Problem A. Draft

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

n players are taking part in a collectible card game tournament. Before the match starts, players *draft* cards from a common pool to even the chances. There are m cards in the pool, numbered from 1 to m. The cards are ordered by their quality so that the card 1 is the best, and the card m is the worst. Initially, each player receives a **non-zero** number of cards so that all cards from 1 to m are distributed. Note that the numbers of cards may differ from player to player (due to the results of some previous stages).

The draft proceeds as follows. The players sit around a table in the order of their numbers. At each stage of the draft each player has a *hand* (which initially consists of the cards distributed to this player at the start) and a *deck* (which is initially empty). On each stage a player chooses the best card from his hand (that is, the card with the lowest number) and permanently adds it to his deck. If a player has an empty hand, then he does nothing on this stage. After adding the cards each player passes his hand to the next player, that is, the i-th player passes his hand to the $(i+1)$-th player if $i < n$, and the n-th player passes the hand to the first one. The draft continues until all players have empty hands.

You know the decks that the players have obtained after this process. You have missed the early tournament stages, so you have no idea which cards were initially distributed to the players before the draft (except that each player must have had at least one card). You want to compute the number of different initial distributions that would have lead to this particular arrangment of decks after the draft. Since the answer may be large, you want compute it modulo 998 244 353.

Input

The first line contains two integers n and m — the number of players and the number of cards in the pool ($1 \le n \le m \le 10^5$).

The next n lines describe the resulting decks of all players. i-th of these lines starts with an integer k_i — the size of the i-th player's deck ($1 \le k_i \le m$) followed by k_i integers $a^i_1, \ldots, a^i_{k_i}$ — the cards in his deck ($1 \le a_{i,1} < \ldots < a_{i,k_i} \le m$). It is guaranteed that each card from 1 to m is present in the deck of exactly one player.

Output

Print the number of initial card distribution that result in the described arrangement of decks after the draft modulo 998 244 353.

Examples

standard input	standard output

5 5
1 1
1 4
1 3
1 5
1 2
	1

2 3
2 2 3
1 1
	2

Note

In the second sample two possible distributions are $\{3\}$ to the first player and $\{1, 2\}$ to the second, and $\{2\}$ to the first player and $\{1, 3\}$ to the second.

Problem B. Power For Military

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 mebibytes

It is year 2112 and humanity has finally started to colonize Mars. To start with, a most cozy region was selected at the surface of the planet, and n space ships carrying power plants were sent there. You may consider the region as a two-dimensional plane. The i-th power plant will be landed at the point (x_i, y_i). You may consider the size of each plant to be negligible and treat it as a point. It is known that no two plants will land at the same moment of time and they were numbered in order the landing will take place.

Association of Crazy Militians (ACM) are obsessed with an idea that aliens will try to attack the plants at some moment of time. That is why they would like to have a plan of defense action. The key idea is to select three different power plants that should be protected and hold them with all forces available. *Inconvenience of defense* of three power plants i, j and k is defined as the maximum Cartesian distance among pairs (i, j), (j, k) and (i, k).

Now, as the power plants will arrive one by one to the planet's surface, ACM wants to know the minimum possible inconvenience of defense of some triple selected among first m plants. They would like to know this value for each m from 3 to n inclusive.

That should be a rather complicated value to compute. However, there is extra knowledge that might help. Although the whole operation was very well-planned and the set of n points to land stations was carefully pre-selected in advance, during the final arrangements there was a huge mess and now the order of points is guaranteed to be **random**. With this fact in stock, can you find out all interesting inconveniences of defense?

Input

The first line of the input contains a single integer n ($3 \le n \le 200\,000$) — the number of power stations to be landed on Mars surface.

Then follow n lines containing two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) each. It is guaranteed that a **random permutation** was used to define the order of points in all tests except samples.

Note, that it is **not guaranteed** that all plants will land at different positions. If two or more plants occupy the same position, any number of them (including three or none) can be used to form an optimal triple.

