COSC 2P13 – Assignment 2 –Shall we play a game?

Developing a networking-enabled game doesn't *have* to be significantly more complicated or difficult than the standalone equivalent. For turn-based games, user input typically comes in a very rigid sequence (e.g. alternating), with limited vocabulary or choices (e.g. chosen column).
Basically, if you *could* play it by having two people alternating clicking buttons or typing simple expressions, then it might be a good candidate for socket programming.

For this assignment, you'll be writing a complete networking-enabled two-player game, that supports multiple concurrent games.
The game will be **one** of Mancala, *or* Connect Four. It doesn't matter which of the two you choose.
Players will be able to connect via a terminal/shell (e.g. using `nc/netcat/ncat`) or via a GUI-based client.

- A terminal-player could be matched up against another terminal-player
- A terminal-player could be matched up against a GUI-player
- A GUI-player could be matched up against another GUI-player

Players won't know the client of their opponent, as the gameplay will be the same on their own ends, regardless. It's fine to assume that the server will simply match up 'every two new connections' together.

Details:

- Most of the 'magic' occurs in the server: accept two connections, and split them off into a thread
- It's up to you whether, after a game finishes, it disconnects or immediately starts a new game
  - So long as it's clear what's going on, of course
- There's no requirement for 'chatting', or other similar features
- In terms of 'input sanitization' and fault-tolerance:
  - This shouldn't be necessary for the GUI-version. At most, you might want to disable buttons (or ignore their events) while waiting for feedback
  - For the terminal-version, this is easiest to implement on the server-side. Let the terminal client send whatever, and then the server can either accept it (and switch to the other client), or defer back to the *same* client to 'go again'
    - The marker won't be *trying* to break anything, but testing over several games is almost guaranteed to trigger at least one or two typos, right?
- Tip: your server doesn't need to distinguish between terminal and GUI clients either! You can easily write your client to expect terminal-friendly text, and simply parse it
- Also tip: if you're using Python, use tkinter for the GUI. You need so little for this that it's easy to learn *very* quickly (or feel free to ask me for help and I'll whip something up). For Java, you don't *have* to use Swing, but it's pretty dang easy (it'd probably be easier than BasicIO)
- Even though it's not on the marking scheme, still include both instructions *and* a sample execution

Marking:

- Server program:
  - Accepting connections, and pairing them up: **2**
  - Separating into threads: **2**
  - Handling IO over sockets: **2**
  - Actual game logic: **4**
  - Error-checking for the game (accept/reject): **2**
- The (GUI) client program:
  - Connecting: **1**
  - Basic control logic: **1**
  - The GUI itself: **1**
- Basic style/commenting: **1**

- General correctness: **4**
    - This one just covers the myriad ways that things can go wrong. Make something that basically works, and you should get ~4/4

So it's out of 20 overall.

There are a few overarching concerns, that don't fit into any particular category:

- If it isn't networked, it gets zero
- If you only allow for a single game at a time, per above that only costs you 3 points total
- This doesn’t give much feedback on how to create the GUI. Don’t make me regret this by submitting a bunch of autogenerated trash. Trash *always* gets a zero, and makes your instructor start questioning whether or not to trust students to take an opportunity to *learn*

Submission

Include *everything* you used to make *everything*.
Bundle everything into a **.zip** file.

- Not a .rar. Not a .7z. Not a .tar.gz
- Ignore this step, and you will receive a zero
    - No, seriously
        - You remember the part about this thing being worth 20% of your final grade, right?
            - I'm not kidding; this is entirely within your control
                - If you think this is getting excessive, this disclaimer gets a bit bigger each term
                    - (For a reason)

Submit the .zip through Brightspace. Remember it can get a bit laggy at times, so don't please don't leave yourself 40 seconds to submit. Or fail to notice the actual time it’s due and leave such little time that the difference between a :55 and a :59 even matters.