Algo Challenge **Connect 4**

Overview: Write an AI to play Connect 4 against other teams.

Description:

Connect 4 is a two-player game dating back to 1974, played on a vertical 8 x 8 board. Two players (X and O) take turns dropping pieces of opposite colors (red and black, respectively) into any of the 8 columns, which then fall straight down the column to occupy the first unoccupied slot.

Although all boards will be 8 x 8, we will provide the starting configuration of the bottom two rows to test your algorithms. The bottom two rows will always be a valid configuration (reachable by a sequence of valid moves by X and O) that does not have 4 pieces in a row. In each *game* between Player X and Player O, X always moves first.

Time allotment:
You will have **exactly 1 second** to make each move (regardless of your coding language). If you take more than 1 second on any move, you will lose the game immediately.

Starter code implementation:
Because we wanted to make this a true algorithmic challenge, not a parsing frustration, we've provided starter code in C, Java, and Python. Our starter code already handles all of the arena update and communication logic and even has a few convenience visualization methods to get you started. All you have to do is change which moves get submitted each turn. Simple, right?

Here are some notes on the starter code implementation:

1. You are free to update the provided code infrastructure, but if you change any of the code infrastructure you do so at your own risk.
2. The board is represented as a two-dimensional int[][] array. The first index is the row / height number (0 to 7 inclusive), and the second index is the column number (0 to 7 inclusive). The height number increases from bottom to top, and the column number increases from left to right. The entries are either EMPTY, SELF, or OPPONENT. Depending on which role you are assigned, SELF will equal PLAYER_X or PLAYER_O as well.
3. To submit a move, you must call submitMove(int) exactly once per turn with an integer between 0 and 7 inclusive.
4. You can play against and test your program manually with the following methods:
    a. Run the program with the `-r` command-line flag. This initializes the initial board to a random state.
    b. Run the program normally and provide the following input on separate lines:
        1. player code (`X` or `O`)
        2. 8-character setup for row 1 (period (`.`) for empty; `X` or `O` for player pieces)
        3. 8-character setup of row 0 (same as above)

    At this point, if your program is playing X, it should output a number corresponding to the column of its move. Otherwise, it should be waiting for you to play first. You play a move against your program by typing in a column, indicating a move. (Your cursor will be right next to your program's move; this is expected.) You can assume that our grading infrastructure will provide only valid input.

During the two hours, you can (but are not required to) submit intermediate versions of your program. If you do, your program is placed into a pool of testing programs. Throughout the two hours, we will be pulling random teams from the pool and running simulation games. The results and playbacks of the games will only be accessible to the teams involved. You will not be able to identify your opponent in any of the games.

Contest structure:
Your program will compete against other programs in your division only in a ladder-style tournament at the end of the two hours. In the ladder, pairs of teams will play a *match* against each other on 5 starting configurations. To be fair, players will play each starting configuration twice, with alternating roles. A win is recorded as 3 points, a tie as 1 point, and a loss as 0 points. The winner of the match is the player with a higher point total after all 10 games. The tournament rules are as follows:

You are initially seeded with your final ranking from the two-hour speed round. Higher seed number teams correspond to the bottom of the ladder. Beginning with the bottommost two teams and climbing up, pairs of adjacent teams will play a match, with the higher-seed-number team challenging the lower-seed-number team. If the higher-seed-number team wins the match, the teams swap places; otherwise, the teams stay in the same order. In either case, the winner of the match is now at the lower seed number and proceeds to challenge the next team up the ladder. This process continues until the top of the ladder is reached.

For example, say there are 5 teams A through E, ranked 1 through 5 initially. Then team E (#5) would play team D (#4). If E scores more points than D in the match, E moves up to (#4) and D drops to (#5); however, if E and D tie or D scores more points, then no swap occurs. Say E wins. Now E (#4) will challenge C (#3). If E wins again, E (#3) will challenge B (#2). Suppose E loses, so E remains at (#3). Now B (#2) will challenge A (#1), and if B wins, then the updated ranking will be BAECD.

To recap, the above paragraph described a *pass* through the rankings. In each pass, you can drop down the ladder at most 1 rank from your starting rank, but you can climb up the ladder any number of ranks. To account for variation, we will make 3 such passes, so you can drop a maximum of 3 ranks but rise any number of ranks. The final ranking after the 3rd pass will determine the overall winners of ProCo 2010.