Heuristic Analysis

Shijie Wang

For this project, I studied with Udacity AIND Courses and came up with three different heuristics on the isolation game. And here is the summary of the result. The chart below presents the outcome on how each heuristic performs against the 'ID_improved' which was provided by the Udacity Staff. Based on the performance, the best heuristic to choose is the third one which I used to submit for review.

	ID_improved	Student	Deviation
Heuristic 1	65.71%	68.75%	3.04%
Heuristic 2	67.14%	70.71%	3.57%
Heuristic 3	**67.86%**	**71.43%**	**3.57%**

I recommend the Heuristic 3 for the following 4 reasons:

1. It rates computer current position against its opponent.
2. It considers a space is close or not.
3. It allows to virtually look ahead to predicting the winner or loser.
4. It translates the notion of positional advantage to the specific L-shape knight like moves.

Heuristic 1

Heuristic one simply returns the different number of moves left for each players. If the player and its opponent has the same number of moves, then the return value is zero. Whoever has the higher return value, wins the game. It is not a good heuristic, although it is easy to interpret and fast to compute. It is oblivious to the notion of positional advantage. Therefore, in the next section, I add the positional advantage implementation into the heuristic 2.

By running the tournament.py, we can analysis the result as showing below:

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random    Result: 15 to 5
  Match 2: ID_Improved vs   MM_Null   Result: 13 to 7
  Match 3: ID_Improved vs   MM_Open   Result: 14 to 6
  Match 4: ID_Improved vs MM_Improved Result: 13 to 7
  Match 5: ID_Improved vs   AB_Null   Result: 15 to 5
  Match 6: ID_Improved vs   AB_Open   Result: 11 to 9
  Match 7: ID_Improved vs AB_Improved Result: 11 to 9
```

```
Results:
----------
ID_Improved         65.71%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random    Result: 18 to 2
  Match 2:   Student   vs   MM_Null   Result: 13 to 7
  Match 3:   Student   vs   MM_Open   Result: 15 to 5
  Match 4:   Student   vs MM_Improved Result: 12 to 8
  Match 5:   Student   vs   AB_Null   Result: 15 to 5
  Match 6:   Student   vs   AB_Open   Result: 15 to 5
  Match 7:   Student   vs AB_Improved Result: 8 to 12


Results:
----------
Student             68.57%
```

Heuristic 2

In this approach, I output the difference in the number of moves available to two players but I added the penalizing and rewarding moves to the maximizing player that in the corner and minimizing player in the corner, respectively. The result compares to heuristic one increases a bit little since they are almost identical in terms of the logic. The difference I used a different way to return the moves on the behavior of two players when their remaining moves are located in the corners. The penalizing and rewards method works more accurately than the heuristic two. However, there are is still a way to increase the performance by implement a heuristic more precisely based on how long the player can be survived, for example, and adding this to the positional advantage.

By running the tournament.py, we can analysis the result as showing below:

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random    Result: 14 to 6
  Match 2: ID_Improved vs   MM_Null   Result: 16 to 4
  Match 3: ID_Improved vs   MM_Open   Result: 12 to 8
  Match 4: ID_Improved vs MM_Improved Result: 12 to 8
  Match 5: ID_Improved vs   AB_Null   Result: 12 to 8
  Match 6: ID_Improved vs   AB_Open   Result: 12 to 8
  Match 7: ID_Improved vs AB_Improved Result: 16 to 4


Results:
----------
ID_Improved         67.14%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random    Result: 15 to 5
  Match 2:   Student   vs   MM_Null   Result: 16 to 4
  Match 3:   Student   vs   MM_Open   Result: 16 to 4
  Match 4:   Student   vs MM_Improved Result: 11 to 9
  Match 5:   Student   vs   AB_Null   Result: 17 to 3
  Match 6:   Student   vs   AB_Open   Result: 12 to 8
  Match 7:   Student   vs AB_Improved Result: 12 to 8


Results:
----------
Student             70.71%
```

Heuristic 3

With this heuristic, I have included the positioning evaluation. If a player is closer to the center of the board, it is more than likely that this player will do better than a player who has the moves near the edges of the board. This heuristic performs a better result that is the reason I choose it to submit. However, there is still an edge case can be included in order to make a even better performance. If there is no clear positional advantage, for example if both players are at the same distance from the center, then we can measure the longest run of moves that can be performed inside the 3x3 squares which is defined by the starting position and each of its legal moves left, then the longest run one can hope to reach is 7. If the returned value is positive, then `player` is doing better than its opponent vice versa. For now, the Heuristic 3 is good enough to submit.

By running the tournament.py, we can analysis the result as showing below:

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random    Result: 14 to 6
  Match 2: ID_Improved vs   MM_Null   Result: 17 to 3
  Match 3: ID_Improved vs   MM_Open   Result: 13 to 7
  Match 4: ID_Improved vs MM_Improved Result: 11 to 9
  Match 5: ID_Improved vs   AB_Null   Result: 17 to 3
  Match 6: ID_Improved vs   AB_Open   Result: 11 to 9
  Match 7: ID_Improved vs AB_Improved Result: 12 to 8


Results:
----------
ID_Improved         67.86%

*************************
   Evaluating: Student
*************************

Playing Matches:
----------
  Match 1:   Student   vs   Random    Result: 14 to 6
  Match 2:   Student   vs   MM_Null   Result: 18 to 2
  Match 3:   Student   vs   MM_Open   Result: 14 to 6
  Match 4:   Student   vs MM_Improved Result: 11 to 9
  Match 5:   Student   vs   AB_Null   Result: 16 to 4
  Match 6:   Student   vs   AB_Open   Result: 14 to 6
  Match 7:   Student   vs AB_Improved Result: 13 to 7


Results:
----------
Student             71.43%
```