**PROBLEM A: MINIMUM COIN CHANGE**

A new vending machine is to be designed for international sale. Therefore, this vending machine must work for any currency. However, only one unit of currency is applicable at each setting. The unique quality of this vending machine is, no matter for what set of coin denominators, and even when some coin runs out, it will give the change with minimum number of coins possible, as long as the coin valued 1 does not run out.

For a product, the price in some currency can be rather a large integer. The amount of change may reach the value as large as 10,000.

The first step is to find the minimum of number of coins required for a change.

INPUT:

Line 1 : the list of coin denominator
Line 2 : the amount of change

OUTPUT: The minimum number of coins required for the change

EXAMPLE

| INPUT | OUTPUT |
|---|---|
| 1 3 4 5<br>7 | 2 |
| 1 2 5 10 13<br>3377 | 260 |

1. Write the program to read in the input in such a way that the coin denominators are kept as a list of integers. The program will refer to this list for coin choices.

2. Assume a situation that the current value of change is v, and a function mincoin(v) will give the minimum number of coins for this value. Approach the following problems in the prescribed order!
   - A coin is to be picked for the change. What are the options (possible coin values) for this coin?
   - What is the remaining value of change after picking each option for that coin?
   - *What **is** the minimum number of coins* for ***each*** *remaining value of change*? (This a critical step that you must learn. The answer is quite philosophical)
   - **THEN**, what would be the value of mincoin(v)?
   - What is(are) the value(s) of change for which the minimum number of coins is obvious?

3. Develop a Python program to solve this problem, based on the concept derived in question 2.
   **NOTE** Don't bother to try the 2nd test case in the example, a brute-force program cannot handle it.

4. Add a global variable to count the number of functions that are called. Simply increment this global variable every time the function is called to execute. For a fixed set of denominators, gradually increase the change value and observe trend of how the number of recursive calls grows.

**PROBLEM B: ROD CUTTING**

Serling Enterprises buys long steel rods and cuts them into shorter rods, which it then sells. Each cut is free. The management of Serling Enterprises wants to know the best way to cut up the rods.

We assume that we know, for $i \in 1, 2, \ldots$, the price $p_i$ in dollars that Serling Enterprises charges for a rod of length i inches. Rod lengths are always an integral number of inches.

| length i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| price $p_i$ | 1 | 5 | 8 | 9 | 10 | 17 | 17 | 20 | 24 | 30 |

The rod-cutting problem is the following. Given a rod of length L inches and a table of prices $p_i$ for $i \in 1, 2, \ldots, L$, determine the maximum revenue obtainable by cutting up the rod and selling the pieces.

INPUT: the sequence of price, $p_i$, $1 \le i \le L$

OUTPUT: the maximum revenue obtainable

1) Given a rod of length $l$, list out all the possible ways to cut the rod "one time"? What is the result of each cutting? (To sell a rod at the cut length and keep the rest for the next sells.)

2) Following question 1 above, if the maximum revenue for a rod of length $l$ is determined by function maxRev($l$), what is the maximum revenue possible as the result of *each* cutting in question 1?

3) THEN, what would be the value of maxRev($l$) ?

4) Develop a recursive solution to rod-cutting problem and implement it as a Python program, using the concept provided in question 2 above.

5) Add a global variable to count the number of functions that are called. Test the program with the test files that contains incremental lengths of rod, L. Then observe trend of how the number of recursive calls grows.