Team Information:

FirstName	LastName	Student ID
Basavaraj	Kaladagi	110562162
Nishanth	Muruganandam	110276247

Report:

Player1	Player2	Eval Fn	Winner	Execution Time	Nodes Expanded
BasicPlayer	AlphaBeta	Basic Eval	Player2	1: ~15secs 2: ~6 secs	1: 3601 2: 2148
NewPlayer	AlphaBeta	New eval	Player1	1: ~9secs 2: 1.2secs	1: 2883 2: 787

Alpha-Beta Logic:

```
def alpha_beta_search_helper(board,depth,level,eval_fn,get_next_moves_fn,is_terminal_fn,alpha,beta):

        """
        Actual implementation of Alpha-Beta pruning search. It returns the score of
        the move to the alpha_beta_search(..) which acts as a
        wrapper which returns the desired index to the actual caller.
        It inverts the score accordingly to maintain the consistency of the search
        algorithm.
        Pruning is done if alpha greater than or equal to beta.
        """
```

Minimax Logic:

```
def minimaxHelper(board, depth, level,eval_fn,
            get_next_moves_fn,
            is_terminal_fn):

        """
        Actual implementaion of Minimax algorithm. It returns the score of the move
        only. minimax(..) acts as a wrapper around it which gives the desired index
        to the actual caller.
        """
```

NewEvaluate Function:

def new_evaluate(board):

```
        """
        Idea behind:
                If the board has been won, then the previous player is the
                winner.
                So, we return -1000.
                Otherwise,score is a function of longest chain of both current
                player and the other player with more weight to our player.
                After empirical analysis, a weight of 3 and 2 for current and
                other player, seems to give better results.
        """
```