VK Cup 2012 Round 2

A. Substring and Subsequence

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

One day Polycarpus got hold of two non-empty strings s and t, consisting of lowercase Latin letters. Polycarpus is quite good with strings, so he immediately wondered, how many different pairs of "x y" are there, such that x is a substring of string s, y is a subsequence of string t, and the content of x and y is the same. Two pairs are considered different, if they contain different substrings of string s or different subsequences of string t. Read the whole statement to understand the definition of different substrings and subsequences.

The *length* of string s is the number of characters in it. If we denote the length of the string s as $|s|$, we can write the string as $s = s_1 s_2 \ldots s_{|s|}$.

A *substring* of s is a non-empty string $x = s[a \ldots b] = s_a s_{a+1} \ldots s_b$ $(1 \le a \le b \le |s|)$. For example, "`code`" and "`force`" are substrings or "`codeforces`", while "`coders`" is not. Two substrings $s[a \ldots b]$ and $s[c \ldots d]$ are considered to be *different* if $a \ne c$ or $b \ne d$. For example, if s="`codeforces`", $s[2 \ldots 2]$ and $s[6 \ldots 6]$ are different, though their content is the same.

A *subsequence* of s is a non-empty string $y = s[p_1 p_2 \ldots p_{|y|}] = s_{p_1} s_{p_2} \ldots s_{p_{|y|}}$ $(1 \le p_1 < p_2 < \ldots < p_{|y|} \le |s|)$. For example, "`coders`" is a subsequence of "`codeforces`". Two subsequences $u = s[p_1 p_2 \ldots p_{|u|}]$ and $v = s[q_1 q_2 \ldots q_{|v|}]$ are considered *different* if the sequences p and q are different.

Input

The input consists of two lines. The first of them contains s $(1 \le |s| \le 5000)$, and the second one contains t $(1 \le |t| \le 5000)$. Both strings consist of lowercase Latin letters.

Output

Print a single number — the number of different pairs "x y" such that x is a substring of string s, y is a subsequence of string t, and the content of x and y is the same. As the answer can be rather large, print it modulo 1000000007 $(10^9 + 7)$.

Sample test(s)

input
aa aa
output
5

input
codeforces forceofcode
output
60

Note

Let's write down all pairs "x y" that form the answer in the first sample: "$s[1 \ldots 1]$ $t[1]$", "$s[2 \ldots 2]$ $t[1]$", "$s[1 \ldots 1]$ $t[2]$","$s[2 \ldots 2]$ $t[2]$", "$s[1 \ldots 2]$ $t[1\ 2]$".

B. Lemmings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

As you know, lemmings like jumping. For the next spectacular group jump n lemmings gathered near a high rock with k comfortable ledges on it. The first ledge is situated at the height of h meters, the second one is at the height of $2h$ meters, and so on (the i-th ledge is at the height of $i \cdot h$ meters). The lemmings are going to jump at sunset, and there's not much time left.

Each lemming is characterized by its climbing speed of v_i meters per minute and its weight m_i. This means that the i-th lemming can climb to the j-th ledge in $\frac{j \cdot h}{v_i}$ minutes.

To make the jump beautiful, heavier lemmings should jump from higher ledges: if a lemming of weight m_i jumps from ledge i, and a lemming of weight m_j jumps from ledge j (for $i < j$), then the inequation $m_i \le m_j$ should be fulfilled.

Since there are n lemmings and only k ledges ($k \le n$), the k lemmings that will take part in the jump need to be chosen. The chosen lemmings should be distributed on the ledges from 1 to k, one lemming per ledge. The lemmings are to be arranged in the order of non-decreasing weight with the increasing height of the ledge. In addition, each lemming should have enough time to get to his ledge, that is, the time of his climb should not exceed t minutes. The lemmings climb to their ledges all at the same time and they do not interfere with each other.

Find the way to arrange the lemmings' jump so that time t is minimized.

Input

The first line contains space-separated integers n, k and h ($1 \le k \le n \le 10^5$, $1 \le h \le 10^4$) — the total number of lemmings, the number of ledges and the distance between adjacent ledges.

The second line contains n space-separated integers $m_1, m_2, ..., m_n$ ($1 \le m_i \le 10^9$), where m_i is the weight of i-th lemming.

