A. Colorful Stones (Simplified Edition)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a sequence of colorful stones. The color of each stone is one of red, green, or blue. You are given a string s. The i-th (1-based) character of s represents the color of the i-th stone. If the character is "R", "G", or "B", the color of the corresponding stone is red, green, or blue, respectively.

Initially Squirrel Liss is standing on the first stone. You perform instructions one or more times.

Each instruction is one of the three types: "`RED`", "`GREEN`", or "`BLUE`". After an instruction c, if Liss is standing on a stone whose colors is c, Liss will move one stone forward, else she will not move.

You are given a string t. The number of instructions is equal to the length of t, and the i-th character of t represents the i-th instruction.

Calculate the final position of Liss (the number of the stone she is going to stand on in the end) after performing all the instructions, and print its 1-based position. It is guaranteed that Liss don't move out of the sequence.

Input

The input contains two lines. The first line contains the string s ($1 \le |s| \le 50$). The second line contains the string t ($1 \le |t| \le 50$). The characters of each string will be one of "R", "G", or "B". It is guaranteed that Liss don't move out of the sequence.

Output

Print the final 1-based position of Liss in a single line.

Examples

input
```
RGB
RRR
```
output
```
2
```

input
```
RRRBGBRBBB
BBBRR
```
output
```
3
```

input
```
BRRBGBRGRBGRGRRGGBGBGBRGBRGRGGGRBRRRBRBBBGRRRGGBBB
BBRBGGRGRGBBBRBGRBRBBBBRBRRRBGBBGBBRRBBGGRBRRBRGRB
```
output
```
15
```

B. Roadside Trees (Simplified Edition)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Squirrel Liss loves nuts. There are n trees (numbered 1 to n from west to east) along a street and there is a delicious nut on the top of each tree. The height of the tree i is h_i. Liss wants to eat all nuts.

Now Liss is on the root of the tree with the number 1. In one second Liss can perform one of the following actions:

- Walk up or down one unit on a tree.
- Eat a nut on the top of the current tree.
- Jump to the next tree. In this action the height of Liss doesn't change. More formally, when Liss is at height h of the tree i ($1 \le i \le n - 1$), she jumps to height h of the tree $i + 1$. This action can't be performed if $h > h_{i+1}$.

Compute the minimal time (in seconds) required to eat all nuts.

Input

The first line contains an integer n ($1 \le n \le 10^5$) — the number of trees.

Next n lines contains the height of trees: i-th line contains an integer h_i ($1 \le h_i \le 10^4$) — the height of the tree with the number i.

Output

Print a single integer — the minimal time required to eat all nuts in seconds.

Examples

input

```
2
1
2
```

output

```
5
```

input

```
5
2
1
2
1
1
```

output

```
14
```

C. Escape from Stones

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Squirrel Liss lived in a forest peacefully, but unexpected trouble happens. Stones fall from a mountain. Initially Squirrel Liss occupies an interval $[0, 1]$. Next, n stones will fall and Liss will escape from the stones. The stones are numbered from 1 to n in order.

The stones always fall to the center of Liss's interval. When Liss occupies the interval $[k - d, k + d]$ and a stone falls to k, she will escape to the left or to the right. If she escapes to the left, her new interval will be $[k - d, k]$. If she escapes to the right, her new interval will be $[k, k + d]$.

You are given a string s of length n. If the i-th character of s is "`l`" or "`r`", when the i-th stone falls Liss will escape to the left or to the right, respectively. Find the sequence of stones' numbers from left to right after all the n stones falls.

Input

The input consists of only one line. The only line contains the string s ($1 \le |s| \le 10^6$). Each character in s will be either "`l`" or "`r`".

Output

Output n lines — on the i-th line you should print the i-th stone's number from the left.

Examples

input

```
llrlr
```

output

```
3
5
4
2
1
```

input

```
rrlll
```

output

```
1
2
5
4
3
```

input

```
lrlrr
```

output

```
2
4
5
3
1
```

Note

In the first example, the positions of stones 1, 2, 3, 4, 5 will be $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{3}{16}, \frac{5}{32}$, respectively. So you should print the sequence: 3, 5, 4, 2, 1.

D. Good Sequences

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Squirrel Liss is interested in sequences. She also has preferences of integers. She thinks n integers $a_1, a_2, \ldots, a_n$ are *good*.

Now she is interested in good sequences. A sequence $x_1, x_2, \ldots, x_k$ is called *good* if it satisfies the following three conditions:

- The sequence is strictly increasing, i.e. $x_i < x_{i+1}$ for each i $(1 \le i \le k - 1)$.
- No two adjacent elements are coprime, i.e. $gcd(x_i, x_{i+1}) > 1$ for each i $(1 \le i \le k - 1)$ (where $gcd(p, q)$ denotes the greatest common divisor of the integers p and q).
- All elements of the sequence are good integers.

Find the length of the longest good sequence.

Input

The input consists of two lines. The first line contains a single integer n ($1 \le n \le 10^5$) — the number of good integers. The second line contains a single-space separated list of good integers $a_1, a_2, \ldots, a_n$ in strictly increasing order ($1 \le a_i \le 10^5$; $a_i < a_{i+1}$).

Output

Print a single integer — the length of the longest good sequence.

Examples

input

```
5
2 3 4 6 9
```

output

```
4
```

input

```
9
1 2 3 5 6 7 8 9 10
```

output

```
4
```

Note

In the first example, the following sequences are examples of good sequences: [2; 4; 6; 9], [2; 4; 6], [3; 9], [6]. The length of the longest good sequence is 4.

E. Choosing Balls

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n balls. They are arranged in a row. Each ball has a color (for convenience an integer) and an integer value. The color of the i-th ball is c_i and the value of the i-th ball is v_i.

Squirrel Liss chooses some balls and makes a new sequence without changing the relative order of the balls. She wants to maximize the value of this sequence.

The value of the sequence is defined as the sum of following values for each ball (where a and b are given constants):

- If the ball is not in the beginning of the sequence and the color of the ball is same as previous ball's color, add (the value of the ball) $\times$ a.
- Otherwise, add (the value of the ball) $\times$ b.

You are given q queries. Each query contains two integers a_i and b_i. For each query find the maximal value of the sequence she can make when $a = a_i$ and $b = b_i$.

Note that the new sequence can be **empty**, and the value of an empty sequence is defined as zero.

Input

The first line contains two integers n and q ($1 \le n \le 10^5$; $1 \le q \le 500$). The second line contains n integers: $v_1, v_2, ..., v_n$ ($|v_i| \le 10^5$). The third line contains n integers: $c_1, c_2, ..., c_n$ ($1 \le c_i \le n$).

The following q lines contain the values of the constants a and b for queries. The i-th of these lines contains two integers a_i and b_i ($|a_i|, |b_i| \le 10^5$).

In each line integers are separated by single spaces.

Output

For each query, output a line containing an integer — the answer to the query. The i-th line contains the answer to the i-th query in the input order.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input

```
6 3
1 -2 3 4 0 -1
1 2 1 2 1 1
5 1
-2 1
1 0
```

output

```
20
9
4
```

input

```
4 1
-3 6 -1 2
1 2 3 1
1 -1
```

output

```
5
```

Note

In the first example, to achieve the maximal value:

- In the first query, you should select 1st, 3rd, and 4th ball.
- In the second query, you should select 3rd, 4th, 5th and 6th ball.
- In the third query, you should select 2nd and 4th ball.

Note that there may be other ways to achieve the maximal value.