Problem A. Almost Longest Increasing Subsequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

This is an interactive problem.

Suppose you are given a permutation a of n numbers from 1 to n. In a usual task you would have to find its *longest increasing subsequence*: indices $i_1, \ldots, i_k$ such that $i_1 < \ldots < i_k$ and $a_{i_1} < \ldots < a_{i_k}$, where k is maximum possible.

However, this time the task is different: the numbers are given **online**, so you must decide whether you take the number or not before you receive the next number. You also know that the input permutation is **random**, that is, selected uniformly at random from the set of all $n!$ possible permutations. Under these requirements it is impossible to surely find the longest increasing subsequence, but you have to just do good enough. Formally, let k be the length of the longest increasing subsequence of the given permutation. Then, finding any increasing subsequence of length at least $0.65k$ will suffice.

Interaction Protocol

First, interactor sends a single number n ($n = 10^5$) which is the length of the permutation. Next, interactor sends n numbers one by one, each on a separate line. After receiving each number you should print one integer to the standard output, which is 1 if you take the given number and 0 otherwise.

The sequence given is a permutation of $1, \ldots, n$: all elements are distinct and each of then is from 1 to n.

Every number selected by participant (except for the first one) should be greater than the previous one.

In the sample case $n = 5$. Any solution which follows the output format passes this testcase.

Example

standard input	standard output
5 2 1 5 4 3 	 1 0 1 0 0

Note

The sample test is used only to illustrate the interaction format and is not included in the testset. Each test from the testset has $n = 10^5$.

In this problem, technically, a *random permutation* is an array of $1, \ldots, n$ shuffled with some pseudo-random number generator.

Problem B. Bitwise Queries

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You are given an array a of size n and you need to perform m queries on it. There are three types of queries:

1. `&` l r x: change a_i to (a_i `AND` x) for all $i = l, l+1, \ldots, r$;
2. `|` l r x: change a_i to (a_i `OR` x) for all $i = l, l+1, \ldots, r$;
3. `?` l r: find the minimal value among $a_l, a_{l+1}, \ldots, a_r$.

Output the answers for all queries of the third type.

Input

The first line contains one integer n ($1 \le n \le 5 \cdot 10^5$) — the size of the array.

The second line contains n space-separated integers a_i ($0 \le a_i < 2^{30}$) — the elements of the array.

The third line contains one integer m ($1 \le m \le 2 \cdot 10^5$) — the number of queries.

Next m lines contain descriptions of queries in the format described above. For all queries $1 \le l \le r \le n$, for queries of the first and second types $0 \le x < 2^{30}$.

Output

For each query of the third type, print the answer on a separate line.

Example

standard input	standard output
5 1 2 3 4 5 4 & 1 2 6 | 3 5 4 ? 1 2 ? 3 5	0 4

Problem C. Cocktails

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

A party is coming tonight, and drinks are not ready yet! You are going to prepare a cocktail for your friends. The cocktail consists of n different ingredients. There are n jars in front of you, numbered from left to right starting from 1. Each of the jars contains a single ingredient. You have to blend each ingredient in its separate jar.

Blending the contents of i-th jar by hand takes a_i seconds. Also, you have a very powerful blender which can blend the contents in any k subsequent jars in B seconds (the blender can be applied any number of times). Finally, you can swap any two jars in C seconds (the two jars don't have to be adjacent). It is allowed to blend the contents of any jar more than once.

What is the smallest possible time to blend the ingredients in all n jars? The final order of jars does not matter as long as they are all blended.

Input

In the first line of input there are four space-separated integers n, k, B, C ($1 \le k \le n \le 500$, $1 \le B, C \le 10\,000$) — the number of jars, the reach of the blender, the time needed to use the blender, and the time needed to swap two jars respectively.

In the second line there are n space-separated integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10\,000$), where a_i is the time needed to manually blend the contents of i-th jar.

Output

Print a single integer — the smallest time needed to blend the contents in all jars.

Example

standard input	standard output
5 2 10 3 10 1 20 2 3	19

Problem D. Downhill

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A climber stands on top of a very steep mountain of height H meters. He has a rope and wants to get down to the foothills. There are n ledges on a mountain, i-th of them is a_i meters over the foothills. A climber can stand on every ledge or on the top, but cannot stand anywhere else on the mountain.

The climber uses his rope to go from one ledge to another. He can tie his rope to the ledge, go down and then cut it being on the ledge. In this case the cut part of the rope stays hanging forever because there is no possibility to untie it after getting down, but the remaining piece of rope stays with the climber and he can use it further.

While hanging on a rope, the climber can cut it in the place he holds it and make a loop on its end, then push the remaining rope through the loop (this can also be done while standing on the ledge). Then he can go down by the second rope, folded in half, and pull it back after descending. Please refer to the Notes section for better understanding.

