6:
The two search depths I used were 2 and 5, on a board of size 7 and a winning length of 4. To demonstrate the impact of the depth, I decided to try to beat the algorithm.

In my game the AI starts with an advantage playing first and in the center. It didn't take many moves before I was playing "offense" and it was playing "defense." I ended up winning in 16 moves when the search depth was 2. This surprised me because it could have blocked my move, which makes me think the evaluation function might not have been the best for smaller depths.

Playing with a depth of 5 was much more difficult. It was able to beat me the first two times I played it, but eventually I found it's flaw. Because of the evaluation function, if it has a stream of *n* pieces that is blocked off and thus useless, it doesn't know or care and is afterwards somewhat blinded. Using this knowledge, and my knowledge of what moves it takes after playing it, I was able to beat it.

6: Using the better eval function

I repeated the experiment with the evaluation function I describe in part 7 with depths of 2 and 4.

With a depth of 2, it beat me after 13 moves. This surprised me because it cannot see that far ahead, and at the place it beat me, there were a number of different places it could have won, or been close to winning from.

Playing it at a depth of 4 it beat me. I played it multiple times and was not able to beat it. It beat me as fast as 8 moves (I should have seen its finishing move coming), and in around 11 when I was paying moderate attention.

7:
My initial evaluation function simply found the longest consecutive streams of each player, and subtracted them from each other. This gives the AI an incentive to have a longer stream of placements than me. However, as for an actual evaluation function, this does not work too well because if they have a stream of 4 that is blocked on both ends, they don't know the difference and think they are always equally close to winning, and thus will make bad moves.

My second, and much better evaluation function, took a different approach. The second evaluation function starts by getting all streams of pieces, but only if they *could* potentially be apart of a winning stretch of pieces. Ex: XOOOXX would give two to X since there is a stream of two X's, but not three to O since that stream is blocked at both ends. From there if one of the players is one move away from winning, the evaluation function returns a very large/very small number on their behalf. Otherwise, it squares all the numbers (so having bigger streams is better than lots of small ones), sums the results, and subtracts them from each other.

The effect of the evaluation function seems to be enormous. With my initial evaluation function, I was able to beat it even when I was making some bad moves and it was checking multiple moves ahead. With the improved evaluation function, even with relatively little depth, it can beat me relatively easily.