Minimax:

The minimax function uses alpha-beta pruning with a max depth of 10 to try and find the best possible move. The function checks to see if there is a winning move using the find result function. Checks the depth of the algorithm and then calculates the score of the best move. The way it’s been implemented is to calculate the scores of all possible moves and then return the highest score. If the scores are equal, it returns the left most column. The get move function then creates the location for the column the AI is going to move to.

Evaluation:

The evaluation runs through the board adding to the total score of a given move.

Appropriate Values:

The calculate score function calculates the score of the evaluation itself with the heuristic given in the assignment specs. For example, returning multiples of 1, 10, 100 and 1000 where the highest score is the best move.

Interesting:

I tried setting up the engine where it’d read the moves and mimic them on a dummy board created internally. Unfortunately, due to my bad time management, I couldn’t make the “v” value in bestmove z v the same as the evaluation when testing it for the dummy board.

Faults:

The first 3 moves from both sides are hard coded in because I couldn’t figure out how to make the first 6 moves be those moves as I am fairly certain they are the best moves after the first 6 tokens are dropped. I believe this is because the minimax function doesn’t work the way I wanted it to or I have missed something key for the algorithm to work as intended.