HEURISTICS 1

Description:

Heuristics 1 returns a score based on the number of moves of a node two levels deep in a search tree from the current board state.

```
def heuristics_1(game, player):

    score = 0.0
    for move in game.get_legal_moves(player):
        score = max(score,len(game.forecast_move(move).get_legal_moves(player)))
    return score
```

Result:

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random       Result: 78 to 22
  Match 2: ID_Improved vs   MM_Null      Result: 68 to 32
  Match 3: ID_Improved vs   MM_Open      Result: 39 to 61
  Match 4: ID_Improved vs MM_Improved    Result: 43 to 57
  Match 5: ID_Improved vs   AB_Null      Result: 51 to 49
  Match 6: ID_Improved vs   AB_Open      Result: 35 to 65
  Match 7: ID_Improved vs AB_Improved    Result: 41 to 59


Results:
----------
ID_Improved         50.71%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random       Result: 81 to 19
  Match 2:   Student   vs   MM_Null      Result: 63 to 37
  Match 3:   Student   vs   MM_Open      Result: 62 to 38
  Match 4:   Student   vs MM_Improved    Result: 59 to 41
  Match 5:   Student   vs   AB_Null      Result: 65 to 35
  Match 6:   Student   vs   AB_Open      Result: 43 to 57
  Match 7:   Student   vs AB_Improved    Result: 42 to 58
```

```
Results:
----------
Student              59.29%
```

Analysis:

The game agent outperforms ID_Improved with a win rate of 59.29% to 50.71% in a 100-match game each against various opponents.

HEURISTICS 2

Description:

Heuristics 2 returns a score based on common legal moves between game agent and opponent. If such a move does not exist, it returns the difference between legal moves available to the game agent and opponent

```
def heuristics_2(game, player):

    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    move = [own_move for own_move in game.get_legal_moves(player) for opp_move in 
game.get_legal_moves(game.get_opponent(player)) if own_move == opp_move]

    if move:
        game_copy = game.forecast_move(move[0])
        result = len(game_copy.get_legal_moves(player))
        if result > 2:
            return float('inf')
        else:
            own_moves = len(game_copy.get_legal_moves(player))
            return float((2 * own_moves) - opp_moves)
    return float((2 * own_moves) - opp_moves)
```

Result:

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random       Result: 53 to 47
  Match 2: ID_Improved vs   MM_Null      Result: 57 to 43
  Match 3: ID_Improved vs   MM_Open      Result: 54 to 46
  Match 4: ID_Improved vs MM_Improved    Result: 41 to 59
  Match 5: ID_Improved vs   AB_Null      Result: 52 to 48
  Match 6: ID_Improved vs   AB_Open      Result: 51 to 49
  Match 7: ID_Improved vs AB_Improved    Result: 54 to 46


Results:
----------
ID_Improved         51.71%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random       Result: 84 to 16
  Match 2:   Student   vs   MM_Null      Result: 71 to 29
  Match 3:   Student   vs   MM_Open      Result: 57 to 43
  Match 4:   Student   vs MM_Improved    Result: 50 to 50
  Match 5:   Student   vs   AB_Null      Result: 47 to 53
  Match 6:   Student   vs   AB_Open      Result: 41 to 59
  Match 7:   Student   vs AB_Improved    Result: 60 to 40


Results:
----------
Student             58.57%
```

