```
                    **************************
                         Playing Matches
                    **************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       7  |   3    10  |   0     7  |   3     8  |   2
    2       MM_Open      8  |   2     6  |   4     6  |   4     5  |   5
    3      MM_Center     8  |   2     8  |   2     9  |   1    10  |   0
    4     MM_Improved    8  |   2     7  |   3     7  |   3     5  |   5
    5       AB_Open      4  |   6     6  |   4     5  |   5     4  |   6
    6      AB_Center     8  |   2     6  |   4     5  |   5     7  |   3
    7     AB_Improved    6  |   4     5  |   5     4  |   6     4  |   6
-----------------------------------------------------------------------------
           Win Rate:      70.0%        68.6%        61.4%        61.4%
```

In the matching table above, AB_Improved is using (own_moves – opp_moves) to evaluate score of the current game state. AB_Custom is using (own_moves – 2*opp_moves) while AB_Custom_2 is using (my_moves), and AB_Cusom_3 is evaluating the square of the distance between the center of the bottom of the board and the position of the player. Below is a detailed view of the comparison between different evaluation functions applied to the player and the opponent. AB_Improved is the best evaluation function in this case. It has the highest win rate with using alpha-beta search and iterative deepening, it's more complex and it's evaluating both my_moves and the opp_moves instead of just my_moves, which goes one more level deeper of the game tree. Below graph is the comparison of evaluation functions between different test agents

competes against the cpu agents.