Introduction:

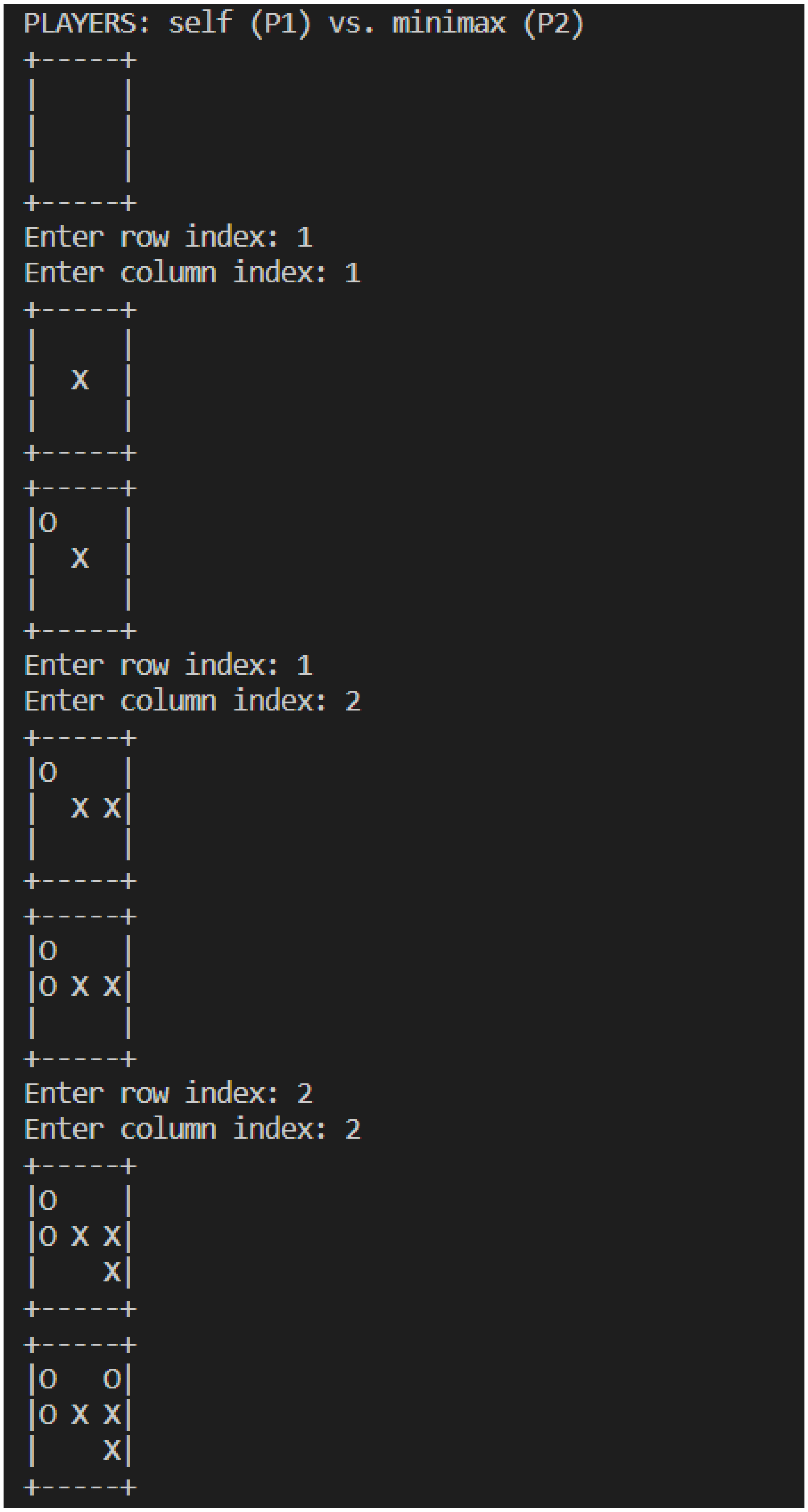
For the final project for this course, I decided to do the 1st option, which was to use an adversarial search algorithm to make an intelligent AI play Tic-Tac-Toe or Connect 4. To solve this problem, I decided to implement both a Minimax and Alpha-Beta algorithm in order to create an agent that could play either Tic-Tac-Toe or Connect 4 and also allow us to evaluate the differences between the two adversarial search algorithms.