The third line contains n space-separated integers $v_1, v_2, ..., v_n$ ($1 \le v_i \le 10^9$), where v_i is the speed of i-th lemming.

Output

Print k different numbers from 1 to n — the numbers of the lemmings who go to ledges at heights $h, 2h, ..., kh$, correspondingly, if the jump is organized in an optimal way. If there are multiple ways to select the lemmings, pick any of them.

Sample test(s)

input
5 3 2 1 2 3 2 1 1 2 1 2 10
output
5 2 4

input
5 3 10 3 4 3 2 1 5 4 3 2 1
output
4 3 1

Note

Let's consider the first sample case. The fifth lemming (speed 10) gets to the ledge at height 2 in $\frac{1}{5}$ minutes; the second lemming (speed 2) gets to the ledge at height 4 in 2 minutes; the fourth lemming (speed 2) gets to the ledge at height 6 in 3 minutes. All lemmings manage to occupy their positions in 3 minutes.

C. Conveyor

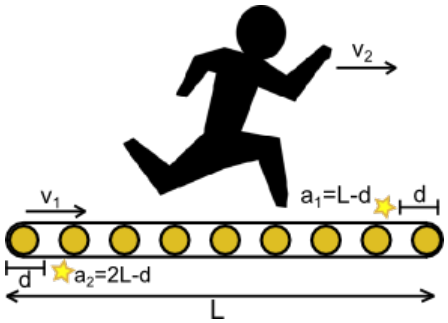
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Anton came to a chocolate factory. There he found a working conveyor and decided to run on it from the beginning to the end.

The conveyor is a looped belt with a total length of $2l$ meters, of which l meters are located on the surface and are arranged in a straight line. The part of the belt which turns at any moment (the part which emerges from under the floor to the surface and returns from the surface under the floor) is assumed to be negligibly short.

The belt is moving uniformly at speed v_1 meters per second. Anton will be moving on it in the same direction at the constant speed of v_2 meters per second, so his speed relatively to the floor will be $v_1 + v_2$ meters per second. Anton will neither stop nor change the speed or the direction of movement.

Here and there there are chocolates stuck to the belt (n chocolates). They move together with the belt, and do not come off it. Anton is keen on the chocolates, but he is more keen to move forward. So he will pick up all the chocolates he will pass by, but nothing more. If a chocolate is at the beginning of the belt at the moment when Anton starts running, he will take it, and if a chocolate is at the end of the belt at the moment when Anton comes off the belt, he will leave it.



The figure shows an example with two chocolates. One is located in the position $a_1 = l - d$, and is now on the top half of the belt, the second one is in the position $a_2 = 2l - d$, and is now on the bottom half of the belt.

You are given the positions of the chocolates relative to the initial start position of the belt $0 \le a_1 < a_2 < ... < a_n < 2l$. The positions on the belt from 0 to l correspond to the top, and from l to $2l$ — to the the bottom half of the belt (see example). All coordinates are given in meters.

Anton begins to run along the belt at a random moment of time. This means that all possible positions of the belt at the moment he starts running are equiprobable. For each i from 0 to n calculate the probability that Anton will pick up exactly i chocolates.

Input

The first line contains space-separated integers n, l, v_1 and v_2 ($1 \le n \le 10^5$, $1 \le l, v_1, v_2 \le 10^9$) — the number of the chocolates, the length of the conveyor's visible part, the conveyor's speed and Anton's speed.

The second line contains a sequence of space-separated integers $a_1, a_2, ..., a_n$ ($0 \le a_1 < a_2 < ... < a_n < 2l$) — the coordinates of the chocolates.

Output

Print $n + 1$ numbers (one per line): the probabilities that Anton picks up exactly i chocolates, for each i from 0 (the first line) to n (the last line). The answer will be considered correct if each number will have absolute or relative error of at most than 10^{-9}.

