CSC 320: HW 5 CONNECT-FOUR A.I.

"Failure is simply the opportunity to begin again, this time more intelligently." -Henry Ford

"No sooner does man discover intelligence than he tries to involve it in his own stupidity."
- Jacques Yves Cousteau

" By playing games you can artificially speed up your learning curve to develop the right kind of thought processes." - Nate Silver

Note: This is an individual (not pair) assignment. You may discuss the assignment and help each other, but you should not be sharing any code with each other.

Background reading/materials

- Python: read about Classes in Python. The first half of this tutorial might be good.
  - http://python-textbok.readthedocs.io/en/1.0/Classes.html
- Minimax: review the slides from class. Google (or wikipedia?) if you need more detail.
- Alpha-beta pruning: review the slides from class (especially the pseudocode slide).

2. Goals

Download the starter code from Moodle. You should ONLY modify **connect4ai.py**, but you'll also want to read **connect4game.py** (which is the file to run/launch in Spyder).

(10 pts) Implement the minimax search algorithm by filling in code into the appropriate methods of the MiniMaxAIPlayer class inside the **connect4ai.py** module.

Notes: the code will be a little more complex than the pseudocode shown in the slides, because in addition to finding the value of each game state, you also need to remember which ***action*** was the best one to take from that game state (at least for the root node) so that the A.I. can take the appropriate action. This means that instead of just using the "max" or "min" functions to find the biggest value, you'll want to use an IF statement to check whether the value is better than the best so far and if it is, save that action by using the **gState.setBestAction(...)** method.

(5 pts) Design an appropriate heuristic function to evaluate non-terminal game state nodes. Fill in code for the *heuristicValueForPlayer* method of the **connect4ai.GameState** class.

(5 pts) Implement the **alpha-beta pruning** enhancement of minimax search by appropriately using the alpha and beta parameters in the functions.

(5 pts) Write up a brief (about 1 page) report describing your A.I. design choices (especially for the heuristic evaluation function), and discussing your A.I.'s performance. Include a graph showing how much long your A.I. "thinks" at different search depth limits (e.g. to calculate early-game moves) Other topics could include – how much faster does it run with alpha-beta pruning? How difficult of an opponent is it (at various search depth levels)? Can it search to the bottom of the tree?

By **Friday of Week 7**, submit (on **Moodle**)

- All Python source code files involved (even the ones you didn't modify)
- Your **README.PDF** file