Partner Names and ID Numbers: Anthony Manis (714 614 39) & Mukesh Kastala (406 463 51)
Team Name: Group

1. Our evaluation function uses a static heuristic based on the number of ways each player can win for any given game state. The end result is an integer representing the player's advantage or disadvantage as a positive or negative number respectively (zero represents an equal position). It evaluates game states in this way: First it checks to see if the state is an end state, and it returns special values for win, draw, or lose. If the state is not an end state it proceeds to count the "ways to win" for each player. The "ways to win" function sweeps the board horizontally, vertically, diagonally up, and diagonally down with a window of size k. If at any point this window contains pieces from only one of the players, it counts that as a possible opportunity to win (for that player). If the window ever has pieces from both players it disregards that opportunity since symmetrically, both players have blocked each other in that window, since the window is size k which is the number of pieces in a row required to win. Windows may overlap, so for instance a column of length 8 with k=5 has 4 vertical windows in it. Windows are always contained on the board, they never hang off the edge, or vary in size. An opportunity to win has a degree, determined by the number of pieces that player already has in the window. For instance a player may have 1 way to win of degree 3, but 6 ways to win of degree 2. The degree does not take into consideration how many pieces are actually in a row, only the number of pieces in that window. This makes the heuristic eliminates blind spots and "traps", where a player uses a single piece to connect two previously separate runs into a win. A window with different colored pieces in it is not considered a way to win for either player, because neither player can fit k pieces in that window any longer. For each player, a tally is kept for each degree of opportunities to be used by the calling heuristic function. Once the ways to win are counted for both players, the heuristic calculates a degree-weighted sum and then takes the difference of the two players' sums to determine which player has the advantage. The weighting factor is 10^(i-1) for 1 <= i < k. Doing this puts more emphasis on ways to win where a player already has many pieces in a particular window. This heuristic also encourages "blocking" opponents, because if an opponent reaches a way to win of k-1 or k-2, that term will dominate the sum and make the game state less valuable than any of the others.

2. Our alpha-beta pruning strategy passes alpha and beta as parameters during recursion, and causes a return whenever alpha >= beta. This reduces the required number of heuristic evaluations in some cases. In practice, this allowed us to set the search depth cutoff limit to 1 ply deeper than without alpha-beta pruning. However, when combined with IDS this improved our search much more significantly.

3. For IDS we used a map to remember the best move for each state from previous iterations. When a state is being examined in future iterations, the best move for that state is discovered in the map and put on the front of a queue, followed by the other possible moves. In practice this means that states satisfy the alpha >= beta requirement for pruning much faster in subsequent iterations. This allows us to start our search at ply 1 and increment by 1 each time until the time deadline. We modified our search function to throw an exception when the deadline has been

reached, and thanks to the grace period we are then able to catch that exception and return the result from the previously completed search before a timeout.

4. We only remember the best move for each node. This was sufficient to trigger pruning and increase our depth achieved during IDS. In the early turns, we see a single digit depth cutoff achieved. However, towards the middle of the game this depth increases quickly. As mentioned before we used a map from BoardModel to Point to keep track of each state's best move, and a queue of moves to determine the move order for each state.

5. We didn't use a quiescence test.

6. We didn't use an alternative search strategy.