Output

Print $n - 2$ real values. For each m from 3 to n print the minimum possible maximum pairwise Cartesian distance in a triple of points selected among the first m. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Examples

standard input	standard output

3
1 1
1 2
1 3
	2.00000000

5
0 0
0 20
20 0
10 10
0 10
	28.28427125
	20.00000000
	14.14213562

Problem C. Coin Sliding

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 megabytes

Mathematician Charlie has n coins. He placed each of them on a strip so that the i-th coin is at the cell x_i. Any number of coins can be at the same cell simultaneously.

After doing that, Charlie performs several *moves*. A *move* consists of choosing two coins at cells x and y such that $y - x \geq 2$, and then moving them to cells $x + 1$ and $y - 1$ respectively.

Given initial positions of the coins, Charlie wonders what is the longest sequence of moves he could make until there are no more possible moves. Help him find the answer.

Input

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

The second line contains n integers $x_1, \ldots, x_n$ — initial positions of the coins ($0 \leq x_i \leq 10^6$).

Output

Print a single integer — the largest length of a sequence of moves starting from the original configuration.

Examples

standard input	standard output
3 0 2 4	3
4 1 2 1 2	0
5 0 3 3 5 8	13

Note

In the first sample one longest sequence of moves is as follows:

1. Move coins from cells 0 and 2 to the cell 1.
2. Move one coin from the cell 1 and the coin from the cell 4 to cells 2 and 3 respectively.
3. Move coins from cells 1 and 3 to the cell 2.

In the second sample no moves can be made from the start, hence the answer is 0.

Problem D. Good Boys

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 megabytes

Travis has n dogs. He let them out to run free in his yard. The yard can be viewed as a segment of length l bounded with a fence at points 0 and l. Initially the i-th dog from the left is at the point x_i, and is running either left or right. No two dogs share the same initial position.

Dogs run with uniform velocity 1. When two dogs meet at the same point, they instantly start running in the opposite directions. When a dog runs into a fence (that is, arrives to the point 0 or l), it instantly starts running in the opposite direction as well. One can show that three or more dogs can never meet at the same point simultaneously.

Travis has q questions for you. i-th of them is "what is the position of the a_i-th dog after exactly t_i seconds?"

Input

The first line contains two integers n and l — the number of dogs and the length of the yard ($1 \le n \le 10^5$, $2 \le l \le 10^9$).

The next n lines describe dogs' initial positions and running directions. i-th of these lines contains an integer x_i and a character d_i ($1 \le x_i \le l - 1$, $d_i \in \{\texttt{L}, \texttt{R}\}$. If the i-th dog initially runs to the left, then $d_i = \texttt{L}$, otherwise $d_i = \texttt{R}$. It is guaranteed $x_{i-1} < x_i$ for all i from 1 to $n - 1$.

The next line contains a single integer q — the number of questions ($1 \le q \le 10^5$).

The next q lines describe the questions. i-th of these lines contains two integers a_i and t_i ($1 \le a_i \le n$, $0 \le t_i \le 10^9$).

Output

For each question, print the answer on a separate line.

Example

standard input	standard output

```
3 5
1 R
2 L
4 R
12
1 1
2 1
3 1
1 2
2 2
3 2
1 3
2 3
3 3
1 4
2 4
3 4
```
	```
	1
	2
	5
	0
	3
	4
	1
	3
	4
	2
	2
	5
	```

Problem E. Prime Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A *binary string* is a string that contains characters `0` and `1` exclusively. In this problem characters of all strings are numbered starting from 0.

Let us define multiplication of binary strings as follows. If a and b are binary strings of length n and m respectively, then $a \times b$ is a binary string c of length $n \cdot m$ with characters defined by $c_i = (a_{i \bmod n} + b_{\lfloor i/n \rfloor}) \bmod 2$. That is, the first n characters of $a \times b$ coincide with the string a if $b_0 = 0$, or with the string $\overline{a}$ (character-wise negation of a) if $b_0 = 1$, the next n characters (with indices from n to $2n - 1$) are decided in the same way by the value of b_1, and so on. For example, `011` $\times$ `01` $=$ `011100`, but `01` $\times$ `011` $=$ `011010`.