What is the minimum length of the rope required for the climber to get to the foothills from the top? You may assume that the rope is infinitely thin and can be cut or tied with negligible (zero) loss of the material.

Input

First line of input contains two space-separated integers n and H ($0 \le n \le 10^6$, $0 \le H \le 10^9$) — number of ledges and height of the mountain in meters.

The next line contains n space-separated integers $a_1, \ldots, a_n$ — heights of ledges ($H > a_1 > \ldots > a_n > 0$).

Output

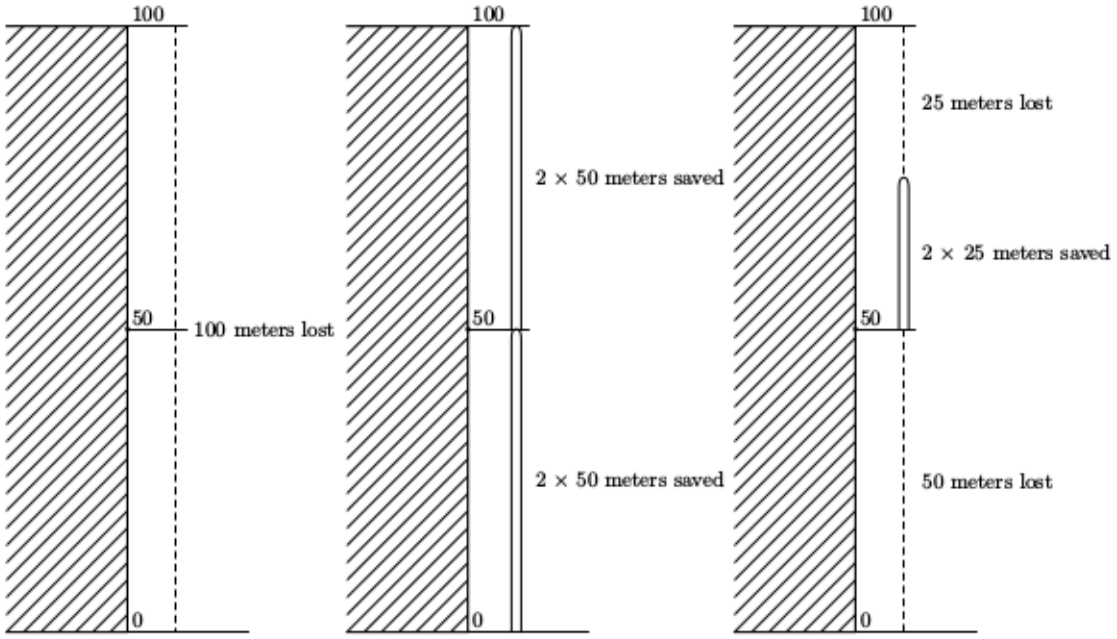
Print one real number — required rope length with absolute or relative error not exceeding 10^{-6}.

Examples

standard input	standard output
1 100 50	75
2 3 2 1	1.75

Note

Consider three possible ways to get down for the first sample.



The first way is to simply use 100 meters rope to get from the top to the bottom. The second way is to make a loop at the top, put double rope through the loop, get to the ledge and retrieve the rope. Repeating this to get from the ledge to the ground, we obtain another way to get down with 100 meters rope.

If we combine two approaches, we can do much better and get down with only 75 meters of rope. Let us hang 25 meters from the top on a single rope, make a loop there and double-rope to the ledge. This way we forfeit 25 initial meters, but save 50 meters we used for double-roping. These 50 meters are just enough to single-rope to the ground.

Problem E. Expected LCP

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Consider a sequence s_1, s_2, ..., s_n of n infinite binary strings (that is, consisting only of zeros and ones), where each character of each string is generated uniformly at random independently from others. Denote

$$f(s_1, s_2, \ldots, s_n) = \max_{1 \le i < j \le n} LCP(s_i, s_j),$$

where LCP is the maximum common prefix of two strings. Compute the expected value of $f(s_1, s_2, \ldots, s_n)$.

Input

The only line of the input contains one integer n ($2 \le n \le 10^4$).

Output

Let the answer in the form of an irreducible fraction be P/Q. Then output $P \cdot Q^{-1} \bmod (10^9 + 7)$. It is guaranteed that $Q \bmod (10^9 + 7) \neq 0$.

Examples

standard input	standard output
2	1
3	333333338

Note

Note that the expected value is always finite, that is, $\mathtt{E} f(s_1, \ldots, s_n) < \infty$.

In the second sample the answer is $\frac{7}{3}$.

Problem F. Finite Walking

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given an undirected graph G with n vertices and m edges. Loops and multiple edges are allowed. Each edge has a positive integer a_i assigned to it, and a counter b_i which is initially zero.

