Assignment 3
Multi Client Chat server

In this assignment you need to implement a client-server chat application using TCP sockets. There will be a single server and multiple clients communicating with the server. Each client process will open a new connection with the server and to handle each client request, the server will create a new child process.

Part 1 : Simple Chat with Group Functionality.

1. Connection Establishment : The client should send the connection request to server. The server should reply client with appropriate messages if connection can be established or not.

Successful :
If the connection can be established successfully, then generate appropriate identifiers for the client and store them at server. Identifiers includes - Unique Id (5 Digit Random Number). After connection is established, send client the above details with a welcome message.

Unsuccessful :
If the number of clients connected are already 5 then no further client is allowed to connect and server should inform client that "Connection Limit Exceeded !!".

Client details table - Keep the details in shared memory. Use synchronization while reading and writing into the memory.

2. Data Transfer Phase : Client should send a query message to server asking the details of online clients. Server should send the details of all the online clients[**each with unique key**]. After receiving details, client can transfer messages to any other client of choice by using its unique id. (Note that this is a one to one communication).

Message info table - Create another shared table which stores message details.

There can be a situation when a client A gets the list of online clients and before it can send any message to client B, client B goes offline. The sender in this case should be notified that client is now disconnected and that message should be discarded.

3. Connection Termination : In order to disconnect, the client should send an EXIT message to the server. The server should **notify all other clients** with the details of client which is going to disconnect. Then terminate the client process.

4. Broadcast : A client should be able to send a broadcast message by typing "/broadcast ". The message should be delivered to all clients connected to the server.

Functionalities to be included:

1. **/active** : To display all the available active clients that are connected to the server.
2. **/send <dest client id> <Message>** : To send message to the client corresponding to its unique id.
3. **/broadcast <Message>** : Message should be broadcasted to all the active clients.
4. **/makegroup <client id1> <client id2> … <client idn>** : A group with unique id will be made including all the mentioned clients along with the admin client.
5. **/sendgroup <group id> <Message>**: The sender should be in the group to transfer the message to all his peers of that group. The message should be send to all the peers along with group info.
6. **/activegroups** : To display all the groups that are currently active on server and the sender is a part of.

Simple scenario

Things to remember:

Server Side

1. Use fork() when client is connected.
2. Make the client socket non blocking.
3. Use send() and recv() system call. It will make your life easier.
4. Each client's process will check in message details table. It it founds an entry with dest_id as it's sock_id, then send the message to the client. Remove that entry.
5. Log messages on server (server's terminal).
6. Handle cases when a client is no longer a part of a group, and similar cases.
7. If a client who initiate the group quits from the server than every group which he owns should be automatically deleted.

Client Side

1. Reading from the standard input and writing to the server (send() system call) and reading from the server (recv() system call) will be handled by different processes.

You can make your own architecture. But try to handle all the corner cases.

Things to remember:

- Include **Readme.txt** file, write 2-3 lines on what functionalities have been implemented and how to execute your code.
- Include **makefile.**
- Follow proper naming conventions.

<<Please do not copy code from internet or from your classmates. >>

If found copied then straight away zero will be given and can lead to your suspension.