



VK Cup 2012 Round 2 (Unofficial Div. 2 Edition)

A. Chores

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

Petya and Vasya are brothers. Today is a special day for them as their parents left them home alone and commissioned them to do n chores. Each chore is characterized by a single parameter — its complexity. The complexity of the i -th chore equals h_i .

As Petya is older, he wants to take the chores with complexity larger than some value X ($h_i > X$) to leave to Vasya the chores with complexity less than or equal to X ($h_i \leq X$). The brothers have already decided that Petya will do exactly a chores and Vasya will do exactly b chores ($a + b = n$).

In how many ways can they choose an integer X so that Petya got exactly a chores and Vasya got exactly b chores?

Input

The first input line contains three integers n , a and b ($2 \leq n \leq 2000$; $a, b \geq 1$; $a + b = n$) — the total number of chores, the number of Petya's chores and the number of Vasya's chores.

The next line contains a sequence of integers h_1, h_2, \dots, h_n ($1 \leq h_i \leq 10^9$), h_i is the complexity of the i -th chore. The numbers in the given sequence are not necessarily different.

All numbers on the lines are separated by single spaces.

Output

Print the required number of ways to choose an integer value of X . If there are no such ways, print 0.

Examples

input
5 2 3 6 2 3 100 1
output
3

input
7 3 4 1 1 9 1 1 1 1
output
0

Note

In the first sample the possible values of X are 3, 4 or 5.

In the second sample it is impossible to find such X , that Petya got 3 chores and Vasya got 4.

B. Replacing Digits

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

You are given an integer a that consists of n digits. You are also given a sequence of digits S of length m . The digit in position j ($1 \leq j \leq m$) of sequence S means that you can choose an arbitrary position i ($1 \leq i \leq n$) in a and replace the digit in the chosen position i with S_j . Each element in the sequence S can participate in no more than one replacing operation.

Your task is to perform such sequence of replacements, that the given number a gets maximum value. You are allowed to use not all elements from S .

Input

The first line contains positive integer a . Its length n is positive and doesn't exceed 10^5 . The second line contains sequence of digits S . Its length m is positive and doesn't exceed 10^5 . The digits in the sequence S are written consecutively without any separators.

The given number a doesn't contain leading zeroes.

Output

Print the maximum value that can be obtained from a after a series of replacements. You are allowed to use not all elements from S . The printed number shouldn't contain any leading zeroes.

Examples

input
1024 010
output
1124

input
987 1234567
output
987

C. 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 $S = S_1S_2\ldots S_{|S|}$.

A *substring* of S is a non-empty string $X = S[a\ldots b] = S_aS_{a+1}\ldots S_b$ ($1 \leq a \leq b \leq |S|$). For example, "code" and "force" are substrings of "codeforces", while "coders" is not. Two substrings $S[a\ldots b]$ and $S[c\ldots d]$ are considered to be *different* if $a \neq c$ or $b \neq d$. For example, if $S = \text{"codeforces"}$, $S[2\ldots 2]$ and $S[6\ldots 6]$ are different, though their content is the same.

A *subsequence* of S is a non-empty string $y = S[p_1p_2\ldots p_{|y|}] = S_{p_1}S_{p_2}\ldots S_{p_{|y|}}$ ($1 \leq p_1 < p_2 < \ldots < p_{|y|} \leq |S|$). For example, "coders" is a subsequence of "codeforces". Two subsequences $u = S[p_1p_2\ldots p_{|u|}]$ and $v = S[q_1q_2\ldots q_{|v|}]$ are considered *different* if the sequences p and q are different.

Input

The input consists of two lines. The first of them contains S ($1 \leq |S| \leq 5000$), and the second one contains t ($1 \leq |t| \leq 5000$). 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 ($10^9 + 7$).

Examples

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: " $s[1\ldots 1]\ t[1]$ ", " $s[2\ldots 2]\ t[1]$ ", " $s[1\ldots 1]\ t[2]$ ", " $s[2\ldots 2]\ t[2]$ ", " $s[1\ldots 2]\ t[1\ 2]$ ".

D. 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 $m_i \leq m_j$ should be fulfilled.

Since there are n lemmings and only k ledges ($k \leq n$), 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 ($1 \leq k \leq n \leq 10^5$, $1 \leq h \leq 10^4$) — 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, \dots, m_n ($1 \leq m_i \leq 10^9$), where m_i is the weight of i -th lemming.

The third line contains n space-separated integers v_1, v_2, \dots, v_n ($1 \leq v_i \leq 10^9$), 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 $h, 2h, \dots, kh$, correspondingly, if the jump is organized in an optimal way. If there are multiple ways to select the lemmings, pick any of them.

Examples

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.

E. 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 $0 \leq a_1 < a_2 < \dots < a_n < 2l$. The positions on the belt from 0 to l correspond to the top, and from l to $2l$ — to 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 ($1 \leq n \leq 10^5, 1 \leq l, v_1, v_2 \leq 10^9$) — 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, \dots, a_n ($0 \leq a_1 < a_2 < \dots < a_n < 2l$) — 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} .

Examples

input
1 1 1 1 0
output
0.75000000000000000000 0.25000000000000000000

input
2 3 1 2 2 5
output
0.33333333333333331000 0.66666666666666663000 0.00000000000000000000

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 $\frac{1.5}{2} = 0.75$.

