CS4750/7750 HW #4 (20 points)

Implement the minimax algorithm to play a two-player, four-in-a-row game, which is a variation of tic-tac-toe: two players, *X* and *O*, take turns marking the spaces in a 6×5 grid. The player who succeeds in placing 4 of their marks consecutively in a horizontal, vertical, or diagonal row wins the game. See an example below where 'x' plays first and wins the game.

	o	x		x
	o	o	x	
o	x	x	o	
	x			

You may use any programming language in your implementation.

You are asked to do the following tasks:

1) (8 points) Implement the minimax algorithm to play the game. Describe your implementation in your report.
2) (4 points) Implement Player 1 as follows.
   As the player making the first move, put the first mark at the center of the board. Then, at every turn, run the minimax algorithm on a 2-ply game tree, i.e., looking ahead 2 moves (one move by the player and one move by the opponent) and apply the following heuristic evaluation function on the resulting board, if it is not a terminal state. Breaking ties randomly. For terminal nodes, return their utility value: -1, 0, or 1.

h(n) = 100[number of two-side-open-3-in-a-row for me]*
– 10[number of two-side-open-3-in-a-row for opponent]*
+ 100[number of one-side-open-3-in-a-row for me]*
– 5[number of one-side-open-3-in-a-row for opponent]*
+ 2[number of two-side-open-2-in-a-row for me]*
– 2[number of two-side-open-2-in-a-row for opponent]*
+ [number of one-side-open-2-in-a-row for me]
– [number of one-side-open-2-in-a-row for opponent]

where

- "one-side-open-3-in-a-row": there is an empty space next to one end of a 3-in-a-row to potentially make it 4-in-a row in the next move.
- "two-side-open-3-in-a-row": there are empty spaces next to both ends of a 3-in-a-row to potentially make it 4-in-a row in the next move.
- "one-side-open-2-in-a-row": there is an empty space next to one end of a 2-in-a-row to potentially make it 3-in-a row in the next move.
- "two-side-open-2-in-a-row": there are empty spaces next to both ends of a 2-in-a-row to potentially make it 3-in-a row in the next move.

For example, for player ‘x’, the value of the following state is

*h = 100*0 - 10*1 + 100*1 - 5*0 + 2*1 - 2*0 + 0 - 2 = 85*

	o	x		
	o	o	x	
o	x	x	o	
	x			

3) (4 points) Implement Player 2 in a way similar to Player 1 and use the same heuristic function, but instead, run the minimax algorithm on a 4-ply game tree, i.e., looking ahead 4 moves (two moves by the player and two moves by the opponent).

4) (4 points) Play a game between Player 1 and Player 2, with Player 1 making the first move. Print out every move made by the two players in the whole game and the corresponding number of nodes generated by the search algorithm for calculating each move.

Submission:

a) A report of your implementation and result.
b) A zip file containing your code with appropriate comments.

You may form a team of up to three persons. Your team may be different from that for HW #2.