**CS-171 Checkers Final AI Report**

**Team name:** justDoIt

**Member #1 (name and NetID):** Carmen Cheung (carmenc4)

**Member #2 (name and NetID or N/A):** Wooyul Chung ( )

**I. In about 1/2 page of text, describe what you did to make your Final AI agent “smart.”**

To make our Final AI “smart,” we used a search algorithm that looked five moves ahead. This depth gave our AI a good balance of time consumption and “smarts.” To speed the search up, we implemented alpha-beta pruning, stopping our AI from exploring moves that are worse than another, previously observed move. Since move ordering drastically improves the performance of alpha-beta pruning, we added a very simple move ordering function that searches the best move from a one-step look-ahead first. This establishes a “good move,” and causes more pruning and, as a result, a speedier search. Our evaluation function calculated the difference between the number of pieces our AI has versus the number of pieces the opponent has. To steer our AI to victory, winning states have a very large score, while losing states have a very low score. As kings are very valuable in a checkers game, kings are counted as two pieces in the evaluation. This prioritizes moves that give our AI kings while denying the opponent kings.

**II. In about 1/4 page of text, describe problems you encountered and how you solved them.**

Very often, near the end of a game, our AI would make the same two moves over and over again, which would cause the game to result in a tie due to timing out. This was because our AI, during its search, would only replace the best move if a move had a greater evaluation score than the current best move. It would not replace the best move if the move had an equal score to the current best move, which was the reason why the AI kept thinking those two moves were the best move. To solve this, we added randomness to our AI if a move’s evaluation score tied with the current best move. If a move’s score was equal to the current best score, the AI would, in a sense, flip a coin to determine which move to select as the best move.

**III. In about 1/4 page of text, provide suggestions for improving this project.**

The pacing of the project was reasonable; we had plenty of time to piece together what to do and how to improve our AI. However, in the beginning, it was difficult to determine what each part of the code did. Thorough reading of the code was needed to have an idea of what each function does. Although this can improve our code comprehension, it’d be better if more comments were in the base code to give people who are new to the code a sense of what each code segment does. Another suggestion is to clarify how to get everything working on openlab. Although students may use different softwares to edit and run the code, the majority of them are using JetBrains’ tools. If there were more instructions on how those tools worked, the setup process would be quickened and we would have more time to work on the AI.