A binary string c is *prime* if for any two binary strings a and b such that $c = a \times b$ exactly one of the strings a and b has length 1. For example, a string `011101` is prime, but strings `011010` $=$ `01` $\times$ `011` and `1` $=$ `0` $\times$ `1` are not.

Given an integer n, count the number of prime strings of length at most n. Since the answer may be large, compute it modulo $10^9 + 7$.

Input

The only line contains an integer n ($1 \le n \le 10^6$).

Output

Print the answer modulo $10^9 + 7$.

Examples

standard input	standard output
4	20
20	2088276

Note

Here is the list of all prime strings of length at most 4: `00`, `01`, `10`, `11`, `000`, `001`, `010`, `011`, `100`, `101`, `110`, `111`, `0001`, `0010`, `0100`, `0111`, `1000`, `1011`, `1101`, `1110`.

Problem F. Restore The Numbers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 megabytes

There is a graph G with n vertices numbered 1 through n, and m edges. There is also a permutation p of the numbers $1, \ldots, n$. The graph G has a property that there is an edge between vertices u and v if and only if $\min(p_u, p_v)$ divides $\max(p_u, p_v)$.

You are given the graph G, but the permutation p is lost. Restore any permutation satisfying the condition above. It's guaranteed that at least one solution exists.

Input

The first line contains two integers n and m — the number of vertices and edges in the graph ($2 \le n \le 10^5$, $1 \le m \le 2 \cdot 10^6$) . Each of the next m lines contains numbers u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — the endpoints of the respective edge of the graph. The graph has no loops nor multiple edges.

Output

Print n numbers $p_1, \ldots, p_n$. If there are several possible answers, output any of them.

Examples

standard input	standard output

4 4
4 1
3 4
1 3
2 4
	2 3 4 1

6 8
2 1
3 1
6 1
1 5
5 2
3 5
1 4
2 4
	1 2 3 4 6 5

Note

In the first sample there are two solutions: $2, 3, 4, 1$ and $4, 3, 2, 1$.

Problem G. Shuffled Switches

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

This problem is interactive.

New, 128th, campus for students of IPMT was built; but an unexpected problem occurs.

There are n rooms numbered from 1 to n in new campus; in each of them, there is a light bulb and a switch. Each switch is connected with exactly one bulb, and each bulb is connected with exactly one switch. But the problem is that each of the switches may turn on a bulb placed in a different room than it's placed in!

It was decided to understand how switches are connected with bulbs. To do that, it's possible to perform no more than $2n$ operations, each of them consists of the following actions:

1. Choose an integer k $(1 \le k \le n)$, and k integers $a_1, a_2, \ldots, a_k$ $(1 \le a_1 < a_2 < \ldots < a_n \le n)$;
2. Ask k workers to go to the rooms $a_1, a_2, \ldots, a_k$ and try to turn on the switches.
3. After that, each of workers goes back and says whether the bulb in the room he visited was turned on or not. Each worker turns off the switch when he leaves the room.

After performing the operations, you are to find out the number of the room with the corresponding bulb for each switch.

Interaction Protocol

Initially your program receives one integer n — the number of rooms $(1 \le n \le 100)$.

Then, your program can perform the following operation at most $2n$ times:

- print a string "`?` s", where s is a string of n characters $s_1, s_2, \ldots, s_n$. s_i should be equal to 1 if we want to send a worker to the room i during current operation, and 0 otherwise.
- Each query of this type will be answered; the answer is a string of n characters $t_1, t_2, \ldots, t_n$. t_i will be equal to:
  - "`-`" if no worker was sent to the room i;
  - "`1`" if a worker was sent to the room i, and he saw the bulb turning on;
  - "`0`" if a worker was sent to the room i, but he didn't see the bulb turning on.

