Task: PHR
Pigeon Hit Rate

AACPP SuSe 2024 Round 6 Memory: 64MiB (Java: 256MiB) 2024.07.09 – 2024.07.23

Byteman was called back to evaluate the progress on the Experimental Assault Pigeon programme. The Task Force was running thousands of test runs in the previous weeks. Or was it hundreds? Dozens?

Turns out the IT Task Force does not know quite enough about information theory – they have been storing the results of batches of test runs from various test sites as the *success rates*. What Byteman is interested in, however, is the actual number of successful test runs compared to the total attempts. Additionally, the precision of the results is not standardised across test sites!

In other words, the test site i conducted t_i XAP tests, and s_i of them were successful. As input, Byteman has only access to $\frac{s_i}{t_i}$ represented as a fixed-point number in the range $[0, 1]$ with at most 10 digits of precision, rounded in the usual manner – $0-4$ down, $5-9$ up. For example, a test site with 7 successes and 25 tries would report a success rate of 0.28, while a test site with 6 successes and 9 tries might have reported 0.6667. It can be assumed the actual t_i and s_i values used to report the ratios are valid, i.e. $s_i \leq t_i$ and $t_i > 0$.

Naturally, there are many possible values of t_i and s_i that can result in a given ratio. Byteman wants at least a lower-bound on the test site's efficiency – the *minimal* possible t_i value.

Input

In the first line of standard input there in one integer n – the number of test sites that provided the data.

The next n lines contain a single fixed-point number r_i – the success rate reported by site i. Numbers are always in the range $[0, 1]$, they always include the decimal point, and have at most 10 digits of precision. They are provided with exactly as many digits as their precision. In particular, 0.1230 means that the actual fraction is in the range $[0.12295, 0.12305)$.

Output

Your program should write n lines to standard output. The i-th line should contain the minimal number of tests for a test site that could result in them reporting the success rate r_i.

Example

For the input:

```
5
0.1000
0.153333333
0.280000000
0.52
0.105
```

the correct output is:

```
10
150
25
```

Analysing test site 2 we come to the conclusion that they had to conduct at least 150 tests. If they reported 23 successes, the ratio would indeed be $0.15(3)$. No smaller t_2 value gives this answer.

For $r_4 = 0.52$, a natural answer would be 25, as $\frac{13}{25}$ is indeed exactly 0.52. However, $\frac{11}{21}$ is approximately 0.523, which rounds down to 0.52, so 21 is the correct answer.

Similarly, $\frac{2}{19} \approx 0.1052$, so t_5 is 19 even though $\frac{21}{200}$ would give exactly 0.105.

```
21
19
```

Additional examples

The following initial tests are also available:

- `0b` – sample for Subtask 1, $n = 10$, precision 2 ratios from 0.0 to 0.9;
- `0c` – small sample for Subtask 3, $n = 12$, max precision 10;
- `0d` – sample for Subtask 2, $n = 10^3$, all ratios of the form $0.\text{xxx}0$;
- `0e` – big test, $n = 10^6$, all ratios from 0.000000001 to 0.001000000.

Limits

Your solution will be evaluated on a number of hidden test cases divided into groups. Points for a group are awarded if and only if the submission returns the correct answer for each of the tests in the group within the allotted time limit. These groups are organised into subtasks with the following limits and points awarded.

Subtask	Limits	Points
1.	$1 \leq n \leq 10^3$, max precision is 4	2
2.	$1 \leq n \leq 10^3$, max precision is 7	2
3.	$2 \leq n \leq 10^6$, max precision is 10	6