Project 1: Connect Four

Due: 25 February 2013

Connect Four is a game where players take turns dropping tokens into columns of a board, and the goal is to place four of your own tokens in a row while preventing the other player from doing so. In this respect it is like a four-in-a-row version of tic-tac-toe, except that players don't choose a row, only a column; their token is automatically placed in the lowest available place on that column. For instance, given the following board:

0	1	2	3	4	5	6
		O				
X	X	X				O
O	X	O	X	O		O
X	O	X	O	X	X	O

The next play is X's, and if they choose column 3 or column 1 it is a win for X (column 3 completes a horizontal group of four, and column 1 completes a diagonal). On the other hand, if X blows their chance and plays in column 0, 2, 4, or 5, then player O has a chance to play column 6 for a win with a vertical group of four.

In this project, you'll implement Connect Four and an AI to play it.

Some format specifications

When you read or write a board, it should follow this format:

```
. . . . . . .
. . . . . . .
. . O . . . .
X X X . . . O
O X O X O . O
X O X O X X O
```

That is, each row is a line, columns are separated by spaces, and empty places are represented by '.' (period). The lines may or may not end in whitespace. When writing, you may print the column numbers above each column, but you shouldn't expect this when reading.

You can assume that player X always goes first.

When reading a board and the board is followed by numbers, those numbers represent moves (column numbers). The exact interpretation of those moves depends on what phase of the project you're in—they may represent both players' moves, or just one.

Checkpoint

For the checkpoint (next Monday), you should have a program that:

- reads board states,
- makes the correct play given a "move" (a column number),
- prints board states,
- and evaluates states for potential wins by either player in at *least* the two cardinal directions (horizontal, vertical).

Since there's no AI yet, an input of a blank board followed by "`2 5 0 2`" would yield the board state

```
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . O . . . .
X . X . . O .
```

and should also print a message to the effect that there's no win or draw condition yet.

Strongly recommended: when you implement a function to "make" a move, do so by constructing a new state and returning it, rather than modifying the existing current state. If you don't do it now, you'll just have to do it later.

Final version

The final version should, ideally, play a competitive game of Connect Four. Since I'm not going to be able to spend a ton of time playing many games against your AI, your documentation will need to tell me what your program does and convince me that it does what you claim (typically by providing test cases or other evidence). Credit is on the following scale:

0: Doesn't compile, or compiles but immediately crashes when it's run.

10: Compiles, runs, does no more than specified for the checkpoint.

25: Runs at least some games to completion without crashing, detecting wins in all four directions (including diagonals). The user alternates moves with the computer, which chooses some valid move and updates the state accordingly.

45: Runs games to completion without ever crashing or getting stuck in a loop, and has an AI that makes good choices at least in the cases when a win or loss is just a few moves away.

55: All of the above plus uses some sort of heuristic values to make good choices even when a win is not imminent.

60: All of the above plus the computer is capable of playing as either X or O based on command-line options.

65: All of the above plus some other good AI work. This is open-ended but possibilities include particularly good heuristic functions, alpha-beta pruning, or clever data structures that permit better caching or increased lookahead.

Again, to get the points you have to not just tell me but *show* me that you've done those things. Document your work and tell me what to look for!

Handing in

Hand it in as `proj1` using the handin script.