Grader Description:

For this lab, submit a script that, when given a board and a token on the command line, will return a preferred move using a simple, non-recursive strategy. Scoring is based on the sum of the total number of the script's tokens at the end of each game divided by the sum of all the tokens at the end of each game. Trying for a value upwards of 75%.

There are some specific time limitations. The submitted script is allowed .01 seconds per move. That is to say, .3 seconds are allowed per game. If it is fast on a particular move (and for this lab, it should be fast on all of them), then it may hoard the extra time and use it in a future move. This consideration, however, motivates the following architecture: the list of all possible moves should be printed at the start of the script. ONLY SUBSEQUENT to this should the script investigate which move is most beneficial. That means that Lab 3 can serve as a base for this code. Note that although the school's Othello server supports time hoarding, the school competition currently does not.

There is an additional consideration which precludes simply dumping Othello3 code into the Othello4 grader. Namely, the script for Othello4 must have a function defined, quickMove(brd, tkn) which takes brd as a lower case string and tkn as one of 'x' or 'o', and returns an integer in [0,63] that is a valid move. quickMove() need not be (but may be) called by any other function in the submitted script. Since the grader will import the submitted script, for the submitted script to work properly there should be a main() function (along with an if __name__ == '__main__': main() line at the end of the script). Global lookup tables should be set prior to this line (presumably via a setGlobals() function). quickMove should avoid printing as it will be called approximately 500*30 times per successful submission.

The submitted script will play 500 scripts against Random. Any error on its part (ie. script error, no move given, or illegal move given, and even a timeout) will cause the entire test to immediately terminate. If Labs 1, 2 or 3 are coming in at 100%, however, this should not be an issue.

The three lowest scoring games that the submitted code achieves will be shown in a condensed format. If you augment your argument parsing a bit, you can handle this without much problem. You can paste the sequence as a command line argument to Othello Lab 4 and it should display the game. There are a few negative numbers that may be observed in this sequence (negative numbers are informational only and the reason that your lab 3 code ignores them): -1 means there is a pass at that point. -2 means a timeout. -3 means the script asked for an illegal move (this usually happens because some other debugging information was being printed), -4 means the script output no moves (which means that the basic instructions were not followed), -5 means a script error. You should not see any numbers less than -1 on this lab and never less than -2.

Eric's Notes:

The grader description is pretty long, and can be shortened to a couple main points.

- There should be a main() function with __name__=='__main__'
- Any globals should be set before.
- Make a non-recursive function quickMove(board,token) that returns a good move for the player (token).
- Your code will be playing 500 games against random.

If you make quickMove non-recursive, there is no way that it should possibly exceed the time limit. (so don't worry about it)

This is where you actually put the I in AI!

The end goal is to have around 75% of total tokens, but with sufficient research, this can easily be low 80s.

First steps (from Eric, an othello sweat)

- Mobility is very important! This is the amount of moves that the players will have at a certain board state. Many people will use the difference of the players' moves over the total sum of moves. ((p1-p2)/(p1+p2+0.0001))

- Corners are good, and when you’re dealing with a non-recursive function, just take them when you can. Conceptually, corners are good because they can never be captured.
- X squares and C squares (search them up) usually aren’t good because they provide access to the corner. X squares are relatively much worse because they connect the center to the corner, so taking one almost always gives up the corresponding corner.Therefore, try not to play in these.
- Also, human experts (hilariously) minimize discs in the early game and midgame
- Search up “computer othello wikipedia”

If you do this well, again, you’ll complete the later labs easier.