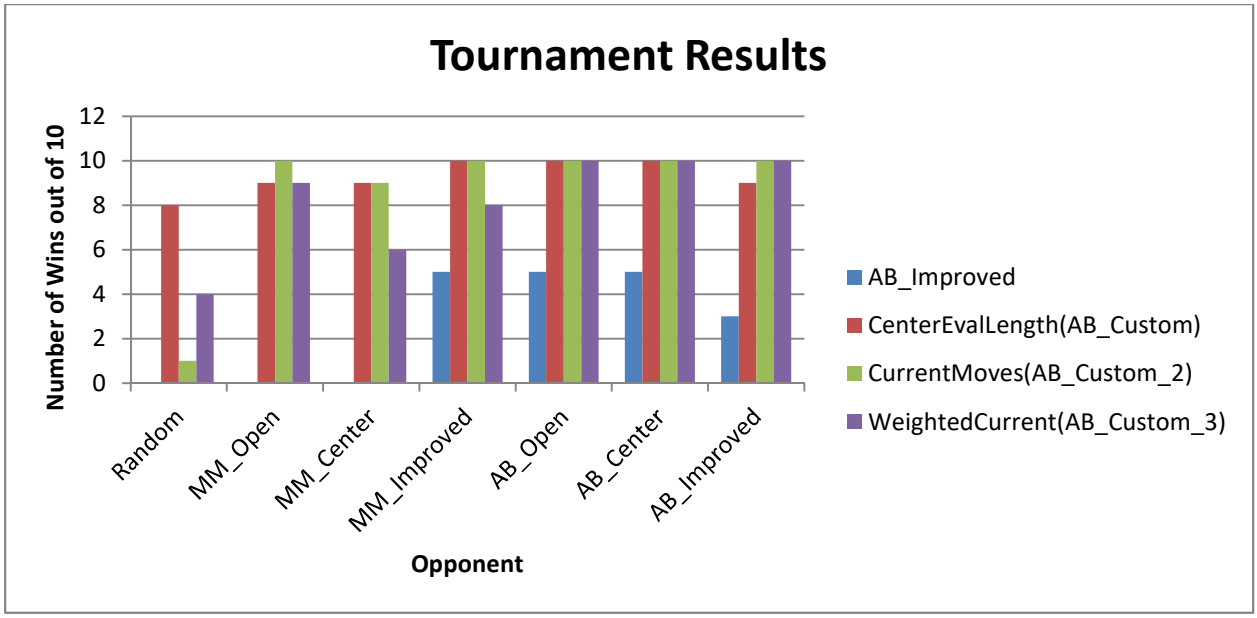
Heuristic Analysis:

Legend of Three Heuristics	
CenterEvalLength:	
custom_score:	= 2 * currentPlyrMovesLen + **CenterEvaluation()**
CurrentMoves:	
custom_score_2:	= currentPlyrMovesLen
WeightedCurrent:	
custom_score_3:	= 2 * currentPlyrMovesLen - opponentMovesLen

The analysis of the three different Heuristics/ Agents CenterEvalLength, CurrentMoves, and WeightedCurrent gave some interesting results, and is scored from the image of the results of the tournament.py output below. The scale from worse to best is WeightedCurrent, CurrentMoves, then CenterEvalLength. First, is the WeightedCurrent function which was from the idea that it is better to slightly weight (multiple of 2) the length of moves of the current player rather than the opponent. This heuristic did come in last with an overall Win Rate of 81.4% in this test which is interesting because it is quite competitive with the CurrentMoves Function in other tests. Next, the CurrentMoves function came in second place, and was from the theory that simple is better, and the number of moves available would be a good heuristic to consecutively win the game. Additionally, as simple as it is this heuristic works quite well, and is relatively simple to compute allowing it to search deeper into the game tree, but it did take second place with an overall Win Rate of 85.7%. Finally, the best function was CenterEvalLength which combined the average distance for each move from the center of the board, and the length of the number of moves available. The equation favored the number of moves (multiple of 2) over the CenterEvaluation() to be more effective, and had the best Win Rate with 92.9%.

The CenterEvalLength function was quite effective. It had two different features summed into one evaluation function to help the agent grip the game and to consistently win. The CenterEvalution() sub-function gave a higher score for moves away from the center, assuming that moves away from the center are harder to achieve, and an agent that has more of those moves has the advantage. The Third reason, is the fast compute time of this function since currentPlyrMoves only needs to be calculated once and not twice, the Length of currentPlyrMoves calculation is fast, and the only calculation that is left is the CenterEvaluation() function. Additionally, the calculation time of this evaluation function would be in second place for the amount of compute time.

In conclusion, the results of this test were quite broad with a max deviation of +/- 11.5%, and the CenterEvalLength function had definitive best score with a Win Rate of 92.9%.




```
C:\Users\Andrews X6i7\AppData\Local\Programs\Python\Python35\python.exe
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                    **************************
                         Playing Matches
                    **************************


 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       0  |  10     8  |  2     1  |  9     4  |  6
    2       MM_Open      0  |  10     9  |  1    10  |  0     9  |  1
    3      MM_Center     0  |  10     9  |  1     9  |  1     6  |  4
    4     MM_Improved    5  |   5    10  |  0    10  |  0     8  |  2
    5       AB_Open      5  |   5    10  |  0    10  |  0    10  |  0
    6      AB_Center     5  |   5    10  |  0    10  |  0    10  |  0
    7     AB_Improved    3  |   7     9  |  1    10  |  0    10  |  0
--------------------------------------------------------------------------
          Win Rate:      25.7%        92.9%        85.7%        81.4%

Your agents forfeited 229.0 games while there were still legal moves available t
o play.


Press any key to continue . . .
```