Important Notes:

- This lab has automated evaluation.
- Do not include unnecessary printf statements. Only print the strings mentioned in question.
- Any strings which are required to be sent by the client or server must be exactly as specified.
- Sending and receiving data requires the connection function to be implemented. If your connection function does not work correctly, the latter parts will not be evaluated.
- You'll receive a script containing sample test cases, and your submission will be evaluated against hidden test cases to determine the final score. So, please refrain from hard coding.
- Store the server.c code file, the client.c code file in a directory named “impl” and zip this folder to create a file named <id_no>_lab4.zip (The zip file should only contain the “impl” folder with the two c files and nothing else.)

Design a client-server system that follows the polling mechanism. Polling in client-server systems is when the server sequentially queries clients for responses. There is one server and 3 clients connecting to the server.

(NOTE: The implementation should be such that it can easily be scaled to accommodate more clients)

The server waits for all the clients to connect to it before the polling mechanism begins. After every client connects to the server , the server sends them the “NAME\n” message as a response to which they should send their username (given as command line argument) to the server. After all the clients are connected to the server, the server polls each client one by one (in the same sequence as they connected to the server) by sending “POLL\n” as the keyword to the client. The polling should work for as many cycles (each client will be polled at least twice in the order client1, client2, client3, client1,…) until all clients EXIT. The client polling order should be the same as the order in which they connect to the server. CLIENTS WHICH HAVE EXITED SHOULD NOT BE CONSIDERED FOR POLLING.

When a client receives the “POLL\n” from the server they can reply with a range of commands that are taken from stdin (entered by the user associated with that client). These commands include “NOOP\n”, “LIST\n”, “MESG:<message>\n”, and “EXIT\n.”

CLIENT Command line inputs -

- argument 1 - server IP
- argument 2 - server port
- argument 3 - user name of the client

CLIENT COMMANDS:

- “NOOP\n”
    - This command is used to tell the server that this client does not have a message to send nor does it want anything from the server.
    - When the server sees this command it continues with the polling process.

- “EXIT\n”
    - This command is used by the client to let the server know that it is going to terminate so that the server can deallocate resources associated with that client and remove it from the pool of clients that have to be polled.
    - After sending this command to the server the client must print “CLIENT TERMINATED: EXITING……\n” before exiting.
    - When the server receives this message it must remove the client from the polling pool and deallocate resources associated with that client
    - The server continues the polling process.
    - A client can only exit after it receives the POLL command from the server.

- “MESG:<message>\n”
    - This command is used to send a message to the server from the client.
    - After the server receives this message from the client it must print it in the format “<clientusername>:<message>\n”
    - After printing the message the server sends the “POLL\n” message to the next client as determined by the polling mechanism
    - A client can send only one message to the server in one POLL

- “LIST\n”
    - This command is used by a client to request the server for the names of all the clients that are currently connected to the server.
    - When the server receives this command it sends a string containing the names of all the clients that are currently connected to it in the format “<name1>:<name2>:<name3>\n”
    - The client on receiving this list must print the list of clients on its terminal. eg., if the server sends client1:bob:eve, then the client prints
      1. client
      2. bob
      3. eve

      raw format = “1. client\n2. bob\n3. eve\n”
    - After it has sent the list of names to the client it moves on to the next client as dictated by the polling mechanism.
    - NOTE: The list of names must not include the clients that have already been terminated with the use of the “EXIT\n” command

- Any other command
    - The client just sends it to the server
    - On receiving this command the server prints “INVALID CMD\n” and then moves on to the next client.

A client is always in receiving mode, i.e., it is waiting for the POLL command from the server, until it receives the “POLL\n” keyword. When it receives the “POLL\n” keyword, it has to transmit a message

(one keyword among “NOOP\n”, “LIST\n”, “EXIT\n”, and “MESG:<message>\n”) and switch over to the receiving mode again.

SERVER Command line inputs

- argument 1 - server ip
- argument 2 - server port

SERVER COMMANDS:

- “NAME\n”
    - This is the first command sent by the server to a new client.
    - When the client receives this command it must print “INITIALIZING……\n”
    - Once the above string has been printed in the client's terminal it should automatically send its username (provided as a command line input) to the server in the format “<name>\n”
    - The server must store this username for future use (if “LIST\n” is invoked by a client)

- “POLL\n”
    - This command is sent by the server to the client when it is that client’s chance to send a message to the server.
    - The polling algorithm is run to identify the next client and this message must be sent to the client.
    - When the client receives the “POLL\n” command from the server it must print “ENTER CMD: ” in its terminal to indicate to the user that the client is waiting for an input.
    - Once the input has been received further operations must be done based on the input

POLLING MECHANISM:

The server must keep track of the order in which the clients connected to it and this order is the order in which the “POLL\n” commands are sent out to the clients. Once the last client to join has been polled the next client to follow will be the first client to join. Clients which have exited MUST NOT be included in the polling process. For example let A, B, C be the clients that connected to the server first, second and last respectively. Then the polling order will be:

A -> B -> C -> A -> **B** -> C -> A -> C -> A -> C

B uses the “EXIT\n” command to terminate the second time it’s polled

IMPORTANT:

- Polling must be done till all the clients have exited. Once all the clients have exited the server first prints “SERVER TERMINATED: EXITING……\n” before exiting.
- Polling must only start once **ALL** clients have connected to the server otherwise they will never be given an opportunity to do so.

Keywords to implement:

At client: NOOP, LIST, MESG, EXIT

At Server: NAME, POLL

Inputs and Checking:

1. The server, as a command-line argument takes:
    a. IP address to bind to
    b. accepts the port to which it should bind

2. The clients (start 3 clients), as a command-line argument takes:
    a. The server’s IP address
    b. Server’s binding port number
    c. The user’s username (you don’t need to perform validation. Just keep the usernames different from each other)

3. The name of the server file must be server.c

4. The name of the client file must be client.c

5. After al 3l clients have connected, the server sends a POLL message to one client, waits for its response and does the operation required. The polling should work for as many cycles (each client will be polled at least twice in the order client1, client2, client3, client1,…) until all clients EXIT. The client polling order should be the same as the order in which they connect to the server.

6. To check for the accuracy of your code you can run check_server.py and check_client.py for the server and client respectively.

7. These python scripts take the ip and port number of the server as command line arguments.

8. To run them type: python3 <script_name>.py <ip> <port>

MARKING SCHEME
CLIENT - 3 MARKS

- Providing the server with name when connection is established - 1 mark
- Checking if the commands work - 0.5 x 4 = 2 marks

SERVER - 3 MARKS

- Checking the output on the servers terminal - 1 mark
- Polling algorithm - 2 marks

What to submit:

The “impl” folder containing server.c and client.c must be zipped to create a file named <id_no>_lab4.zip