After all these operations, your program should print an answer in the format "! p_1 p_2 $\ldots$ p_n", where p_i is the number of a room in which the bulb can be turned on by the switch in the i-th room. After printing the answer, the program should terminate immediately.

Example

standard input	standard output
3	
	? 000

	? 100
0--	
	? 010
-0-	
	? 001
--0	
	? 101
0-1	
	! 3 1 2

Problem H. String Sequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a string s, you can produce a sequence of strings s_1, s_2, …, defined as following:

- $s_1 = s$;
- $s_2 = s_1 + t_1$, where t_1 is a nonempty prefix of s_1;
- $s_i = s_{i-1} + t_{i-1}$, where t_{i-1} is a prefix of s_{i-1} and $|t_{i-1}| \geq |s_{i-2}|$.

You can select any prefix of s_i as t_{i-1} as long as it satisfies the length constraint.

Given a string p, what is the maximum length of the sequence that ends in p? Formally, what is the maximum k such that there exist strings s_1, …, $s_k = p$ which were produced by the described process?

Input

The only line contains the string p of lowercase English characters. The length of p is between 1 and 100 000, inclusive.

Output

Print a single integer — the length of the longest valid sequence that ends in p.

Examples

standard input	standard output
`abcababca`	`3`
`bebebe`	`3`
`moscode`	`1`

Note

In the first sample the sequence may be "`abc`", "`abcab`", "`abcababca`".

In the second sample the sequence may be "`be`", "`bebe`", "`bebebe`".

In the third sample the sequence consists of only one element.

Problem I. Adding Palindromes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

You have an array of numbers $a_0, \ldots, a_{n-1}$ initially filled with zeroes that you need to transform to another array.

You can modify the array as follows. Choose a position i ($0 \le i \le n-1$) and a sequence of numbers $p_0, \ldots, p_{l-1}$ ($i + l - 1 < n$, $0 \le p_j < 10^9 + 7$). The sequence must be a *palindrome*, that is, for each index $0 \le j < l$ we must have $p_j = p_{l-1-j}$. The operation results in element-wise addition of p_j to the array modulo $10^9 + 7$ starting from the position i, that is, for each index j from 0 to $l - 1$ the value of a_{i+j} changes to $(a_{i+j} + p_j) \bmod (10^9 + 7)$.

Find the shortest sequence of operations that reaches your goal.

Input

The first line of the input contains a single integer n — the length of the array ($1 \le n \le 10^3$). The next line contains n integers $b_0, \ldots, b_{n-1}$ — the desired values of the elements ($0 \le b_i < 10^9 + 7$).

Output

In the first line print a single integer k — the number of operations in your sequence. Then k descriptions of operations should follow. A description of an operation starts with a line containing two integers — the starting index i and the length l of the palindromic array. The following line should contain l integers $p_0, \ldots, p_{l-1}$ satisfying $0 \le p_j < 10^9 + 7$, and $p_j = p_{l-j-1}$ for all j from 0 to $l - 1$.

Examples

standard input	standard output

5
2 1 1 2 2
	2
	0 5
	2 2 2 2 2
	1 2
	1000000006 1000000006

6
1 2 3 3 2 1
	1
	0 6
	1 2 3 3 2 1

Problem J. Shrinking Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 megabytes

You are given a weighted undirected tree with n vertices. Denote the weight of the edge xy by $w_{x,y}$. Let's say that a tree path is a *diameter* if the sum of weights of its edges is the largest among all simple paths in the tree. *Shrinking* by a number a is the following operation:

- Choose an arbitrary diameter $u_1, \ldots, u_k$ in the tree.
- Shrink all edges in the diameter by a, that is, replace $w_{u_i,u_{i+1}}$ with $\max(0, w_{u_i,u_{i+1}} - a)$ for all i from 1 to $k-1$.

Consider the following process. Choose a number $\varepsilon > 0$, and repeatedly perform shrinking by ε until all edges have zero weight. Let $f(\varepsilon)$ be the number of shrinking operations in this process. Your task is to find $\lim\limits_{\varepsilon \to 0} \varepsilon \cdot f(\varepsilon)$.