Consider the following process on this graph: we choose the starting vertex v_1 arbitrarily, then we go along some edge e_1 starting in v_1 and ending in v_2, then we go along some edge e_2 from v_2 to v_3 and so on. In other words, we follow some path in the graph, which may contain some edges and vertices multiple times. Every time we traverse edge i in any direction, its counter b_i changes to $(b_i + 1) \bmod a_i$. We can stop the process after any number of steps (possibly, zero).

Let us call the array $(b_1, b_2, \ldots, b_m)$ after this process a *configuration array*. Consider all possible processes corresponding to all possible finite paths in G. How many different configuration arrays can they produce? Two configuration arrays are considered different if they differ in at least one element. As the answer can be quite large, give it modulo $10^9 + 7$.

Input

The first line of input contains two space-separated integers n, m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 4 \cdot 10^5$) — number of vertices and number of edges of the graph respectively.

Next m lines contain the description of edges, one per line: i-th edge is described by three integers u_i, v_i, a_i ($1 \le u_i, v_i \le n$, $1 \le a_i \le 10^9$) — the endpoints of the edge and the number assigned to this edge. Vertices are indexed starting from 1.

Note that loops and multiple edges are **allowed**.

Output

In the only line print the number of different configuration arrays modulo $10^9 + 7$.

Example

standard input	standard output
5 5 1 2 2 2 3 2 3 4 2 4 5 2 5 1 1	14

Problem G. Guess the Distribution

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	256 mebibytes

Let $p \in (0; 1)$ be a real number and $n \in [1; 100]$ be a positive integer. Consider $\xi_1, \xi_2, \ldots, \xi_n$ — independent random variables such that $P(\xi_i = 1) = p$ and $P(\xi_i = 0) = 1 - p$. Consider the random variable

$$\theta_n = \frac{\xi_1 + \ldots + \xi_n + u - np}{\sqrt{np(1-p)}},$$

where u is a random variable which is uniformly distributed in $[-\frac{1}{2}, \frac{1}{2}]$ independently from all ξ_i.

You are given p and a sample from the distribution θ_n for some n. You need to determine n.

Input

The first line of input contains one integer T ($T = 30$) — the number of samples.

The second line contains a real number $p \in (0; 1)$ with at most two digits after the decimal point.

Each of next T lines contains the description of the sample. It consists of an integer N ($N = 10^4$) — the size of the sample, and N space-separated real numbers $x_1, x_2, \ldots, x_N$ with at most 10 digits after decimal point, describing the sample of the distribution θ_n for some n.

It is guaranteed that the test case is generated as follows: we choose p and seed s by hand and then choose all n as T random integers from $[1; 100]$ using a pseudo-random number generator with initial seed s.

In the sample test case $T = 2$ and $N = 3$ just to show the format.

Output

For each of T samples print one integer on a separate line — the number $ans_i \in [1; 100]$ which you think defines the distribution θ_n.

Your answer for the whole test case will be considered correct, if the average absolute error does not exceed 5, so that

$$\frac{1}{T}\sum_{i=1}^{T} |n_i - ans_i| \le 5.$$

Example

standard input	standard output
2 0.5 3 0.1 0.05 -0.2 3 0.05 -0.1 0.01	3 7

Problem H. Hash Table

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

In this problem we will deal with a *hash table with open addressing strategy* — an implementation of "set" data structure based on hashing. Naturally, hash table is concerned with *hash values* of elements, which are non-negative integers computed in a certain way based on the contents of the elements.

The hash table data structure operates on an infinite array with 0-based indexation. Each cell of the array can either be empty or contain an element. We will only consider one type of instruction: *add* an element with hash h. While performing this instruction, we try to put current element in cells h, $h+1$, $h+2$, ... in this order. The first empty cell among these is then filled with the element.

We consider the *cost* of the instruction to be the number of occupied cells we skipped while trying to put the element. For example, if an element with hash 5 was immediately put into cell 5, the cost is zero, but if it ended up in cell 10, we must have performed five unsuccessful tries and thus the cost is 5.

Consider a sequence of "add an element with hash h_i" instructions. We are interested in changing the sequence (inserting or omitting instructions) while maintaining the total cost of all instructions performed in this order. The hash table is cleared before every sequence is processed.

Initially the instruction sequence is empty. You have to process two types of queries changing the sequence:

- "+ h p": insert the instruction "add an element with hash h" so that it results on p-th place in the sequence;
- "- q": delete the instruction that was inserted on q-th query.

After performing each query you should print the total cost of all instructions currently in the sequence performed in this order.

Instructions and queries are numbered starting from 1. It is guaranteed that all queries are valid.

Input

First line of input contains a single integer n — the number of queries to the sequence ($1 \le n \le 150\,000$).

Next n lines describe the queries. Each of these lines is either formatted as "+ h p" or "- q", describing an insertion or deletion query according to the rules above.

