The 3 heuristic functions are as described below:

1. custom_score_2: Here the assumption is that if the player can chase the opponent to a corner then there is a chance of winning. The evaluation function finds the distance between the two player locations and computes the value as (game_width * game_height)/distance. Thus it gives a low score if the distance between the players is more and high score if the distance between the players is low. When compared against the other agents as shown in the below tables we can see that the win rate varies from 37% to 45.7%. So this indicates that although this assumption helps us to win some games but there are other factors which are equally important in winning the game.

2. custom_score _ 3: Here I added one more factor which is if the opposition player has any move that I can block then give a higher score(infinity) for that board position. Else chase the opponent to a corner as described under custom_score_2. So basically here I just combined 2 factors that I think can win a game. But the winning rate for this heuristic varies from 33% to 38% as seen in the below tables. So its performance is still below that of AB_Improved.

3. custom_score (submitted): After carefully observing the behavior of above two heuristics and also that of AB_Improved it appears that a heuristic function to correctly predict the winning states has to be a combination of several factors . So I came up with these 3 factors and calculated a weighted score based on them given by :

   score = 0.5 * difference in moves + 0.1 * distance of the player from the center + 0.3 * val (val is 1 if the player location blocks any of opposition's moves or 0, otherwise)
   The reasons for choosing the factors are as detailed below:

   i. If the Difference between move count difference of the player and opposition player is high then the score will also be high. In other words, if the opposition has less moves available then that means the score will be high as there is a chance to win.
   ii. Distance of the player from the center – give a high score to a move which is away from the center. Here the assumption is that the computer player will be chasing the opposition to the corner.
   iii. Give a higher score to a Move that blocks one of the move of opposition player

   So in view of the above factors I designed a heuristic which is a weighted combination of the above factors. The values of the parameters I arrived by trial and errors after observing several games.
   As can be seen in the tables below, this evaluation function helps in winning the maximum games and its winning rate is above 70% all the time.

So this heuristic custom_score wins all the games against Random most of the time whereas other heuristics loses games against random quite frequently. The win rate of the other two custom heuristics is always below 50%. It even outperforms AB_Improved. So it is definitely a better heuristic than all other heuristics tried so far.

Test 1

```
In [21]: runfile('C:/Users/Suman/Documents/AIND/AIND-Isolation/tournament.py', wdir='C:/Users/Suman/
Documents/AIND/AIND-Isolation')
Reloaded modules: sample_players, isolation.isolation, isolation

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
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      10  |   0     10  |   0     9  |   1     8  |   2
    2       MM_Open      4  |   6      7  |   3     2  |   8     2  |   8
    3      MM_Center     9  |   1      7  |   3     5  |   5     3  |   7
    4     MM_Improved    5  |   5      8  |   2     3  |   7     1  |   9
    5       AB_Open      4  |   6      7  |   3     4  |   6     6  |   4
    6      AB_Center     6  |   4      7  |   3     6  |   4     3  |   7
    7     AB_Improved    6  |   4      6  |   4     3  |   7     4  |   6
--------------------------------------------------------------------------
          Win Rate:      62.9%        74.3%        45.7%        38.6%

There were 43.0 timeouts during the tournament -- make sure your agent handles search timeout correctly, and
consider increasing the timeout margin for your agent.


Your ID search forfeited 117.0 games while there were still legal moves available to play.
```

Test 2

```
In [33]: runfile('C:/Users/Suman/Documents/AIND/AIND-Isolation/tournament.py', wdir='C:/Users/Suman/
Documents/AIND/AIND-Isolation')
Reloaded modules: sample_players, isolation.isolation, isolation, game_agent

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
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       8  |   2     10  |   0     9  |   1     8  |   2
    2       MM_Open      6  |   4      9  |   1     1  |   9     0  |  10
    3      MM_Center     9  |   1     10  |   0     7  |   3     4  |   6
    4     MM_Improved    5  |   5      7  |   3     5  |   5     3  |   7
    5       AB_Open      4  |   6      4  |   6     3  |   7     4  |   6
    6      AB_Center     4  |   6      6  |   4     5  |   5     4  |   6
    7     AB_Improved    5  |   5      5  |   5     3  |   7     4  |   6
--------------------------------------------------------------------------
          Win Rate:      58.6%        72.9%        47.1%        38.6%

Your ID search forfeited 146.0 games while there were still legal moves available to play.
```

Test 3.

.

```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
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       7  |   3    10  |   0    7  |   3    8  |   2
    2        MM_Open      8  |   2     8  |   2    3  |   7    3  |   7
    3       MM_Center     9  |   1    10  |   0    2  |   8    2  |   8
    4     MM_Improved     7  |   3     9  |   1    4  |   6    2  |   8
    5        AB_Open      5  |   5     7  |   3    3  |   7    4  |   6
    6       AB_Center     5  |   5     5  |   5    4  |   6    4  |   6
    7     AB_Improved     6  |   4     4  |   6    2  |   8    3  |   7
----------------------------------------------------------------------------
           Win Rate:      67.1%        75.7%        35.7%        37.1%

Your ID search forfeited 142.0 games while there were still legal moves available to play.
```