I coded three evaluation functions for my game playing agent:

1. The first function calculates the difference between the number of moves for me and two times the number of moves for my opponent. I chose this because I liked the discussion in the lecture videos, and believe it's a simple but powerful function.
2. The second function focuses on distance from center. It calculates the difference between my opponent's distance from center and my distance from center. I believe managing the middle of the board is an important key to many two player perfect information games.
3. The third function is somewhat related to the second as it focuses on the number of "wall" moves, i.e. any board space that is up against an edge. It calculates the difference between my opponent's wall moves and mine.

All of my evaluation functions are relatively simple, but could be combined to deliver positive results. I chose the functions that I did based on discussions in the video lectures, as well as my own experiences playing games like chess and checkers. I'd actually never heard of or played isolation until this course. One function I was interested in writing but could not figure out how to implement is related to the "horizon effect" and basically looking for partitions on the board, and moves that would put my player on the wrong side of a partition.

My results from utilizing these evaluation functions are as follows:

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                     Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       5 |   5     4 |   6     5 |   5     7 |   3
    2       MM_Open     10 |   0    10 |   0    10 |   0    10 |   0
    3      MM_Center    10 |   0    10 |   0    10 |   0    10 |   0
    4     MM_Improved   10 |   0    10 |   0    10 |   0    10 |   0
```

5	AB_Open	5	5	5	5	4	6	4	6
6	AB_Center	3	7	4	6	3	7	6	4
7	AB_Improved	4	6	5	5	4	6	2	8

Win Rate: 67.1% 68.6% 65.7% 70.0%

All of the evaluation functions perform somewhat similarly and are less than 5% apart from each other when looking at the overall win rate. The only function that does not beat AB_Improved is the second heuristic that looks at distance from center. This heuristic probably needs to be combined with maybe the first heuristic to be more successful.

Based on this information, I would recommend utilizing the third function as it has the highest win rate overall. Another reason is that it performed the best against random opponents. It is also a fairly simple heuristic which does not require unnecessary overhead.