Consider again the problem of making change when the denominations are arbitrary.

> Input: Positive integer amount A, positive integer denominations d[1] < d[2] < ... < d[m] for m >= 1.
> Output: List of coins c = [c[1],....,c[n]], where each c[i] is in d, repeated coins are allowed (possible for c[i] = c[j] with i != j), c[1] + ... + c[n] = A, and n is minimum. If no solution is possible, the empty list [] is returned.

For example, if we only have pennies, dimes and quarters to make change for 30c, then the input is d = [1,10,25] and an optimum output is c = [10,10,10]. If we only have nickels, dimes and quarters to make change for 52c, then an optimum output is c = [] -- no solution exists.

Follow the dynamic programming paradigm given in class to solve this problem.

Then, analyze the worst-case runtime of your algorithm carefully. Does it run in polynomial time? Explain.