For this assignment, the server must allow multiple clients connect the server at the same time. You are required to implement a multithreaded server (using Pthread or Java thread) and a client that can monitor the server’s message and the user’s input at the same time (e.g. using select() or threads).

You will need to implement, ADD, DELETE, LIST, SHUDOWN, QUIT LOGIN, LOGOUT, WHO, LOOK, on the client side and the corresponding functions on the server side.

**ADD**

Add a name and number record to the address book. A client sends the ASCII string “ADD" followed by a space, followed by a First Name, followed by a space, followed by a Last Name, followed by a space, followed by a Phone Number, and followed by the newline character (i.e., '\n').

When the server receives an ADD command from a client, it will add a new record in the address book and return the “200 OK” message.

A client-server interaction with the ADD command thus looks like:

c: ADD Jinhua Guo 313-583-6439

s: 200 OK

The new Record ID is 1001

**DELETE**

Delete a record from the address book. A client that sends the ASCII string "DELETE", followed by a space, followed by a Record ID, followed by the newline character (i.e., '\n'). The client should then wait for the server to return a "200 OK" message (indicating that the record is successfully deleted), or a "403 The Record ID does not exist."

A client-server interaction with the DELETE command thus looks like:

c: DELETE 1001  
s: 200 OK

**LIST**

List all records in the address book.

A client-server interaction with the LIST command looks like:

c: LIST

s: 200 OK

The list of records in the book:

1001 Jinhua Guo 313-583-6439

1002 John Smith 313-583-1234

1003 Mary Miller 313-594-4567

**SHUTDOWN**

The SHUTDOWN command, which is sent from the client to the server, is a single line message that allows a user to shutdown the server. A user that wants to shutdown the server should send the ASCII string "SHUTDOWN" followed by the newline character (i.e., '\n').

Upon receiving the SHUTDOWN command, the server should return the string "200 OK" (terminated with a newline), close all open sockets and files, and then terminate.

A client-server interaction with the SHUTDOWN command looks like:

c: SHUTDOWN  
s: 200 OK

**QUIT**

Terminate only the client. The client exits when it receives the confirmation message from the server.

A client-server interaction with the QUIT command looks like:

c: QUIT

s: 200 OK

Note, “300 invalid command” or “301 message format error” should be returned to the client, if a server receives an invalid command or the command in the wrong format.

**LOGIN**

Identify the user to the remote server. A client that wants to login should begin by sending the ASCII string “LOGIN" followed by a space, followed by a UserID, followed by a space, followed by a Password, and followed by the newline character (i.e., '\n').

Your server should be initialized with the UserIDs and Passwords of at least four users who will be allowed to execute the ADD, DELETE, and SHUTDOWN (the root user only) commands at the server. However, a user is allowed to execute the LIST, WHO, LOOK, and QUIT commands anonymously (without login).

When the server receives a LOGIN command from a client, it should check if the UserID and Password are correct and match each other. If they are not correct or don’t match each other, the server should return the string “410 Wrong UserID or Password,” otherwise it should return the “200 OK” message.

A client-server interaction with the LOGIN command thus looks like:

c: LOGIN john john01

s: 200 OK

**LOGOUT**

Logout from the server. A user is not allowed to send ADD, DELETE, and SHUTDOWN commands after logout, but it can still send LIST, LOOK, WHO, and QUIT commands.

A client-server interaction with the LOGOUT command looks like:

c: LOGOUT

s: 200 OK

**WHO**

List all active users, including the UserID and the users IP addresses.

A client-server interaction with the WHO command thus looks like:

c: WHO

s: 200 OK

The list of the active users:

john 141.215.69.202

root 127.0.0.1

**LOOK**

Look up a name in the book. Display the complete name and phone number record. A client sends the ASCII string “LOOK" followed by a space, followed by a number (1 – look for the first name, 2 – look for the last name, 3 – look for the phone number), followed by a name or a phone number, and followed by the newline character (i.e., '\n').

When the server receives an LOOK command from a client, it will look up the name or the phone number in the address book. When there is a match, it returns the “200 OK” message and all the matched record (s). If there is no match, it returns the “404 Your search did not match any records”.

A client-server interaction with the LOOK command thus looks like:

c: LOOK 2 Miller

s: 200 OK

Found 2 match

1001 David Miller 313-510-6001

1004 John Miller 315-123-1345

c: LOOK 3 313-231-1324

s: 404 Your search did not match any records

**ADD** and **DELETE**

The basic requirement is the same as the assignment 1 except that a user needs to login first to execute these commands. If a user has not logged in, the server will return a "401 You are not currently logged in, login first" message

**LIST**

Same as the assignment 1.

**SHUTDOWN**

The basic requirement is the same as the assignment 1 except that only the “root” user can execute this command. When your server receives a SHUTDOWN command from a client, it should check if the current user is the root. If it is not the root user, the server should return a "402 User not allowed to execute this command" message.

In addition, **the SHUTDOWN will make all clients and the server terminate.**

A client-server interaction with the SHUTDOWN command thus looks like:

c: SHUTDOWN  
s: 200 OK

At the windows of all clients

s: 210 the server is about to shutdown ……

**QUIT**

If a user logged in from the current client, QUIT also logout the user.

*Note, a user can execute WHO, LIST, and QUIT commands before logged in.*

## Format

You should form a team of two students (or work individually) and then jointly design your project. While each project should be an integrated effort, you should identify in your README file what part of the project each member is responsible for.

I assume you will work in the same group as you did for project 1. If there is any change, please let me know.

## Programming Environment

You can use either C++. please don’t use any GUI interface.

1. **Requirements**

* implement all nine commands: ADD, DELETE, LIST, QUIT, SHUTDOWN, LOGIN, LOGOUT, WHO, LOOK
* all users share the same address book. The users information should be maintained by the server. You must have the following users (lower case) in your system:

UserID Password

root root01

john john01

david david01

mary mary01

* make sure that you do sufficient error handling such that a user can't crash your server. For instance, what will you do if a user provides invalid input?
* the client should be able to connect to the server running on any machine. Therefore, the server IP address should be a command line parameter for the client program.
* the server should print out all messages received from clients on the screen.
* when the previous client exits, the server should allow the next client connect.
* your source codes must be commented
* include a **README** file in your submission.
* include a **Makefile** in your submission.

Note 1, in your README file, the following information should be included: the functions that have been implemented, the instructions about how to compile and run your program, known bugs, and *sample outputs of a complete test of all commands you implemented*.

Note 2, your Makefile might be the exact same as the sample Makefile if your file names are the same as those of the sample codes.