ECS 170: Spring 2021
Homework Assignment 3

Due Date:

No later than Sunday, May 16, 11:59pm PDT. Note that your next homework assignment may be posted before then. You are expected to do this assignment on your own, not with another person.

The assignment:

Hexapawn is played on a 3 x 3 chessboard. The two players face each other across the board. Each player begins with either three white pawns or three black pawns, placed one to a square in each of the three squares nearest that player. The player with the white pawns always moves first. A player may choose to move one of their pawns in one of these ways:

- A pawn may be moved one square forward to an empty square
- A pawn may be moved one square diagonally forward to a square occupied by an opponent's pawn. The opponent's pawn is then removed.

The players alternate moving pawns until one of the players wins. A player wins when:

- A player's pawn has advanced to the other end of the board.
- The opponent has no pawns remaining on the board.
- It is the opponent's turn to move a pawn but is unable to do so.

As envisioned by its inventor, whose name is now forgotten but whose ideas live on, hexapawn was intended to be played with only six pawns on a 3 x 3 board. Now, however, we are extending the definition of hexapawn to include any similar game involving n white pawns, n black pawns, and a n x n board. You are to construct a Python function (and all supporting functions) which takes as input a representation of the state of a hexapawn game (i.e., a board position), an integer representing the size of the board, an indication as to which player is to move next, and an integer representing the number of moves to look ahead. This function *returns* as output the best next move that the designated player can make from that given board position. The output should be represented as a hexapawn board position in the same format that is used for the input board position.

Your function must select the best next move by using MiniMax search. You will need to devise a static board evaluation function; feel free to use the one that was given in class or supply a better or more interesting one.

Your top-level function should be named "hexapawn". Your function must then be ready

to accept exactly four parameters when called. This sample function call explains what goes where in the argument list:

```
>>> hexapawn(["www","---","bbb"],3,'w',2)
                                ^           ^   ^   ^
                                |           |   |   |
The first argument is an n-element list,    |   |   |
and each of these elements represents       |   |   |
a row of the board.  The first element      |   |   |
is the first or "top" row, the second       |   |   |
element is the next row, and so on.         |   |   |
Each of these elements is itself an         |   |   |
n-character string.  The characters in      |   |   |
the strings are either 'w' to indicate      |   |   |
a white pawn on that square, 'b' to         |   |   |
indicate a black pawn, or '-' to indicate   |   |   |
an empty square. The leftmost character     |   |   |
in one of these strings represents the      |   |   |
leftmost square in the row represented by   |   |   |
that string, and so on.                     |   |   |
                                            |   |   |
The second argument is an integer to -------+   |   |
indicate the size of the board being            |   |
played on (or the number of pawns each          |   |
player begins with).  Thus, 3 here              |   |
indicates a 3 x 3 board.                        |   |
                                                |   |
The third argument will always be 'w' or 'b',-+ |   |
to indicate whether your function is playing   |   |
the side of the white pawns or the side of     |   |
the black pawns.  There will never be any      |   |
other color of pawns.                          |   |
                                               |   |
The fourth argument is an integer to indicate ---+
how many moves ahead your minimax search is to
look ahead.  In this example, the 2 indicates
that the function should look at white's move and
black's move in reply, but no further.  Your
function must not look ahead any further than
the number of moves given as this argument.
```

This function should then return the next best move, according to your search function and static board evaluator. So, in this case, the function might return:

```
["w-w","-w-","bbb"]
```

or some other board in this same format.

Some extra notes:

1) We may need to modify the specifications a bit here or there in case we forgot something. Try to be flexible. Don't complain.

2) Just like in assignment 2, we want to see lots of well-abstracted, well-named, cohesive, and readable functions. Comments are required at the beginning of every

function and must provide a brief description of what the function does, what arguments are expected by the function, and what is returned by the function. Your comments should make it clear where the board evaluation is done, where the move generation is done, and where the minimaxing happens. Additional comments that help the TA who grades your program understand how your program works may make the TA happy, and happy graders are generous graders.

3) A static board evaluation function is exactly that -- static. It doesn't search ahead. Ever.

4) You can convert our board representation to anything you want, just as long as when we talk to your top-level hexapawn function or it talks to us, that function is using our representation.

4) We are not asking you to write the human-computer interface so that your program can play against a human. You can if you want to, just for fun, but we don't need to see it.

5) Program early and often. A simple board evaluator is easy. The move generator is harder. The minimax mechanism is harder still.

6) Your top-level function should be the first function defined in the file you send us. Don't make us hunt for it.