Each insertion query "+ h p" satisfies $0 \le h \le 10^5$ and $1 \le p \le k + 1$, where k is the size of the sequence before the query.

For each deletion query "- q", it is guaranteed that q-th query was an insertion query and it wasn't deleted as a result of an earlier deletion query.

Output

Print n numbers, each on a separate line: q-th line should contain the total cost of performing the instruction sequence after q-th query.

Example

standard input	standard output
5 + 1 1 + 1 2 + 2 3 - 2 - 1	0 1 2 0 0

Problem I. Immigration

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Peter the Pig is riding a tractor, starting at the origin at moment 0 and moving along the x-axis with speed u. Since the road is infinite and not interesting enough, Peter is looking for entertainment. At moment t_0 he notices some object at point (x_0, y_0) moving with velocity $v_0 = (v_{0x}, v_{0y})$. So, Peter starts to rotate his head always keeping his eyes fixed at the object. From time to time (n times in total), the object changes its velocity. Namely, at moment t_i the object instantly changes its velocity to $v_i = (v_{ix}, v_{iy})$ for $i = 1, 2, \ldots, n$. It is guaranteed that the object never collides with Peter.

Define $f(t)$ as the directed angle (measured in radians) between x-axis and the direction from Peter to the object at moment t. We allow $f(t)$ to differ from the actual angle by a multiple of 2π as long as the function $f(t)$ is continuous. Define the *angular speed* of Peter's head as the derivative $f'(t)$. You need to find the maximum of $|f'(t)|$ from the moment t_0 to the infinity. In other words, you need to find the maximum of the absolute value of angular speed. It is guaranteed that the answer is finite.

Input

The first line contains four integers u, x_0, y_0, n ($1 \le u \le 100$, $|x_0|, |y_0| \le 10^8$, $0 \le n \le 10^5$) — the speed of Peter's tractor, the coordinates of the object when first observed by Peter, and the number of times the object changes its velocity.

Each of the next $(n + 1)$ lines contains three space-separated integers t_i, v_{ix}, v_{iy} ($0 \le t_i \le 10^6$, $|v_{ix}|, |v_{iy}| \le 100$). It is guaranteed that $t_0 < t_1 < \ldots < t_n$. Note that it is possible that $(v_{ix}, v_{iy}) = (0, 0)$. Also note that movement of the object after moment t_n is infinite.

Output

In the only line print one real number with absolute or relative error not exceeding 10^{-6} — the maximum absolute value of the angular speed of Peter's head during his observation of the object.

Examples

standard input	standard output
1 0 -2 0 0 1 0	0.0000000000
2 0 -2 0 0 3 1	1.0000000000

Problem J. Jumping on a Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You are given a tree on n vertices. Suppose we are at the vertex v. In one step we can go from v to any other vertex u such that there are exactly d edges on the shortest path between v and u. A vertex u is *reachable* from v if we can get to u from v using zero or more steps.

Naturally, all vertices can be divided into *reachability classes*. A reachability class is a set of vertices C such that any vertex in C is reachable from any other vertex in C, but no vertex which is not in C is reachable from any vertex in C. How many reachability classes are there in the given tree?

Input

The first line contains two space-separated integers n and d ($1 \le n \le 10^6$, $0 \le d \le n$).

Next $n - 1$ lines describe edges of the tree. Each line contains two space-separated integers — indices of vertices connected by the corresponding edge. Indices are 1-based.

Output

Print one integer — the number of reachability classes.

Example

standard input	standard output
4 2 1 2 2 3 3 4	2

Problem K. King's Roads

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

King Byteasar wants to introduce an innovative road network in Byteland. There are n cities in Byteland. There are a_i citizens living in i-th city.

Initially, there are no roads in Byteland. The king wants to build several roads in such a way that all cities are connected (directly or indirectly). He also wants to minimize expenses on maintaining the roads.

Naturally, if a road connects populous cities, it is used more heavily and thus is more expensive to maintain. Formally, maintaining a road between cities i and j costs $a_i + a_j$ bytalers per year. However, if $a_i + a_j$ is at least M, the road is considered *a national highway*, and it brings revenue of M bytalers per year from taxation (thus, the effective cost for this road is $a_i + a_j - M$ per year). Note that all a_i are less than M.

Help King Byteasar find the minimum annual cost for maintaining a set of roads so that all cities become connected.

Input

The first line of input contains two space-separated integers n and M ($1 \le n \le 200\,000$, $1 \le M \le 10^9$).

The second line contains n space-separated integers $a_1, \ldots, a_n$ ($0 \le a_i < M$).

Output

Print one integer — the minimum annual cost for maintaining a spanning set of roads in bytalers.

Example

standard input	standard output
5 9 1 3 5 8 8	6