5A Find the Minimum Number of Coins Needed to Make Change

The Change Problem
Find the minimum number of coins needed to make change.

Input: A non-negative integer *money* and an array *Coins* of positive integers.
Output: The minimum number of coins with denominations *Coins* that changes *money*.

Formatting

Input: A non-negative integer *money* followed by a space-separated list of positive integers *Coins*.
Output: The minimum number of coins with denominations *Coins* that changes *money*.

Constraints

- The value of *money* will be between 1 and 10^5.
- The number of positive integers in *Coins* will be between 1 and 10^1.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code.

Input:

```
7
1 5
```

Output:

```
3
```

Case 2

Description: This dataset makes sure that your code is correctly considering the last coin denomination in the *Coins* array. If your solution has an off-by-one indexing mistake while iterating over the *Coins* array it could be possible that the last *coin* is not considered. In this case code run on this dataset will output `2` instead of the correct answer, `1`.

Input:

```
10
1 2 3 4 5 10
```

Output:

```
1
```

Case 3

Description: This dataset makes sure that your code is correctly considering the first coin denomination in the *Coins* array. If your solution has an off-by-one indexing mistake while iterating over the *Coins* array it could be possible that the first *coin* is not considered. In this case code run on this dataset will output `2` instead of the correct answer, `1`.

Input:

```
10
10 5 4 3 2 1
```

Output:

```
1
```

Case 4

Description: This dataset checks if your code correctly returns the final value in the array in the dynamic programming approach for solving this problem. It is possible that an off-by-one indexing error (could be related to confusing 0/1 indexing) results in your code outputting the minimum number of coins needed to make change for $money-1$ instead of *money*. In this case your code will output `2` instead of the correct value of 3.

Input:
```
11
1 5
```

Output:
```
3
```

Case 5

Description: This dataset checks to make sure you are not using a greedy algorithm to solve this problem. While a greedy algorithm in which the largest valued *coin* is used on each iteration may work in some cases, it will fail on this dataset. A greedy approach would start by using the 9 *coin* which would only allow for the use of 3 of the 1 *coin* to get to 12, resulting in an output of `4`. Using 2 of the 6 *coin* will result in exact change with only 2 coins.

Input:
```
12
9 6 1
```

Output:
```
2
```

Case 6

Description: A larger dataset of the same size as that provided by the randomized autograder. Check input/output folders for this dataset.