The first evaluation function(Student_1) is modified from the improved_score. I add more weights to the opponent player's legal moves so that the Student Player should behave more aggreesive.

```
def custom_score(game, player):

    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float(own_moves - 4*opp_moves)
```

The second evaluation function(Student_2) will predict all the next states of next legal_moves then calculate the sum of legal_moves of next states. This function should be able to evaluate how well the next moves of next move are.

```
def custom_score(game, player):

    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = 0.0

    for move in game.get_legal_moves(player):
        if player.time_left() < player.TIMER_THRESHOLD:
            raise Timeout()
        own_moves += len(game.forecast_move(move).get_legal_moves(game.forecast_mo
ve(move).__inactive_player__))

    return float(own_moves)
```

The third evaluation function(Student_3) is like the second one. The function will calculate the number of legal_moves of next moves of Student minus those of opponent agent.

```
def custom_score(game, player):

    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    own_moves = 0.0
    opp_moves = 0.0

    for move in game.get_legal_moves(player):
        if player.time_left() < player.TIMER_THRESHOLD:
            raise Timeout()
        own_moves += len(game.forecast_move(move).get_legal_moves(game.forecast_mo
ve(move).__inactive_player__))

    for move in game.get_legal_moves(player):
        if player.time_left() < player.TIMER_THRESHOLD:
            raise Timeout()
        opp_moves += len(game.forecast_move(move).get_legal_moves(game.forecast_mo
ve(move).__active_player__))

    return float(own_moves - opp_moves)
```

For each evaluation I have ran tournament.py for 5 times so the Student Agent and ID_Improved Agent had play 100 times with each opponent Player so that the result should be more precise.

I recommend the third evaluation function(the Student_3 as below), for these reasons:
* It has the highest scores among the three functions
* Its behaviour are more stable while the other two are sometimes good and sometimes bad
* Comparing to the first function, it considers more moves in the future, and comparing to the second one, it calculates moves of both players but not just the moves of active player

---

	ID_improved	**Student_1**	**Student_2**	**Student_3**
Random	86%	90%	80%	84%
MM_Null	72%	86%	81%	80%
MM_Open	66%	75%	62%	76%
MM_Improved	63%	79%	69%	70%

AB_Null	69%	91%	70%	78%
AB_Open	76%	83%	78%	80%
AB_Improved	73%	85%	83%	80%
Average	72.13%	73.57	74.71%	77.43%