2023-04-24 21:02:29] Amank: Hi

Enter Chat else QUIT_CHAT to exit

Whatsup
[2023-04-24 21:02:34] Amank: Whatsup

Enter Chat else QUIT_CHAT to exit
```

Joining and chat history

```
Enter Chat else QUIT_CHAT to exit

Hi
[2023-04-24 21:02:29] Amank: Hi

[2023-04-24 21:02:34] Amank: Whatsup

[2023-04-24 21:02:53] Aman: Hi

Enter Chat else QUIT_CHAT to exit

QUIT_CHAT
Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom
```

Chatting and QUIT_CHAT

```
Enter Command
1. QUIT to Exit
2. LIST to view Active users
3. CREATE to Create Chatroom
4. JOIN to Join Chatroom

QUIT
```

Quitting

CHAT.txt

Q2 > CHAT.txt

```
1  [2023-04-24 21:01:53] Aman: Hi
2  [2023-04-24 21:01:54] Aman: Hello
3  [2023-04-24 21:02:29] Amank: Hi
4  [2023-04-24 21:02:34] Amank: Whatsup
5  [2023-04-24 21:02:53] Aman: Hi
6
```

Chat saved in CHAT.txt

**Q3**

To run the program:

gcc server.c -o s

./s

gcc client.c -o c

./c

Upon running client the user will have to write the postfix notation in the terminal and server will reply with the solved value of the postfix notation.

```
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q3# gcc server.c -o s
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q3# ./s
Successfully Binded
Waiting for incoming connections...
Waiting for incoming connections...
```

```
root@LAPTOP-MA03V1M3:/mnt/c/Users/Aman/Documents/C++/Assignments/CS304/Lab4/Q3# ./c
Connected to server
Please enter the message to the server: 1 2 +
Server replied: 3.00
Please enter the message to the server: 2 3 *
Server replied: 6.00
Please enter the message to the server: 4 7 3 + -
Server replied: -6.00
Please enter the message to the server: 30 1.0 /
Server replied: 30.00
Please enter the message to the server: 22 44 +
Server replied: 66.00
Please enter the message to the server: 3 4 *
Server replied: 12.00
Please enter the message to the server:
```

Q3 > server_records.txt

```
1 1 2 + 3.00 13
1 2 3 * 6.00 5
1 4 7 3 + - -6.00 10
1 30 1.0 / 30.00 19
1 22 44 + 66.00 28
1 3 4 * 12.00 4
```