It is guaranteed that the limit exists, and the answer is the same regardless of a method to choose a diameter among several options when shrinking.

Input

The first line contains a single integer n — the number of vertices of the tree ($2 \le n \le 1000$).

Next $n-1$ lines describe the edges. Each of these lines contains three integers a_i, b_i, c_i — the endpoints indices and the weight of the respective edge ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le c_i \le 1000$).

Output

Print a single real number $\lim\limits_{\varepsilon \to 0} \varepsilon \cdot f(\varepsilon)$. Your answer will be considered correct if the absolure or relative error does not exceed 10^{-4}.

Examples

standard input	standard output
2 1 2 15	15.00000000000000000000

4
1 2 3
2 3 2
2 4 2
	3.50000000000000000000

Note

In the first sample the only edge is going to be shrinked repeatedly, hence $\varepsilon \cdot f(\varepsilon) = 15 + O(\varepsilon)$.

In the second sample the following will happen. First, the diameters $(1, 2, 3)$ and $(1, 2, 4)$ are going to be shrinked alternately, until all edges have weights close to 1 (this will take about $2/\varepsilon$ operations). After that, the diameters $(1, 2, 3)$, $(1, 2, 4)$, and $(3, 2, 4)$ are going to be shrinked in some order so that weights of all edges remain close (this will take about $1.5/\varepsilon$ operations).

Problem K. Wrong Subtraction

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

asya was too busy playing Angry Birds in his arithmetics class, so he's not very good at doing basic calculations. For example, Vasya uses the following algorithm for subtracting two numbers a and b:

- If one of the numbers has fewer digits than the other, pad it with zeros from the beginning.
- Write numbers a and b in two rows so that the digits are aligned in columns.
- Process the digits from right to left. Let d_a and d_b be the current digits of a and b respectively, there is also a *carry* value of c equal to -1, 0 or 1. For the first pair of digits c is set to 0.
- If $0 \leq d_a - d_b + c < 10$, write $d_a - d_b + c$ as the corresponding digit of the answer and set the carry be equal to 0.
- If $d_a - d_b + c \geq 10$, write $d_a - d_b + c - 10$ as the corresponding digit of the answer, and set the carry to be equal to -1.
- If $d_a - d_b + c < 0$, write $d_a - d_b + c + 10$ as the corresponding digit of the answer, and set the carry to be equal to 1.
- If there are no next pair of digits and carry is non-zero, prepend both numbers with zeroes and proceed.

Here is one example of this "subtraction". If Vasya subtracts 26 from 51, he obtains:

```
51
26
==
45
```

He can ever subtract a larger integer from a smaller one, from example, computing $1 - 303$ will result in:

```
0001
0303
====
1718
```

Another example:

```
0090
0001
====
1909
```

Despite this procedure being clearly wrong, it always terminates and produces a non-negative integer as a result.

Vasya was practicing with "subtraction", and has written down numbers $a_1, \ldots, a_n$ on a piece of paper. He is sure that at some point he "subtracted" his favourite number x from one of the integers and written down the result, hence for some indices $i \neq j$ we must have $a_i - a_j = x$, where $-$ denotes the result of Vasya's algorithm. Help him verify his conjecture, and find any pair of suitable indices.

Input

The first line contains two integers n and x — the amount of numbers on the piece of paper and Vasya's favourite number ($2 \leq n \leq 10^5$, $1 \leq x \leq 10^9$).

The second line contains n integers $a_1, \ldots, a_n$ ($0 \le a_i \le 10^9$).

Output

If there are two indices i and j ($1 \le i, j \le n$, $i \ne j$) such that $a_i - a_j = x$ according to Vasya's algorithm, print them in the only line, otherwise print "`-1 -1`" (without the quotes). If there are several answers, print any of them.

Examples

standard input	standard output
2 45 51 26	1 2
3 9 9 18 27	-1 -1