Goal

Remember, in one of the previous problems, you had a mysterious function obtained by linear interpolation, and you wanted to know how many times it reached the **N** / 2 value. Now, you apply the function several times, which, you hope, will improve the efficiency of your cryptographic countermeasure by better mixing the interval [0, N]. The question then is: how many times does the iterated function pass through the **N** / 2 value**. To avoid many special cases, we will limit ourselves to odd N values.**

Data

Input

Row *1*: an odd integer **N** between *1* and *99*.
Row *2*: **N** + 1 integers (not N, beware!) Separated by spaces, indicating f (0), f (1), ..., f (N) (where f is your function).
Row *3*: an integer **k** between *1* and *100*, the number of iterations.

Output

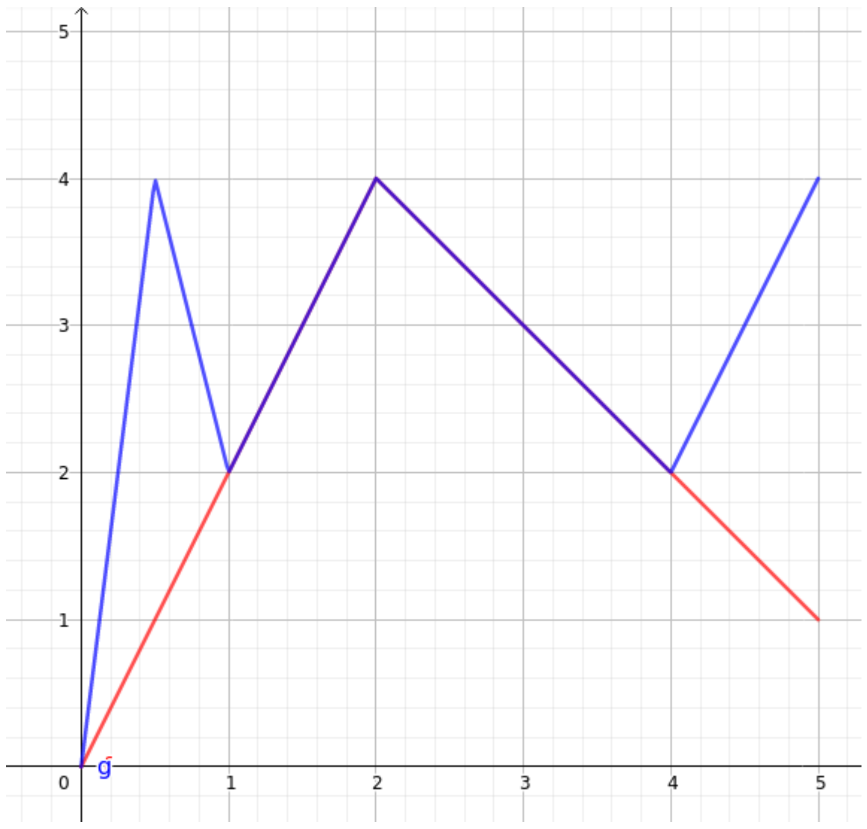
The last three digits (in base 10) of the number of different entries between 0 and **N** for which **f** iterate **k** times is **N** / 2.

The reason you are only asked for the last 3 digits is that when the number of iterations **k** is large, the number of points on which the iterated function is **N** / 2 can become very large.

Example

For **N** = 5, consider the function that is worth:
f (0) = 0, f (1) = 2, f (2) = 4, f (3) = 3, f (4) = 2, f (5) = 1

f itself reaches 2.5 twice, but the function that returns f(f(f(x))) on the input **x** reaches 2.5 five times. In the figure below, **f** is shown in red and **f** compounded three times in blue.



So, on the following entry:

```
5
0 2 4 3 2 1
1
```

your code will have to output 2, while on the following:

```
5
0 2 4 3 2 1
3
```

he will have to output back 5.

You can download sample input and output data files to work locally by clicking on the link at the bottom of the French version of the question.

Téléchargez des fichiers d'exemple ainsi qu'un modèle de code pour travailler localement.