Network Project Specification:
Web Server in C/C++
Client (mozilla firefox)

NOTE: The following is subject to change and will be updated as discoveries are made and discussed during class.

Client side requirements:

1. Provides a login to the system. The login information is stored on the server.
2. Client initiates the connection to server on port 4242.
3. A web browser (mozilla firefox) is the basis for the client side. The browser window is the canvas to put a face on the application.
4. You may use html, style sheets and java scripts, i.e. a web page. I strongly recommend using a single web page and utilizing javascripts to dynamically update the web page as information is collected from the user and sent/received to/from the server.
5. DO NOT use react, angular, etc. Must be implemented with standard javascripts/css/html.

Server Requirements:

1. Server should handle HTTP protocol to handle connections from browser RFC here: https://datatracker.ietf.org/doc/html/rfc2616
2. Server should handle Websocket protocol to handle websocket connections from javascript (in browser) RFC here: https://datatracker.ietf.org/doc/html/rfc6455
3. The server handles the game via websocket messages (see above RFC and Mozilla docs for JS websocket API: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API). Design your own protocol within websocket messages (frames) to communicate game state between clients and server.
4. Modular code, well documented. Functions should be held to +/- 20 lines of code.
5. Use port 4242
6. For websocket connection use route (example): http://localhost:4242/ws (or http://<ip address?:4242/ws)
7. Must be written using standard socket programming. No outside libraries to hide details. If in doubt, ask before using.

Suggestions:

- Use lucy.highpoint.edu to host your static files (html, css, javascript) to get websocket working first, then implement http later

Levels:

1. Server can handle 1 game, 2 players
    a. Initial html page (containing javascripts and style sheets) is available on lucy.highpoint.edu. See below.
    b. The game server can be anywhere your team can access. Several options here. A proxmox server, a dell blade in the networking racks, a workstation in the lab, etc.
    c. No login – two users join up and play the game.
    d. The javascript makes a websocket connection to the server on port 4242 - written in C/C++.
    e. The javascript monitors game activities on the browser. When a user makes a move, a message (frame) is sent to the server for processing. At that point, the server may send a response back to one or both players.
2. Server can handle 1 game, 4 players

    a. Each user must login to play the game
    b. Increase the number of users – up to 4.
    c. The initial html page (containing javascripts and style sheets) is from your server port 4242, i.e. lucy is removed from the picture.
3. Many games (2-4 players)
    a. All of level 2.
    b. Server can now handle multiple games simultaneously.
    c. Each game, 2-4 players.
    d. Update the server to use secure websockets.

The levels will evolve into grade levels.

NOTE: Must have working Web Server with HTTP and WebSocket implemented before starting work on game (i.e. every group should have basically same code before starting game implementation). Also, while you need to setup the general communication between the client (browser) and the server first, each team needs to develop a protocol, the communication language, of the messages that will be passed between the client and server during game play.

Lower-level Game Suggestions (try to be turn based):

1. Battleship
2. Connect 4
3. Tic tac toe
4. Hang man

Higher-level Game Suggestions (ONLY IF YOU HAVE TIME)

1. Pong
2. Monopoly
3. Poker

About lucy.highpoint.edu

1. We have setup apache2 on lucy to handle either secure or unsecure web pages.
2. Use your username/password from department to login.
3. If you don't already have one, create a directory named public_html.
4. Your home directory "/home/students/<username> and the public_html directory needs permissions set to allow world execute permission. This translates to a 701 when using chmod command.
5. Create an index.html file inside the public_html directory.
6. If you enter the url in a browser, http://lucy.highpoint.edu/!<username>, this should access the index.html page created in step 5.