HW 2: Connect Four

Your job is to create an AI to play “Connect Four”. It will use the minimax algorithm.

If you are using Java, download the files ConnectFour.java, ComputerConnectFourPlayer.java, HumanConnectFourPlayer.java, GraphicsWindow.java, and MouseHandler.java. The ConnectFour class is the main one. It takes one command line argument: the number of moves that the computer player should look ahead. (If there is no argument, or the argument isn’t a number, it simply uses 1. If you compile everything and run it, you should have a functional game of Connect Four (with a very stupid computer opponent).

If you are not using Java, you are responsible for making your own infrastructure. However, the game doesn’t need to be graphical: you can simply display the current board with text, and ask the user to enter a column in which to play (1 through 7).

You are responsible for modifying the class ComputerConnectFourPlayer. At present it simply plays as far to the left, every turn. It will look ahead the number of rows it’s been assigned, and return the best move at each turn. The only public method it needs (aside from the constructor) is getNextPlay(). This takes a 2D array of bytes, representing the current board. This is column-major, with element [0][0] indicating the top, left space in the rack. Given the rack, it must return an int 0 through 6, with 0 meaning that it will drop a token in the left column.

The evaluation function that you will use is as follows. Look at every location in the rack in which four spaces are in a row, where a player could possibly win. (The standard rack has 24 horizontal locations, 21 vertical ones, and 24 diagonal ones.) If the location has no tokens or at least one token of each color, it is worth 0 points. If it has four tokens of the AI’s color, it is worth infinite points. If it has three of the AI’s color and no opposing tokens, it is worth 100 points. Two of the AI’s color with no opposing gets 10 points, and one gets 1 point. If the tokens are of the opposing color, the score magnitudes are the same, but the sign is reversed. (So two tokens of the opposing color but none of the AI’s color is worth -10 points.) Sum the point values of all 73 four-in-a-row locations, and that is the current score.

You do not need to implement alpha-beta pruning. However, if you do, you will get a modest amount of extra credit (up to 15 points). If you choose to do this, make sure that you can turn it on and off easily, so that you can check your results. Remember, proper alpha-beta pruning will never change a move, though it will be calculated more quickly!

Some hints:
- If you need a refresher on how the game works, go to https://en.wikipedia.org/wiki/Connect_Four.
- Be sure to bug test extensively, using different levels. This code can get twisted, fast.
- It may be easier to implement negamax than straight minimax. Mathematically, the two algorithms are identical.