CS446/646 Computer Communications and Networks
Homework 1 (Programming)
Due date: 2/20/2018, 12:30PM (submission of source codes by email to duc.tran@umb.edu).

Your source code and pre-built project must be placed in your directory in the shared drive by the due date and time. Otherwise, the late policy applies. Hardcopies are not required, but preferred.

The purpose of this assignment is to get you familiar with socket programming. You are expected to design an instant-messenger application called *SimpleChat* that allows two remote users to exchange command messages typed from the keyboard. The messages have to be instant meaning that there should not be a noticeable delay for a message sent by a user to be seen by the other user. The connections used in this application should be TCP. You need to write two programs for this application: SimpleChatServer and SimpleChatClient. They are explained as follows:

1. SimpleChatServer
    a. Run as the central server, whose role is to manage a list of existing users: their IP, name, and status (busy or idle). "busy" means the user is in a conversation, and an "idle" user is available for any conversation
    b. You need to provide a port for this server via the command line when this program is first started
    c. When a user starts, it sends a JOIN command to the server which will add the user into the list of existing users. This JOIN message has to include a nickname that the user wants to use. If the nickname already exists, the server sends a NAMECONFLICT message back to the user.
    d. If a user sends a LIST command to the server, the server sends back to this user the list of existing users and their busy/idle status. The user needs to display this list on its screen
    e. If a user wants a conversation with somebody, e.g., Darren, in this list, it sends a "TALK Darren" command to the server. The server will return the IP address of Darren back to the user. The user will use this address to connect to Darren and starts a chat conversation. The port for this conversation is set by the server via the command line when it first starts
    f. If a user starts a conversation with another, they both have to send a BUSY message to the server so that the server knows that these users are not available for any other conversation.
    g. When this conversation is close, both users have to send an IDLE message to the server so that the server knows that these users are now available.
    h. When a user closes its program, a CLOSE message is sent to the server. The server will delete this user from the list of existing users.
2. SimpleChatClient
    a. Running as a shell (just like Windows console screen)
    b. The user types commands from the keyboard. The user should be able to type any command at any time, even during a conversation. Therefore, you need to be able to distinguish between a command and a conversation message typed from the keyboard. (One way to do so is to type "$" before any command name. For instance, "$LIST" means the LIST command.)
    There are different commands:
        i. JOIN nickname: explained above. A user cannot register to the server more than one time.
        ii. LIST: get the list of online users and display on the screen.
        iii. TALK buddyname: create a connection with *"buddyname"* and start exchanging messages with him. Make sure that "buddyname" must be currently available, otherwise an error message should be displayed. If connection is successful, the user can start chatting using the keyboard. It types "quit" if it wants to end the conversation. Once a user is in a conversation, it cannot participate in another until it types "quit" to close the current conversation
        iv. EXIT: exit the program. A CLOSE message will be sent to the server as a result of this command

You are free to design your screen, and use more commands and message types if that makes your programming easier.