Artificial Intelligence
Fall 2015
Project #1

As you are already aware, for your first AI project, you will implement a game-playing program that plays either Checkers (a.k.a. English Draughts) or Othello (a.k.a. Reversi) against the user. You may write the program using any programming language that you wish, as long as I can read, compile, and execute your code. We will discuss the rules of these games in class, but ***it is your responsibility to make sure that you understand the rules of your selected game correctly***. Google searches for "Othello rules" or "Checkers rules" will lead to pages that discuss the rules of these games. If you are unsure about a particular rule, ask me.

You do not need to implement a graphical user interface; an ASCII interface is fine, as long as you follow these guidelines:

- I should not have to strain my eyes when playing the game; part of this means that each square of the board should occupy multiple rows and columns (and still look like a square).
- Experiment to determine which characters are easily distinguishable; I have learned from experience, for example, that X and O are not the best choices.
- Use two different, clearly distinguishable colors for the two players; most platforms provide simple hacks for displaying text with color (if you can't figure it out, ask me or a classmate).
- Do not clear the screen or overwrite the board for each move; if you implement a text-based interface, I want to be able to scroll up and see previous positions (because it helps to recognize which move that the computer has just made).

If you wish, you may implement a GUI, but you will not receive any extra credit for this.

Your program must allow the user to select who moves first and *there must be an option to have the computer play against itself*. Furthermore, *your program must allow the user to specify a starting position other than the typical starting position for the game*; the reasons for this requirement will be discussed more in class. At the end of a game, the program should announce the winner (or that the game is a draw); in Othello, the final score should also be displayed.

On each of the user's turns, the program should prompt the user for his or her move, giving the user an intuitive, simple way to specify the move. The program should check to ensure that the move is legal, and prompt the user again otherwise. *The program must also provide the user an*

option to list all of the legal moves, or it can display a list of all the legal moves automatically (this will help me verify that your program is determining all of the legal moves correctly). If legal moves are displayed automatically, one possibility is to number them and have the user select a number as his or her move. The program should not crash or hang if the user responds incorrectly, even if they type something totally unexpected. (You do not have to protect against the user pressing Ctrl-C or anything like that.)

At the start of the game, *your program must also prompt the user for a time limit.* This time limit, expressed as an integral number of seconds, determines how long the program has to make each of its moves. *Your program must combine iterative deepening and an alpha-beta search* (i.e., a minimax search with alpha-beta pruning), *along with a heuristic function* (a.k.a. an *evaluation function*) of your own construction, to search the game space and decide each of its moves. *If multiple moves are tied for the best, the program should choose between them randomly*. You may include other potential improvements if you wish (e.g., singular extensions, transposition tables, etc.), but this is not required and I do not expect it. If you implement the search correctly, and you come up with a good heuristic, your program will play very well. Checkers programs should try to win quickly, rather than slowly, when possible, and Othello programs should try to maximize their wins; we will discuss how to accomplish this in class.

After every move by either the computer or the user, the program should display the updated board. After each of the computer's moves, *the program should also output the amount of time spent searching for the current move and the maximum depth that has been searched*. The time must be specified in terms of actual time (a.k.a. calendar time), not processor time (a.k.a. CPU time). You will lose points if any search takes longer than the specified time limit. In certain instances, the computer should not search for its entire time limit. If only one legal move is available, it should be taken immediately. Towards the end of the game, when the remainder of the game space can be searched in its entirety using less than the time limit, the computer should make its move after this has occurred. Optionally, you may decide that if more than half of the time limit is used up after a search to some specific depth, the program can use the result of the latest search and not bother starting the search to the next depth.

Artificial Intelligence
Fall 2015
Project #1

Assuming that your program works correctly (meaning that you have adhered to all of the requirements specified here, and there are no bugs), your program will be graded mostly based on how well it plays against me. I will play each program multiple times until I feel that I have a fair assessment of its level of play. I know from several previous semesters that a great Othello project should be able to beat me almost every game; I might squeak out a win once every five or ten games. A great Checkers project should rarely lose to me, but many games might end in a draw, and again I might squeak out a win once every five or ten games (I do not expect your project to reach the level of Chinook). A great Checkers project should also be also able to win every time if I give it a two king vs. one king advantage, even if my king starts off in a double corner (this will be discussed further in class).

Every now and then, a program may make a move that appears to be a bad move, but two, three, or four moves down the line, you will realize it was actually a good (or at least reasonable) move; I am aware of this. However, there are some moves that are clearly bad. When an Othello program gives up a corner for no reason in the middle of a game, or when a Checkers program makes a move that gives the opponent a double jump leading to a king and gets nothing in return, these are bad moves. Things like this do not happen due to some subtle flaw with the minimax algorithm or alpha-beta pruning; they indicate that one or more bugs are present. Sometimes, there are bugs that only occur in very specific situations that are hard to reproduce, and the bug itself can be subtle. I am fully aware that tracking down such bugs is difficult, but it is your responsibility to do so! (I will suggest some strategies for tracking down such bugs in class.)

You will have one month to complete this project, and I will propose a time schedule for you to follow in class. Upon completion of the assignment, you will e-mail me all of your code, instructions on how to compile and run your programs, and a brief (approximately one page) write-up describing your program's implementation, features, and whatever else you would like me to know. *I can not stress this enough*: ***Get started early!*** And make frequent backups. This is a tough assignment. That being said: have fun with it!