Option 3: Build an agent using advanced search techniques (for example: killer heuristic, principle variation search (not in lecture), or monte carlo tree search (not in lecture))

- Create a performance baseline using `run_search.py` to evaluate the effectiveness of a baseline agent (e.g., an agent using your minimax or alpha-beta search code from the classroom)

- Use `run_search.py` to evaluate the effectiveness of your agent using your own custom search techniques

- You must decide whether to test with or without "fair" matches enabled--justify your choice in your report

Run 20 games using Fair matches

MCTS - Win 35%

MINIMAX- Win 65%

Run 20 games without Fair matches

MCTS - Win 35%

MINIMAX- Win 65%

As the win percentage is the same for both runs (i.e. with/without fair matches), it is logical to test without fair matches as it will achieve the same result but in half the time.

Hints:
- If the results are very close, try increasing the number of matches (e.g., >100) to increase your confidence in the results

Run 1000 games using Fair matches (150ms)

MCTS - Win 34.5%

MINIMAX- Win 65.5%

- Experiment with adding more search time--does adding time confer any advantage to your agent?

Run 1000 games using Fair matches (1500ms)

MCTS - Win 35.5%

MINIMAX - Win 64.5%

Adding more time does confer advantage to the MCTS agent. This is likely that the agent is able to find more optimal policies with the added time.

- Choose a baseline search algorithm for comparison (for example, alpha-beta search with iterative deepening, etc.). How much performance difference does your agent show compared to the baseline?

The MINIMAX agent conistently outperforms the MCTS agent, based on the evaluations above.

- Why do you think the technique you chose was more (or less) effective than the baseline?

I believe the MCTS was less effective because the agent behaved closely to the random search due to the high branching factor and few iterations. The MINIMAX agent was more effective as it could find more optimal polcies in fewer iterations.