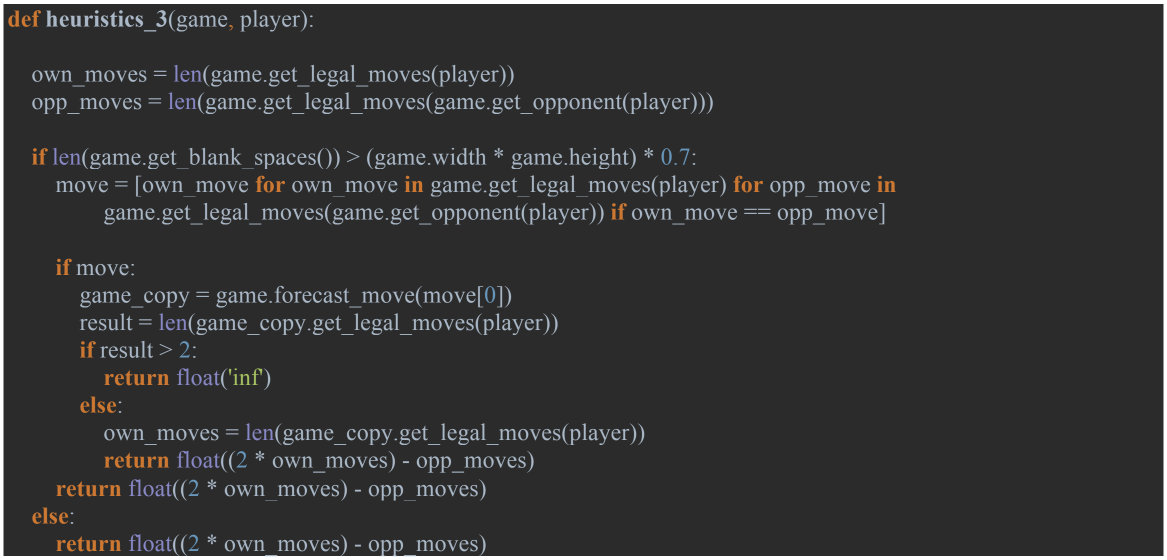
Analysis:

The game agent outperforms ID_Improved with a win rate of 58.57% to 50.71% in a 100-match game each against various opponents.

HEURISTICS 3

Description:

Heuristics 3 is a modification of heuristics 2.Scores are based on the number of available spaces left in the game.



```
def heuristics_3(game, player):

    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

    if len(game.get_blank_spaces()) > (game.width * game.height) * 0.7:
        move = [own_move for own_move in game.get_legal_moves(player) for opp_move in
                game.get_legal_moves(game.get_opponent(player)) if own_move == opp_move]

        if move:
            game_copy = game.forecast_move(move[0])
            result = len(game_copy.get_legal_moves(player))
            if result > 2:
                return float('inf')
            else:
                own_moves = len(game_copy.get_legal_moves(player))
                return float((2 * own_moves) - opp_moves)
        return float((2 * own_moves) - opp_moves)
    else:
        return float((2 * own_moves) - opp_moves)
```

Result:

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 61 to 39
  Match 2: ID_Improved vs   MM_Null     Result: 56 to 44
  Match 3: ID_Improved vs   MM_Open     Result: 42 to 58
  Match 4: ID_Improved vs MM_Improved   Result: 48 to 52
  Match 5: ID_Improved vs   AB_Null     Result: 47 to 53
  Match 6: ID_Improved vs   AB_Open      Result: 52 to 48
  Match 7: ID_Improved vs AB_Improved    Result: 54 to 46


Results:
----------
ID_Improved         51.43%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random       Result: 78 to 22
  Match 2:   Student   vs   MM_Null      Result: 75 to 25
  Match 3:   Student   vs   MM_Open      Result: 64 to 36
  Match 4:   Student   vs MM_Improved    Result: 52 to 48
  Match 5:   Student   vs   AB_Null      Result: 47 to 53
  Match 6:   Student   vs   AB_Open      Result: 48 to 52
  Match 7:   Student   vs AB_Improved    Result: 52 to 48


Results:
----------
Student             59.43%
```

Analysis:

The game agent outperforms ID_Improved with a win rate of 59.43% to 51.43% in a 100-match game each against various opponents.

Recommendation:

Heuristics 3 is recommended for the game agent, as it is the best performing heuristics with the highest win rate of 59.43%. It generalizes better than the other two heuristics when playing against various opponents. Though winning marginally against MM_Improved and AB_Improved, it performs better than heuristics 1 and 2.