CONNECT-K FINAL REPORT ***[TEMPLATE --- do not exceed two pages total]***

Partner Names and ID Numbers: Hongkun Chen (62214504)

Team Name: ChenSohn

Note: this assumes you used minimax search; if your submission uses something else (MCTS, etc.), please still answer these questions for the earlier versions of your code that did do minimax, and additionally see Q6.

1. Describe your heuristic evaluation function, Eval(S). This is where the most “smarts” comes into your AI, so describe this function in more detail than other sections. Did you use a weighted sum of board features? If so, what features? How did you set the weights? Did you simply write a block of code to make a good guess? If so, what did it do? Did you try other heuristics, and how did you decide which to use? Please use a half a page of text or more for your answer to this question.

The evaluation function in our AI contains three parts. The first part will evaluate the broad horizontally.

2. Describe how you implemented Alpha-Beta pruning. Please evaluate & discuss how much it helped you, if any; you should be able to turn it off easily (e.g., by commenting out the shortcut returns when alpha >= beta in your recursion functions).

We write a new wrap function called alpha\_beta\_pruning which will call min\_search and max\_search functions. In both min\_search and max\_search functions, we added two more parameters called alpha and beta. These two parameters will be passed to their childen to do the pruning and the children will return their node value to help them update the value of alpha and beta. Furthermore, we compare the value of alpha and beta after each updated best value in min\_search and max\_search. If beta is less than alpha, the function will just return current node value and will not explore any of its remaining children.

3. Describe how you implemented Iterative Deepening Search (IDS) and time management. Were there any surprises, difficulties, or innovative ideas?

4. Describe how you selected the order of children during IDS. Did you remember the values associated with each node in the game tree at the previous IDS depth limit, then sort the children at each node of the current iteration so that the best values for each player are (usually) found first? Did you only remember the best move from a given board? Describe the data structure you used. Did it help?

In our AI, we just simply remember the best move from previous iteration. We use another global variable called final\_best\_move to remember the best move from previous iteration. After the 2 iteration, when our AI expands root nodes, it will expand the final\_best\_move first. We use a queue to implement frontier. So we just push last best move into the queue first to make sure it will be expanded first. IDS do increase the depth of search, but it does not increase much, especially in the beginning of the game.

5. [Optional] Did you try variable depth searches? If so, describe your quiescence test, Quiescence(S). Did it help?

6. [Optional] If you implemented an alternative strategy search method, such as MCTS, please describe what you did, how you implemented it, and how you decided whether to use it or your minimax implementation in the final submission.