Artificial Intelligence – Fall 2020 - Suggested Schedule and Advice for Project #1

The project is due the night of Friday, November 6, by 11:59 PM. I suggest that you spend the first three days thinking about the project and choosing a game. That would leave four weeks to implement the project, and I suggest following the schedule below. Barring extreme circumstances, *there will be no extensions*. The late penalty will be 1.5 points per day rounded up (and that can add up fast).

Week #1:

- Implement your data structures and your legal moves function. (This is much harder for Checkers than for Othello.) Efficiency matters more than elegance here!
- Create and test a function to display the board.
- Allow the user to load a game state from a text file. The file should indicate the current board position and whose turn it is. Use this functionality to test the legal move function thoroughly.

Week #2:

- Implement a function to apply a move to a position.
- Create a nice ASCII interface (or a GUI) and enable your program to play a complete game:
    - Allow the user to select who makes the first move.
    - On each of the computer's moves, choose between all moves randomly (or choose the move that maximizes a simple heuristic).
    - On each of the players' move, display a numbered list of all the legal moves, and prompt the user to select one. Apply the selected move.
    - Display the board after every move and check to see if the game is over.
    - Provide an option for the computer to play against itself.

Week #3:

- Implement the alpha-beta search and iterative deepening (this is difficult for both projects):
    - If you implement a single recursive function, make sure you are handling the alpha and beta parameters correctly, especially if the perspective changes, as with negamax.
    - Make sure your program is handling the time limit correctly.
    - Provide yourself the option of producing an ASCII representation of the searched portion of the game space; this can be very useful for debugging.
- Implement a semi-simple heuristic and test your program thoroughly from various starting positions.

Week #4:

- Experiment with your heuristic function and discover one that makes your program play great! (In my opinion, this is harder for Othello than for Checkers.)
- Remember that the heuristic function should be efficient, and it must be zero-sum.
- If you organized your program well, you should be able to have the program play itself using various heuristics against each other to help evaluate which ones perform better than others (not a requirement).
- Reread the project requirements carefully and make sure you are not forgetting anything.

Feel free to read up on the game of your choice (in fact, I strongly advise it). Search for websites and applications that allow you to play against other programs or humans. For Othello, I have found that Gunnar Andersson's program called Zebra plays extremely well. For Checkers, the Android app called Checkers Free from AI Factory Limited plays well at its most advanced level (make sure to click yes for forced captures). For Checkers, you can also play against Chinook on-line! You can also try playing your program against other programs or humans (but in the latter case, it may be unethical if you do not let them know).