Methods:

In order to solve this problem, I decided to implement Minimax and Alpha-Beta search algorithms. I decided to use those choices because I knew that the problem was involved between two agents, so I had to use an adversarial search algorithm. I also felt that Minimax and Alpha-Beta would be a good test of my coding skills since they were challenging, but not impossible for me. I would then test my systems by playing against each search algorithm in Tic-Tac-Toe and putting each search algorithm up against each other to see how they performed and if they seemed to be playing optimally.

Results:

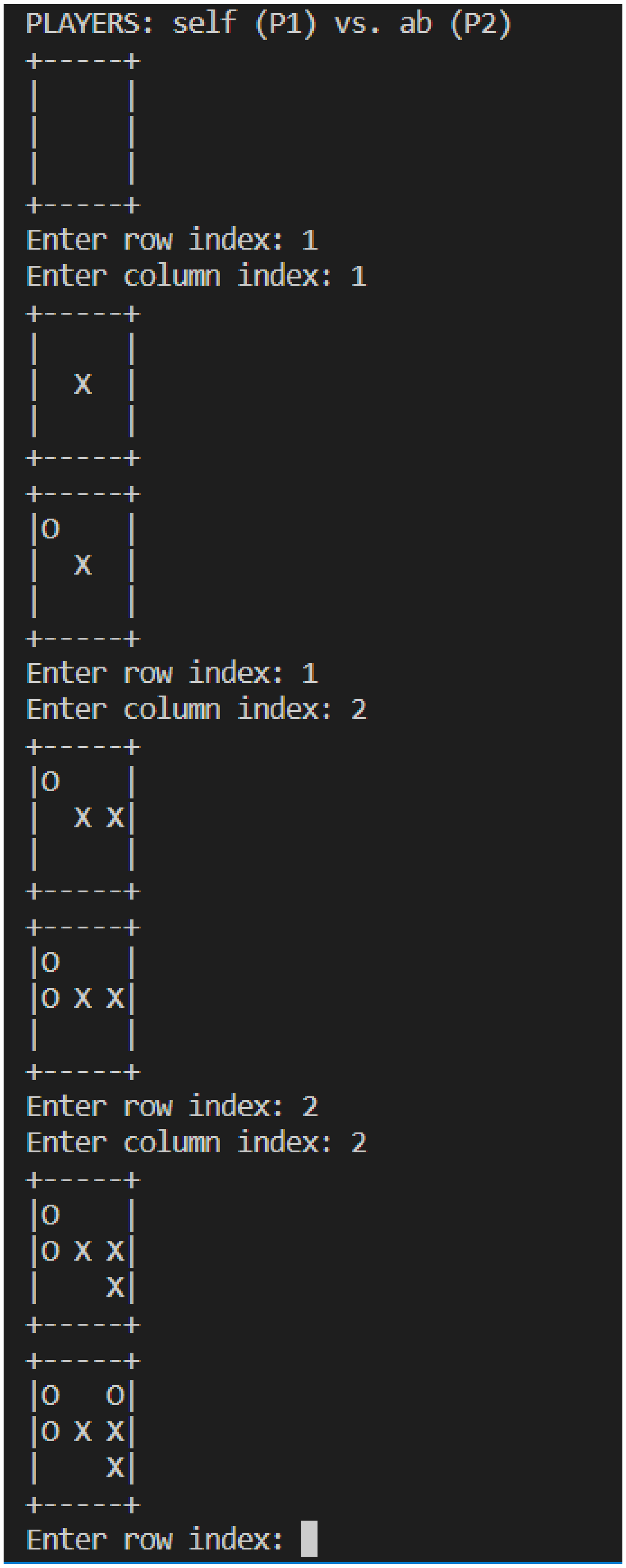
After writing the code for both the Minimax and Alpha-Beta search algorithms, I decided to test the algorithms in multiple ways. First, I tried to play against Minimax and Alpha-Beta, then I tried pairing the two against each other and seeing how they fared. For the latter test, I also made sure to switch the agents from Player 1 to Player 2 and vice versa in order to reduce any possible error. While both agents were able to play Tic-Tac-Toe on a 3x3 board, the agents weren't able to play on any higher dimensions due to the large number of states that had to be evaluated. This also meant that I couldn't test my agents on Connect 4 since the dimensions had to be greater than a 3x3 space. In my first test, I played Minimax 10 times, and drew each game. I repeated the same test with the same number of attempts, but instead played suboptimally to give Minimax opportunities to win. While Minimax did find the immediate win in 8 out of the 10 states, it would miss the immediate win in the other 2 states and get a draw. One of the two states where Minimax missed the immediate win is shown below.



```
PLAYERS: self (P1) vs. minimax (P2)
+-----+
|     |
|     |
|     |
+-----+
Enter row index: 1
Enter column index: 1
+-----+
|     |
|  X  |
|     |
+-----+
+-----+
|O    |
|  X  |
|     |
+-----+
Enter row index: 1
Enter column index: 2
+-----+
|O    |
|  X X|
|     |
+-----+
+-----+
|O    |
|O X X|
|     |
+-----+
Enter row index: 2
Enter column index: 2
+-----+
|O    |
|O X X|
|    X|
+-----+
+-----+
|O   O|
|O X X|
|    X|
+-----+
```

