Assignment 3 - Coding component

● Graded

Student

Tyler NGUYEN

Total Points

100 / 100 pts

Autograder Score

50.0 / 50.0

Passed Tests

1.1) Test case I (10/10)
1.2) Test case II (10/10)
1.3) Test case III (10/10)
1.4) Test case IV (10/10)
1.5) Test case V (10/10)

Question 2

Manual Grading **50** / 50 pts

✓ **– 0 pts** Correct

– 10 pts Not sufficient document/Not clean code.

– 20 pts Not caching the partial results

– 30 pts Not implementing the algorithm in class.

– 50 pts Assignment is submitted after allowed number of late days. Note that the grade of 0 will be applied to the

Autograder Results

1.1) Test case I (10/10)

1.2) Test case II (10/10)

1.3) Test case III (10/10)

1.4) Test case IV (10/10)

1.5) Test case V (10/10)

Submitted Files

coin_game.py Download

```python
#CPSC 413 Assignment 3 question 3
#Author: Tyler Nguyen
#UCID: 30158563
class Game:
    def __init__(self, coins):
        self.coins = coins

    def run(self, start, end):

        cache = [[-1] * len(self.coins) for _ in range(len(self.coins))] #initialize the
        #2D array, cache to store the results of the subproblems, initialized to -1
        #to indicate the result for a subproblem has not been computed
        max_win = 0
        margin = 0
        takeRight = True

        def CG(start, end): #CG, recursive function to calculate the maximum win
            if start > end: #base case 1
                return 0 #if there are no more coins left
            if start == end: #base case 2
                return self.coins[start] #if there is only one coin left
            if cache[start][end] != -1: #check if the result is already in the 2D array
                return cache[start][end]
            #determines the score by the current player if they choose to take the leftmost coin
            left = self.coins[start] + min( #recursive formula 1
                CG(start + 2, end), #opponent takes left coin
                CG(start + 1, end - 1)  #opponent takes right coin
            )
            #determines the score by the current player if they choose to take the rightmost coin
            right = self.coins[end] + min( #recursive formula 2
                CG(start + 1, end - 1),  #opponent takes left coin
                CG(start, end - 2)  #opponent takes right coin
            )
            cache[start][end] = max(left, right) #update the cache with the calculated result
            return cache[start][end]
        max_win = CG(start, end) #call CG function to determine the maximum win
        #determine the margin
        opponent_max = min(
            CG(start + 1, end), #if player 1 takes left the opponent starts here
            CG(start, end - 1)  #if player 1 takes right the oppent starts here
        )
        margin = max_win - opponent_max
        #determine which side the opponent will take from
        right = self.coins[end] + min(
            CG(start + 1, end - 1), #Opponent takes left coin
            CG(start, end - 2) #Opponent takes right coin
        )
        takeRight = right >= max_win #checking if right coin results in a win greater or
        #equal to the maximum win achieved
    return (max_win, margin, takeRight) #returns the maximum win, margin and decision on which side to take

```