Section Activity #8
Connect 4

What’s the Point?

We’ve just finished Exam 2. In this Section, we’ll practice some more with methods, with 2D arrays, and with loops. We’ll be using some of the new tricks we’ve learned in class - such as else if, break, and continue.

Pair up!

Remember, we’ll be doing pair programming this semester. So choose a partner (somebody you’ve never worked with in section), and find a computer.

What is Connect 4?

Connect 4 is a simple game of strategy; two players take turns dropping disks into the top of a set of columns. Each disk falls to the bottom of the column, and disks start to stack up over time.

The objective of the game is for one player to make a line of 4 disks in a row - either horizontally, vertically, or diagonally.

https://en.wikipedia.org/wiki/Connect_Four

Part 1: Converting Pseudocode

For this Section, I’ve provided pseudocode for the entire first version of the program. But it’s not very good pseudocode - I’ve left out some key details, and “forgotten” to mention a few necessary steps.

Your job, in Part 1, is to work with your partner to convert the pseudocode to Java. Fill in the details where I’ve left them out. Debug your Java code, and add new code as necessary.

For part 1, you need to have a basic version of the program - one which can play the game (with both red and blue using the keyboard). It will lack some key features; for instance, it doesn’t ever detect when one player has won the game.

✓ CHECKPOINT 1

Raise your hand. Your SL will come over and verify that you completed this step correctly.
Part 2: Game Over

For Part 2, I’d like you to write a method which will check to see if either player has won the game. Call this method from your main() method, and end the program (with some sort of message) if they have.

You may choose how advanced to make this method. If you’d prefer, you may simply check for only a single direction - like only horizontal sets, or only vertical. That’s fine - but you must check for that win condition *everywhere* on the board. This is going to require at least doubly-nested loops (so that you can check all of the positions), and triply-nested will probably be useful, so that you can check all the values. That is, your code will have this overall structure:

```python
checkForGameOver(board):
    for (2d) over all possible starting positions
        for 4 positions in a row
            ...
```

If you have time left over at the end of Section, go back and try to expand your method so that it will check for all of the win conditions: horizontal, vertical, and diagonal.

✓ CHECKPOINT 2

Raise your hand. Your SL will come over and verify that you completed this step correctly.

Part 3: Computer Player

For Part 3, I’d like you to have the computer play one of the players. To make a computer player work, start by finding all of the places where you mention the Blue player, and change them to be automated. That is, the computer should choose all of the moves for Blue.

For the first version, you’re welcome to use a random number generator to choose the moves for Blue. However, more advanced students could implement more complex algorithms.

In addition, you need to update your code - when the computer plays - to add a small pause. It doesn’t have to be long, but make it such that the computer will seem to “think” before it makes a move. This is partly to make the game more interesting - and it’s partly just so that the player can see their own move take place (and show on the screen for a second or two) before the computer responds.

If you have extra time at the end of Section, implement command-line arguments which will decide whether the Red player is human or computer - and the same for Blue. Design your program so that it can be played by two humans,
by a human versus a computer, or even by two computers playing against each other. For an extra-interesting version, add the ability to have two computers play each other many times in a row - and then report how many games were won by each player (and how many were times).

☑️ CHECKPOINT 3
Raise your hand. Your SL will come over and verify that you completed this step correctly.

Extra Tasks
If you finish all three checkpoints, please go back and finish the “extra” tasks I mentioned above.

The End: Clean Up
The G/S 930 lab is host to sections from several of our programming classes, and is available the rest of the time for any CS student to use to work on their programming assignments. To help you get in the habit of leaving your work space neat and clean, we’ll end each section activity with this checkpoint, which also serves as confirmation that you attended section - even if you get nothing else done, you can clean up before you go, thereby earning one checkpoint and proving you attended.

• Log out of your computer.
• Pick up your papers, writing implements, cell phones, etc.
• Push in your chairs.

☑️ CHECKPOINT 4
Raise your hand. Your SL will come over and verify that you completed this step correctly.

As with so much in this course, thanks to Dr. McCann for some of the ideas - and even some of the exact text - for this Section Activity!