1)
min_coin_with_plan part is wrong. First error occurred when n = 6006

2)
min_coin_with_plan violated the **overlapping of sub-problem** elements. As the algorithm is tracking all possible combinations that lead to the minimum number of coins for certain n.

Such approach will increase the complexity and memory usage, but in DP what we want is the value of the optimal sub-problem, not all the possible combinations for the optimal sub-problem.

3)
The first occurred error is when n = 6006.
For denom = [(1, 10), (2, 10), (3, 13), (7, 17), (14, 14), (29, 18), (57, 20), (115, 12), (231, 17), (462, 12)]

When the amount for coin change = **6006 cents**, the corresponding optimal solution should be **14 coins**(12 coins of 462 cents and 2 coins of 231), but the code(min_coin_with_plan) gives the following results : **AttributeError: 'NoneType' object has no attribute 'num_coin'**. When i = 6006

Which means that min_coin_with_plan method does not provide a solution for 6006 cents.
Due to the third for loop, when i=9, it gets the plan for min_coin_with_plan[5544].plan (which will use 12 coins of 462 cents) and then it would not go the if statement, as 5544 cents will used all the supply for 462 cents coin.

4)
To rectify the bug for min_coin_with_plan. For the first for loop, instead of starting i from 0 to m -1(increment), start from m - 1 to 0(decrement),so that when j = 6006, and all the supply for the largest is used.

Code :
Original code(line 29) => **for i in range(m):**
Proposed correction => **for i in range(m - 1, -1, -1):**

As the result, the code will consider the largest coin first and then the smallest coin, which is similar with the greedy approach.
Unlike the current code, when all the supply for the largest is used, that is no way for it to consider the smaller value coin, as i cannot go back, since it is start from 0 to m - 1.