

Little Monkeys (1/8 Points)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

points: 100 · problem code: littlemonkeys · author: glipR

n little monkeys jumping on the bed,
Pick a random monkey, the instructions read
Off the sleepy trampoline, the monkey fled
One less monkey jumping on the bed

There are n little monkeys jumping on the bed. You've devised a program that can randomly select monkeys to fall off the bed, so you are left with 0 little monkeys jumping on the bed.

You've decided to model this by assigning a positive weight to each of the monkeys. For example you might start with the following monkey names and associated weights:

```
names = ["George", "Kong", "Rafiki", "Caesar", "Koko"]
weights = [ 4,      2,      3,      1,      5 ]
```

Then, for a list of size n we generate $n - 1$ random numbers between 0 and 1. We then use these random numbers to compute the next distinct monkey in the sequence bit by bit in the following manner. Suppose the 4 random numbers we generated was 0.8, 0.2, 0.5, 0.8.

Iteration 1

- First, we sum up all of the weights in the current list ($4+2+3+1+5=15$), and multiply this by your random number ($0.8*15=12$)
- Next, we find the first remaining element whose *left partial sum* is larger or equal to the computed value ($4 + 2 + 3 + 1 = 10 < 12$, but $4 + 2 + 3 + 1 + 5 \geq 12$, so "Koko" is selected)
- The selected item and weight is removed from the list and added to the order

The left partial sum is the result of summing all weights left of the associated name. So the left partial sum of "Rafiki" is $4+2+3=9$.

Current remaining list:

```
names = ["George", "Kong", "Rafiki", "Caesar"]
weights = [ 4,      2,      3,      1 ]
ordering = ["Koko"]
```

Iteration 2

- Summing up the remaining weights gives 10, multiplying by 0.2 we get 2.
- $4 \geq 2$, so "George" is selected.
- This is removed from our values and added to the current order.

```
names = ["Kong", "Rafiki", "Caesar"]
weights = [ 2,      3,      1 ]
ordering = ["Koko", "George"]
```

Iteration 3

- Summing up the weights gives 6, multiplying by 0.5 gives 3
- $2 < 3$ but $2 + 3 = 5 \geq 3$, so "Rafiki" is selected
- This is removed from our values and added to the current order.

```
names = ["Kong", "Caesar"]
weights = [ 2,      1      ]
ordering = ["Koko", "George", "Rafiki"]
```

Iteration 4

- Summing up the weights gives 3, multiplying by 0.8 gives 2.4
- $2 < 2.4$ but $2 + 1 = 3 \geq 2.4$, so "Caesar" is selected
- This is removed from our values and added to the current order.
- And since only "Kong" is left, it is added to the end of the ordering, with no random number required.

```
ordering = ["Koko", "George", "Rafiki", "Caesar", "Kong"]
```

Your problem is, given the monkey names, weights, and the $n - 1$ random numbers generated, generate the order of monkeys.

Input

Input will start with a single integer n , the number of monkeys on the bed.

The next line will contain n space separated strings, these are the *names* of the monkeys.

The next line will contain n space separated integers, these are the *weights* of the monkeys.

The next line will contain $n - 1$ space separated floating point values, these are the random numbers generated.

Output

Output n space separated strings representing the correct ordering.

Constraints

- Each string only contains characters from `a-z` and `A-Z`, and is at most 6 characters long
- You may assume that changing the random numbers by $\pm 10^{-13}$ will not change the result (IE you can ignore floating point shenanigans, with enough precision)
- All values are less than or equal to 10^6

Test Set A: 12.5%

- $1 \leq n \leq 1000$

Test Set B: 87.5%

- $1 \leq n \leq 2 \times 10^5$

Example

For input:

```
5
George Kong Rafiki Caesar Koko
4 2 3 1 5
0.8 0.2 0.5 0.8
```

The output should be:

```
Koko George Rafiki Caesar Kong
```

For the reasoning outlined earlier in the statement.