6. The increasing depth of the function effects the run time greatly as the number of iterations goes up. Running at a depth of 5 it takes **24ms** to do, depth 10 takes **402ms** and depth 100 takes **>10min.** This obviously has an extreme case, running them for about 3 time each those are the averages. This is just for one move after the user has made their first move. This could make a game much longer than needed. The higher the iteration level, the more nodes it has to go through for each move. This can be tedious as the iteration number increasing. Upon further examination the higher the iterations are the “longer” the game will play. The AI will play the “long game” when you play it with high iterations. The more iterations that there were, the more memory that it took up to do a single move. I cannot begin to comprehend doing an entire game with 100 iterations.

7.

My eval functions eval1 and eval2 where eval 1 is a basic evaluation function where it takes the kinds of pieces and gave each one a value and subtracted the black while adding the white pieces. The values I gave were based on how many directions the piece could go, 1 for man pieces that could only go forward and 2 for kings that can go forward and backward. This was a very rudimentary eval function but it gave me something to improve upon. The way I though of improving upon it with adding how far the pieces are to becoming kings. This means the AI would have a secondary goal to get the pieces to be kings instead of just letting them die. I couldn’t think of many other ways to improve it. I was going to add the number of pieces to each side but figured that was pointless as the side with more pieces would obviously have a bigger advantage in the score. After copying the methods I used in my code, I was able to make it to verse 2 ai against each other with the depth of 5 to take less time and ran it 10 times with the eval2 function winning 7 out of 10. I believe that I could have made it even better.