Sample test(s)

input
1 1 1 1 0
output
0.7500000000000000000 0.2500000000000000000

input
2 3 1 2 2 5
output
0.3333333333333331000 0.6666666666666663000 0.0000000000000000000

Note

In the first sample test Anton can pick up a chocolate if by the moment he starts running its coordinate is less than 0.5; but if by the moment the boy starts running the chocolate's coordinate is greater than or equal to 0.5, then Anton won't be able to pick it up. As all positions of the belt are equiprobable, the probability of picking up the chocolate equals $\frac{0.5}{2} = 0.25$, and the probability of not picking it up equals $\frac{1.5}{2} = 0.75$.

D. Large Refrigerator

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

Vasya wants to buy a new refrigerator. He believes that a refrigerator should be a rectangular parallelepiped with integer edge lengths. Vasya calculated that for daily use he will need a refrigerator with volume of at least V. Moreover, Vasya is a minimalist by nature, so the volume should be no more than V, either — why take up extra space in the apartment? Having made up his mind about the volume of the refrigerator, Vasya faced a new challenge — for a fixed volume of V the refrigerator must have the minimum surface area so that it is easier to clean.

The volume and the surface area of a refrigerator with edges a, b, c are equal to $V = abc$ and $S = 2(ab + bc + ca)$, correspondingly.

Given the volume V, help Vasya find the integer lengths for the refrigerator's edges a, b, c so that the refrigerator's volume equals V and its surface area S is minimized.

Input

The first line contains a single integer t ($1 \le t \le 500$) — the number of data sets.

The description of t data sets follows. Each set consists of a single integer V ($2 \le V \le 10^{18}$), given by its factorization as follows.

Let $V = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$, where p_i are different prime numbers and a_i are positive integer powers.

Then the first line describing a data set contains a single positive integer k — the number of different prime divisors of V. Next k lines contain prime numbers p_i and their powers a_i, separated by spaces. All p_i are different, all $a_i > 0$.

Output

Print t lines, on the i-th line print the answer to the i-th data set as four space-separated integers: the minimum possible surface area S and the corresponding edge lengths a, b, c. If there are multiple variants of the lengths of edges that give the minimum area, you are allowed to print any of them. You can print the lengths of the fridge's edges in any order.

Sample test(s)

input
3 1 2 3 1 17 1 3 3 1 2 3 5 1

output
24 2 2 2 70 1 1 17 148 4 6 5

Note

In the first data set of the sample the fridge's volume $V = 2^3 = 8$, and the minimum surface area will be produced by the edges of equal length.

In the second data set the volume $V = 17$, and it can be produced by only one set of integer lengths.

E. e-Government

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The best programmers of Embezzland compete to develop a part of the project called "e-Government" — the system of automated statistic collecting and press analysis.

We know that any of the k citizens can become a member of the Embezzland government. The citizens' surnames are $a_1, a_2, ..., a_k$. All surnames are different. Initially all k citizens from this list are members of the government. The system should support the following options:

- Include citizen a_i to the government.
- Exclude citizen a_i from the government.
- Given a newspaper article text, calculate how politicized it is. To do this, for every active government member the system counts the number of times his surname occurs in the text as a substring. All occurrences are taken into consideration, including the intersecting ones. The degree of politicization of a text is defined as the sum of these values for all active government members.

Implement this system.

Input

The first line contains space-separated integers n and k ($1 \le n, k \le 10^5$) — the number of queries to the system and the number of potential government members.

Next k lines contain the surnames $a_1, a_2, ..., a_k$, one per line. All surnames are pairwise different.

Next n lines contain queries to the system, one per line. Each query consists of a character that determines an operation and the operation argument, written consecutively without a space.

Operation "include in the government" corresponds to the character "+", operation "exclude" corresponds to "-". An argument of those operations is an integer between 1 and k — the index of the citizen involved in the operation. Any citizen can be included and excluded from the government an arbitrary number of times in any order. Including in the government a citizen who is already there or excluding the citizen who isn't there changes nothing.

The operation "calculate politicization" corresponds to character "?". Its argument is a text.

All strings — surnames and texts — are non-empty sequences of lowercase Latin letters. The total length of all surnames doesn't exceed 10^6, the total length of all texts doesn't exceed 10^6.

Output

For any "calculate politicization" operation print on a separate line the degree of the politicization of the given text. Print nothing for other operations.

Sample test(s)

input
7 3 a aa ab ?aaab -2 ?aaab -3 ?aaab +2 ?aabbaa

output
6 4 3 6