Lab 4D

Problem 1: Ancient Mystery

In an ancient scroll, each letter hides a number—its position in the alphabet ('a' is 1, 'b' is 2, and so on). To unlock the secret, the sum of the digits must be found and repeated k times. Can you reveal the scroll's hidden number?

Example

Input: `zbax 2`
`"z"` is written as "26", `"b"` as "2",`"a"`:1,`"x"`:24
So the number obtained is `262124`
In operation 1: `2+6+2+1+2+4 = 17`
In operation 2: `1+7 = 8`
Hence answer is `8`.

Input:

- A single line containing a string `s` and an integer `k`, the given string and number of operations.

Output:

- A single line conatining 1 integer, the final number after conducting `k` operations on the string `s`

Constraints:

- `1 ≤ |s| ≤ 100` where `|s|` is the length of the string `s`;
- `1 ≤ k ≤ 10`

Example:

Input:

```
zbax 2
```

Output:

```
8
```

Example 2:

Input:

```
naimsqfijjckfeccoxrsdt 9
```

Output:

```
5
```

Problem 2: Reach Your Goal

In a mystical world, you are a hero on a quest to defeat a powerful enemy. To vanquish this foe, you must gather a specific amount of energy, represented by a target value E.

land is filled with magical stones, each containing an amount of energy. These stones are scattered across the land, and their energies are present in an array. You can teleport to any stone and collect its energy, but to defeat the enemy as quickly as possible, you want to gather the required energy in the minimum number of steps.

Your task is to collect energy from the stones until you reach or exceed E and you need to determine the minimum number of steps required to reach this target energy. If you can't reach the target energy you need to print -1.

Input:

- First line contains the number of elements n.
- Second line contains the elements of the array A, where A[i] represents the energy of the i-th stone
- Third line cotains the target energy E

Output:

- A single integer which denotes the minimum number of steps required to reach the

target or -1 (if not possible to reach the sum)

Constraints:

- 1 ≤ n ≤ 10^3 (size of the array)
- 1 ≤ A[i] ≤ 10^9(energy values of the stones).
- 1 ≤ E ≤ 10^12 (target energy to collect).

Example 1:

Input:

```
5
8 7 5 3 1
8
```

Output:

```
1
```

Example 2:

```
4
4 3 2 1
50
```

Output:

```
-1
```

Problem 3: Cyborg Circuitry Conundrum

Objective: In the futuristic world of Neo-City, elite hackers like you are tasked with manipulating advanced circuits to keep the AI systems running smoothly. Today, you're facing a challenge from the Cyborg Syndicate. They've created two energy arrays of different lengths, and your job is to combine them strategically to balance the energy flow across the system.

These two energy arrays are like looping circuits—they repeat when their limits are reached! If one array is smaller than the other, its energy values "wrap around" to keep the calculations going. Your mission is to add corresponding elements from these arrays, taking the wrap-around effect into account.

Here's how you'll handle the situation:

- Take two arrays of energy values, each with a different number of elements.
- Add them together, element by element. If one array runs out of elements, restart from the beginning of that array.
- Print the final combined energy array.

Can you master the Cyborg Circuitry Conundrum and keep the systems running before the next AI attack?

Input:

- The first line contains two integers, `M,N` representing the size of the arrys.
- The second line contains `M` integers, each representing the energy values of the first array.
- The third line contains `N` integers, each representing the energy values of the second array.

Output:

- Print the final combined energy array.

Constraints:

- 1 ≤ `N,M` ≤ 1000000.
- -10^6 ≤ `Arr[i]` ≤ 10^6

Example 1:

Input:

```
3 1
1 2 3
1
```

Copy

Output:

```
2 3 4
```

1 gets added to each element in the first array

Example 2:

Input:

```
3 6
1 2 3
1 2 3 4 5 6
```

Copy

Output:

```
2 4 6 5 7 9
```

1 gets added to 1, 2 to 2, 3 to 3, and then the first array wraps around so 1 gets added to 4 and so on

Good luck!