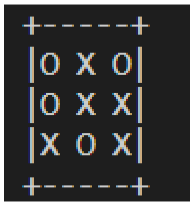
Next I decided to test Alpha-Beta. After playing Alpha-Beta to the best of my ability 10 times, I managed to draw every single game. Then, I played against Alpha-Beta and made more sub-optimal moves. Just like with Minimax, Alpha-Beta managed to find the immediate win in 8 of the 10 games and missed the immediate win in the other 2 games to get a draw. Just like in the picture above, Minimax and Alpha-Beta both missed the immediate win in this state.

Sadly, I’m not able to explain why Minimax and Alpha-Beta both do this in certain states.



```
PLAYERS: self (P1) vs. ab (P2)
+-----+
|     |
|     |
|     |
+-----+
Enter row index: 1
Enter column index: 1
+-----+
|     |
|  X  |
|     |
+-----+
+-----+
|O    |
|  X  |
|     |
+-----+
Enter row index: 1
Enter column index: 2
+-----+
|O    |
|  X X|
|     |
+-----+
+-----+
|O    |
|O X X|
|     |
+-----+
Enter row index: 2
Enter column index: 2
+-----+
|O    |
|O X X|
|    X|
+-----+
+-----+
|O   O|
|O X X|
|    X|
+-----+
Enter row index: 
```

Finally, I played Minimax and Alpha-Beta against each other 10 times, with each agent being Player 1 for 5 of the matches. In each game, the same moves were played, the same state was reached every time (draw), and neither agent missed any easy wins. The reason both agents draw every time is because Tic-Tac-Toe on a 3x3 board doesn't have a lot of states as compared to boards of higher dimensions, which makes evaluating the optimal moves much easier. I have a picture below of one game between Minimax and Alpha-Beta (Remember that Alpha-Beta is a faster version of Minimax.)



```
+-----+
|O X O|
|O X X|
|X O X|
+-----+
```

Since Connect 4 had to have at least dimensions of 4, both Minimax and Alpha-Beta weren't able to play Connect 4 at all.

Discussion:

After analyzing my results, I believe that my methods and choices that I used to complete this task were appropriate because this was an adversarial search problem, so I had to use some sort of adversarial search like Minimax or Alpha-Beta. Looking at my results, I think that I implemented both of my algorithms appropriately since they worked well and seemed to play optimally most of the time. If I had more time, I'd most likely try to figure out why in some states, the agent wouldn't always find the optimal move. I could then fix this issue to make sure that the agent would always play the optimal move in any case. Because the agents are simply fit for the purpose of playing Tic-Tac-Toe and Connect 4, I don't think there are any ethical implications of my system that others should know.