Problem A. Task Manager

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 megabytes

Bomboslav is an intern in Yandex who works on improving the internal task manager. The current version of the manager stores two parameters c_i and u_i for each of n tasks assigned to some employee. These parameters denote importance and urgency of the task, respectively. Higher values correspond to higher importance or urgency.

An employee can choose to perform tasks in any order, with only one condition: if some task i is **both** more important and more urgent than some task j, that is, conditions $c_i > c_j$ and $u_i > u_j$ hold simultaneously, task i should go before task j.

Bomboslav decided to add a tool that will advice employees in what order to perform tasks so that they will not break any condition. That turned out to be too easy, so Bomboslav added one more feature: for every task i, employee could specify the pleasure p_i he would get from performing this task. These values must be distinct for all tasks.

When the values p_i are specified for all n tasks available, the task manager should suggest an order of tasks that does not break any initial conditions, and the values p_i arranged in the same order would form the lexicographically maximum sequence. In other words, from all orders that break no conditions on importance and urgency, the task manager should choose the one where p_i of the first task is the maximum possible. If there is still a tie, the task manager should choose an order that has the maximum possible p_i of the second task, and so on.

Input

The first line of input contains a single integer n, the number of tasks in the manager for some employee ($1 \le n \le 100\,000$).

Then follow n lines with task descriptions. Each of them consists of three non-negative integers c_i, u_i and p_i: importance, urgency and pleasure from this task, respectively ($0 \le c_i, u_i, p_i \le 10^9$). All p_i are guaranteed to be pairwise distinct.

Output

Print a correct order of performing this set of tasks such that the corresponding sequence of p_i is lexicographically the maximum possible. The tasks are numbered from 1 in the order they are given in the input. Each task must occur in this order exactly once. One could easily prove that the answer is always unique.

Examples

standard input	standard output

3
1 2 7
2 1 5
3 3 0
	3 1 2

Problem B. Bus for a Picnic

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

Every summer, Yandex.Picnic is organized in the countryside for Yandex employees. For employees without their own car, buses are ordered to transport them from the office to the picnic location.

You know how many employees will use the bus. Moreover, some groups of employees would like to sit in the same row of seats. The bus provided by the transporter has six seats in each seating row.

Before making the order, you have to determine how many rows of six seats each are needed to accomodate all employee groups according to their requests.

Input

The first line of input contains the only integer n, the number of employee groups ($1 \le n \le 100$).

The second line contains n space-separated integers a_i, the sizes of employee groups ($1 \le a_i \le 6$).

Output

Print one integer: the minimal number of six-seat rows to accomodate all employee groups.

Examples

standard input	standard output

`7`
`1 1 1 1 1 1 1`
	`2`

`7`
`2 3 2 3 2 4 4`
	`4`

Problem C. Integer Product

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

Given are $N - 1$ fractions u_i/d_i such that $2 \le u_i \le N$, $2 \le d_i \le N$, all u_i are pairwise distinct, and all d_i are pairwise distinct. Choose at least 1 and at most $N/10$ of these fractions so that each of their denominators is a prime power, and the product of the chosen fractions is an integer.

Input

The first line of input contains one integer N ($10 \le N \le 10^6$). Each of the next $N - 1$ lines contains one fraction in the format u_i/d_i ($2 \le u_i, d_i \le N$, $u_i \ne u_j$ if $i \ne j$, $d_i \ne d_j$ if $i \ne j$). There are no spaces between the numbers and the "/" sign.

Output

The first line of output must contain an integer M, the number of selected fractions. Each of the next M lines must contain one of the selected fractions in the same format as the input. The denominator of each of the selected fractions must be a prime power, that is, equal to $p_i^{k_i}$ where p_i is a prime number and k_i is a positive integer. Fractions can be listed in any order. A fraction can not be selected more than once.

If more than one solution exists, print any of them. If no solution exists, print -1 on the first line.

Examples

standard input	standard output
11 3/3 4/9 5/8 7/6 2/11 9/4 11/2 10/10 6/7 8/5	2 5/8 8/5

Problem D. Unite and Conquer

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Not a long time ago, Yandex employees patented a new algorithm for optimal data fragmentation that will help to transfer huge data over the network. Now they face the task of finding an occurrence of already fragmented pattern in fragmented data. Sounds to be hard, but they are ready to accept the challenge, especially with your help. From this point, we would skip all the motivation and formulate the task itself.

You are given a sequence a_i formed by n integers in range from 1 to 5. These sequence describe the sizes of consecutive elements in data fragmentation. Also, you are given a sequence b_i formed by m integers in the same range. This sequence describe the sizes of consecutive elements in pattern fragmentation.

The only operation you are allowed to perform is to pick **any (but only one)** sequence and two neighboring elements in it and unite them in one with value equal to their sum. For example, having a sequence $1, 2, 2$, one could obtain only $3, 2$ and $1, 4$ after one operation. It is not prohibited to form elements that are greater than 5.

The goal is to count the minimum number of operations that must be performed to make the sequence b_i a substring of a_i.

