4100/5100 Assignment 2: Othello Values
Due Thursday May 24 9PM

In this assignment, you'll program a minimax module for the board game Othello, also known as Reversi. The main module that you'll program and test in HackerRank will calculate the value for a given board position. But, code is also provided for you that will let you play against your AI if you like.

The rules of Othello are as follows:
a) The two player colors are white and black. The white player goes first.
b) You capture an opponent's pieces when they lie in a straight line between a piece you already had on the board and a piece you just played. (A straight line is left-right, up-down, or a 45 degree diagonal.)
c) You can only play a piece that would capture at least one piece. If you have no legal moves, the turn is passed.
d) The game is over when neither player has any legal moves left. Whoever controls the most pieces on the board at that point wins.

Something that is slightly unusual about Othello for minimax is the fact that **a turn might be skipped** if a player has no legal plays. You'll have to take that into account in your minimax calculations. (Don't have skipped turns count against the search depth.)

The AI is always presumed to be white for this assignment; if you try the demo mode, you as the human will be playing black.

The input will be the search depth on a single line, followed by an ASCII representation of the board (W for white, B for black, - for an empty space).

The HackerRank site is:

`https://www.hackerrank.com/contests/cs41005100-summer-1-hw2-othello-and-minimax/challenges`

1) Download the provided `OthelloSolver.java` code.
2) Implement basic depth-limited minimax for the `minimax_value` function, ignoring the alpha and beta arguments for now. Your evaluation function, when you bottom out, should just be the difference in piece count between white and black. You should be able to effectively use a depth of 5 or so without waiting too long.
Tip: While debugging, you'll find it useful for minimax to print the board state it is evaluating and the value that it assigns to that board. Also, debug small depths before attempting larger ones.
3) Submit a first pass to HackerRank. Try to pass all the tests where the search depth is 5 or less. You are expected to time out on the deeper search depths at this point, except the endgame test (the last test).
4) Implement alpha-beta pruning.

5) Submit to HackerRank, and repeat until all tests are passed.
6) Optional: You can now play against your AI using the command line argument “play”:
```
java OthelloSolver play
```
Set `DEMO_SEARCH_DEPTH` to whatever degree of lookahead you have the patience for.

Othello is actually a somewhat tricky game to craft an evaluation function for; here we adopt the philosophy of not overthinking it.