Codeforces Round #162 (Div. 1)

A. 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.

Sample test(s)

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.

B. 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 $a_1, a_2, ..., a_n$ are *good*.

Now she is interested in good sequences. A sequence $x_1, x_2, ..., x_k$ 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 $a_1, a_2, ..., a_n$ 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.

Sample test(s)

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.

C. 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.

Sample test(s)

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.

D. Colorful Stones

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

There are two sequences of colorful stones. The color of each stone is one of red, green, or blue. You are given two strings s and t. The i-th (1-based) character of s represents the color of the i-th stone of the first sequence. Similarly, the i-th (1-based) character of t represents the color of the i-th stone of the second sequence. 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 of the first sequence and Cat Vasya is standing on the first stone of the second sequence. You can perform the following instructions zero or more times.

Each instruction is one of the three types: "RED", "GREEN", or "BLUE". After an instruction c, the animals standing on stones whose colors are c will move one stone forward. For example, if you perform an instruction «RED», the animals standing on red stones will move one stone forward. You are not allowed to perform instructions that lead some animals out of the sequences. In other words, if some animals are standing on the last stones, you can't perform the instructions of the colors of those stones.

A pair of positions (position of Liss, position of Vasya) is called a state. A state is called *reachable* if the state is reachable by performing instructions zero or more times from the initial state (1, 1). Calculate the number of distinct reachable states.

Input

The input contains two lines. The first line contains the string s ($1 \le |s| \le 10^6$). The second line contains the string t ($1 \le |t| \le 10^6$). The characters of each string will be one of "R", "G", or "B".

Output

Print the number of distinct reachable states in a single line.

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.

Sample test(s)

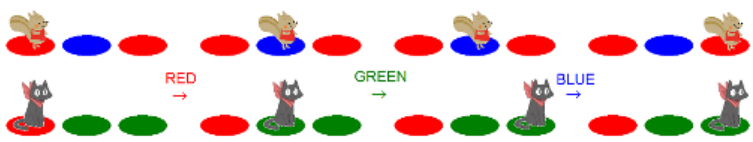
input
RBR RGG
output
5

input
RGBB BRRBRR
output
19

input
RRRRRRRRRR RRRRRRRR
output
8

Note

In the first example, there are five reachable states: (1, 1), (2, 2), (2, 3), (3, 2), and (3, 3). For example, the state (3, 3) is reachable because if you perform instructions "RED", "GREEN", and "BLUE" in this order from the initial state, the state will be (3, 3). The following picture shows how the instructions work in this case.



E. Roadside Trees

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

Squirrel Liss loves nuts. Liss asks you to plant some nut trees.

There are n positions (numbered 1 to n from west to east) to plant a tree along a street. Trees grow one meter per month. At the beginning of each month you should process one query. The query is one of the following types:

1. Plant a tree of height h at position p.
2. Cut down the x-th existent (not cut) tree from the west (where x is 1-indexed). When we cut the tree it drops down and takes all the available place at the position where it has stood. So no tree can be planted at this position anymore.

After processing each query, you should print the length of the longest increasing subsequence. A subset of existent trees is called an *increasing subsequence* if the height of the trees in the set is strictly increasing from west to east (for example, the westmost tree in the set must be the shortest in the set). The length of the increasing subsequence is the number of trees in it.

Note that Liss don't like the trees with the same heights, so it is guaranteed that at any time no two trees have the exactly same heights.

Input

The first line contains two integers: n and m ($1 \le n \le 10^5$; $1 \le m \le 2 \cdot 10^5$) — the number of positions and the number of queries.

Next m lines contains the information of queries by following formats:

- If the i-th query is type 1, the i-th line contains three integers: 1, p_i, and h_i ($1 \le p_i \le n$, $1 \le h_i \le 10$), where p_i is the position of the new tree and h_i is the initial height of the new tree.
- If the i-th query is type 2, the i-th line contains two integers: 2 and x_i ($1 \le x_i \le 10$), where the x_i is the index of the tree we want to cut.

The input is guaranteed to be correct, i.e.,

- For type 1 queries, p_i will be pairwise distinct.
- For type 2 queries, x_i will be less than or equal to the current number of trees.
- At any time no two trees have the exactly same heights.

In each line integers are separated by single spaces.

Output

Print m integers — the length of the longest increasing subsequence after each query. Separate the numbers by whitespaces.

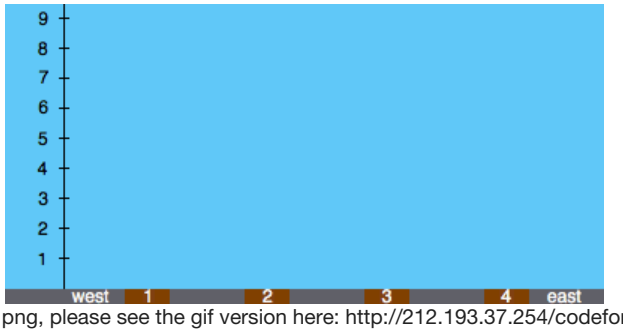
Sample test(s)

input
4 6 1 1 1 1 4 4 1 3 4 2 2 1 2 8 2 3

output
1 2 3 2 2 2

Note

States of street after each query you can see on the following animation:



If your browser doesn't support animation png, please see the gif version here: http://212.193.37.254/codeforces/images/162/roadtree.gif