Input

The first line of input contains two integers n and m: the number of elements in fragmentation of the initial data and the number of elements in fragmentation of the pattern, respectively ($1 \le n, m \le 200\,000$).

The next line contains n integers a_i, each of them in the range from 1 to 5 inclusive: the sizes of the corresponding elements.

The last line contains m integers b_i — the fragmentation of the pattern in the same format.

Output

Print one integer: the minimum number of moves required to make the second sequence a substring of the first sequence. If it is impossible, print -1.

Examples

standard input	standard output

6 4
5 1 1 1 2 2
2 1 1 1
	2

8 4
1 2 1 2 1 2 1 2
1 3 1 2
	3

3 2
3 3 3
4 4
	-1

Problem E. Compact Strings

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

This problem is about strings of lowercase English letters ("`a`"–"`z`").

A string is called compact if the occurrences of each letter form a contiguous substring. For example, "`zyx`", "`aacccb`", and "`eeeee`" are compact, while "`aba`" and "`aazbbzc`" are not.

You are given a pattern that consists of lowercase English letters and wildcards. Each wildcard (denoted "`?`") represents a single unknown lowercase English letter.

Count all compact strings that correspond to the pattern. Report the number modulo $10^9 + 7$.

Input

The first line of input contains the number t of test cases ($1 \le t \le 100$). Each of the following t lines contains one test case. Each test case consists of between 1 and 10^4 characters, inclusive. Each of those characters is either a question mark or a lowercase English letter.

Output

For each test case, output the number of matching compact strings, modulo $10^9 + 7$.

Examples

standard input	standard output
5 aa??zz aacccb ?x? a??b??a a??a	627 1 651 0 1

Problem F. Random Points

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

Consider the following linear congruential generator of pseudo-random values. The generator has a state: an integer s ($0 \le s < 2^{32}$). To produce the next pseudo-random integer in the interval $[0, X)$, the generator first assigns $s_{\text{new}} = (a \cdot s_{\text{old}} + c) \bmod 2^{32}$. After that, s_{new} becomes the new state of the generator, and the result is the number

$$\left\lfloor \frac{s_{\text{new}} \cdot X}{2^{32}} \right\rfloor .$$

In this problem, $a = 134\,775\,813$ and $c = 1$ (these values are used in the builtin pseudo-random number generator in Borland Delphi 7).

The generator is used to obtain n random points on a plane. The coordinates are integers in the interval $[0, 100\,000\,000)$. Let us consider two methods of generation.

In the first method (code named `RAW`), the state is initially set to s, and after that, the generator sequentially produces $2n$ pseudo-random numbers a_1, a_2, …, a_{2n} in the required interval. After that, the n points are defined by the following pairs of coordinates: (a_1, a_2), (a_3, a_4), …, (a_{2n-1}, a_{2n}).

In the second method (code named `SHUFFLED`), the beginning is the same: the state is initially set to s, and after that, the generator sequentially produces $2n$ pseudo-random numbers a_1, a_2, …, a_{2n} in the required interval. However, after that, the $2n$ numbers are randomly shuffled with the following variant of Fisher-Yates algorithm (in pseudocode):

```
for $i = 1, 2, \ldots, 2 \cdot n$ :
    $k = \text{random } [0, i) + 1$
    swap $(a_i, a_k)$
```

Here, random $[0, i)$ is a pseudo-random integer from the interval $[0, i)$ which is produced by the same generator, and swap (a_i, a_k) exchanges the numbers a_i and a_k; if $i = k$, nothing happens. Before the shuffling starts, the generator is in the state in which is ended up after generating a_1, a_2, …, a_{2n}.

Finally, when the numbers a_1, a_2, …, a_{2n} are shuffled, the n points are defined similarly by the following pairs of coordinates: (a_1, a_2), (a_3, a_4), …, (a_{2n-1}, a_{2n}).

You are given a sequence of n points ($10\,000 \le n \le 100\,000$). It is known that it is a result of either the first or the second method of generation specified above. It is however not known which method was chosen and what was the initial value of s. Find out which of the two methods was used to generate the sequence.

Input

The first line of input contains a single integer n, the number of points ($10\,000 \le n \le 100\,000$). The next n lines contain the descriptions of points, one per line. Each description consists of two integers separated by a space: the coordinates x and y of a point ($0 \le x, y < 100\,000\,000$). It is guaranteed that the given sequence of points is a result of either the first or the second method of generation, as specified in the statement.

Output

Print "`RAW`" if the points were generated by the first method, or "`SHUFFLED`" if they were generated by the second method.

Examples

standard input	standard output

25000
72284508 84234312
31215087 251372
...
10396121 60229057
	RAW

25000
50750089 47132
96007660 62545522
...
37227017 64995066
	SHUFFLED

Note

The full version of the examples can be downloaded using the contest system interface. In Yandex.Contest, to download all examples to all problems, use the “Download statements” link at the top right.

In both examples, the initial value of s is equal to 1234.