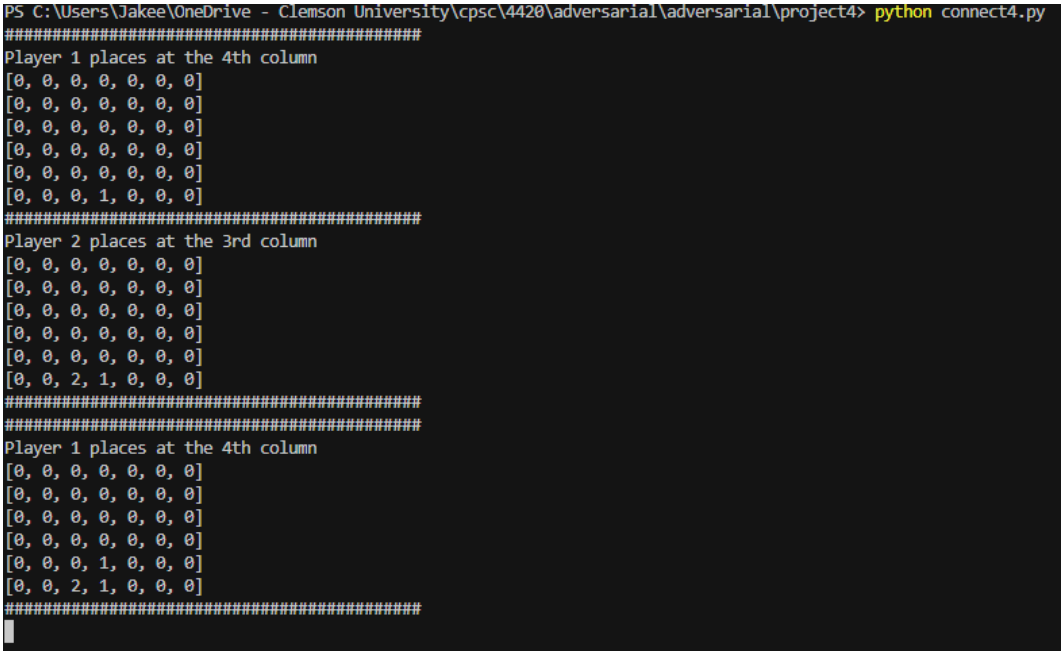
Jake Lunski
11/26/2024
Assignment 4 Report

Question 1
For this part, I implemented the recursive value function that stops searching when either the game ends (win, loss, or draw) or the depth limit is reached. For the functions max_value and min_value, they handle the turns of the maximizing player and their opponent, respectively. For the maximizing player, it calculated the maximum score among all possible moves, for the minimizing player it calculated the minimum score among all possible moves, assuming the opponent plays optimally. It goes through all valid moves and selects the column that resulted in the highest score for the maximizing player




Question 2
For Alpha-Beta Pruning, it optimizes the Minimax algorithm so it reduces the number of nodes evaluated by pruning branches that cannot affect the final decision. I added alpha and beta parameters to the value, max_value, and min_value functions. These represent the best scores the maximizing and minimizing players can have so far. In the max_value function, I stopped evaluating moves if a score greater than or equal to beta was found. In the min_value function, I pruned branches where the score was less than or equal to alpha.

```
PS C:\Users\Jakee\OneDrive - Clemson University\cpsc\4420\adversarial\adversarial\project4> python connect4.py
########################################
Player 1 places at the 4th column
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0]
########################################
Player 2 places at the 1st column
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[2, 0, 0, 1, 0, 0, 0]
########################################
########################################
Player 1 places at the 4th column
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0]
[2, 0, 0, 1, 0, 0, 0]
########################################
```

Question 3

The Expectimax algorithm  situations where the opponent makes probable random moves rather than optimal ones. Replacing the min_value function with expected_value calculates the expected utility of all possible moves. For each possible move, I assigned an equal probability and calculated the average score across all moves. I calculated the probability of each move as 1 / len(child_boards) to ensure the algorithm considered all moves fairly.

```
PS C:\Users\Jakee\OneDrive - Clemson University\cpsc\4420\adversarial\adversarial\project4> python connect4.py
########################################
Player 1 places at the 4th column
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0]
########################################
########################################
Player 2 places at the 6th column
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 2, 0]
########################################
########################################
Player 1 places at the 4th column
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 1, 0, 2, 0]
########################################
########################################
```