For our minimax algorithm, we recursively call our minimax function alternating between returning the maximum and minimum of our possible actions. The maximum and minimum scores are found by determining the utility of a state when a terminal state is reached. When maximizing and minimizing, we also make sure to check for the event where there are no possible actions to take, in which case, we skip the current turn.

For our alpha-beta pruning algorithm, we use the same minimax algorithm, but keep track of the global variables alpha and beta. Alpha keeps track of the global max while beta tracks the global min. Keeping track of the global maximum and minimum helps us prune search in order to not search down a series of actions that we know will not result in results better than results that have already been seen. In addition to the alpha-beta pruning, we also use a transposition table. The transposition table uses a cache in order to save states that we have found the optimal path for already. When we run into a state that has already been seen before, we can obtain the correct action from the cache and avoid recalculating the path.

For our evaluation function, we use the given evaluation function in order to calculate the value of each action and choose the action which yields the best results.

For our custom agent, we combine the various methods used in the previous problems. Each move, we use iterative deepening to find a result as soon as possible. At the end of each iteration of iterative deepening, we check if there was any better choice. If we find a winning path, then we immediately break from the search and use that move. Otherwise we continue searching deeper until we search the full tree. In order to search, we use the alpha-beta pruning with a transposition table algorithm from problem 2. In order to search our tree more efficiently, we use the evaluation function from problem 3 in order to sort the possible actions each turn so that we check the action with the highest yield first while maximizing and the action with the lowest yield first while minimizing.