# Problem A. Accumulator Battery

Time limit: 3 seconds
Memory limit: 512 megabytes

Anna loves her cell phone and becomes very sad when its battery level drops to 0 percent.

In normal mode, Anna's phone battery drains at a constant speed. When the battery level reaches 20 percent, the phone automatically switches to eco mode. In eco mode, the battery drains two times slower than in normal mode.

Alex has invited Anna for a date. Anna needs $t$ minutes to get from her home to the meeting place. When Anna leaves home, her phone's battery level is 100 percent. At the moment she reaches the meeting place, the battery level will be $p$ percent.

Alex wonders for how long Anna will be in a good mood after they meet. Help him solve this problem!

## Input

The only line of the input contains two integers $t$ and $p$ — time Anna needs to get from her home to the meeting place, in minutes, and the battery level of her phone at the moment of meeting, in percent ($1 \le t \le 360$; $1 \le p \le 99$).

## Output

Output a single real number — time since the moment of meeting before Anna's phone runs out of battery, in minutes.

Your answer will be considered correct if its absolute or relative error doesn't exceed $10^{-4}$.

## Examples

| standard input | standard output |
|---|---|
| 30 70 | 90.0 |
| 120 5 | 10.909091 |

## Note

In the first test case, the battery drains at a rate of one percent per minute. In 50 minutes after the meeting, the battery level will reach 20 percent and the phone will switch to eco mode. 40 minutes later the phone will run out of battery.

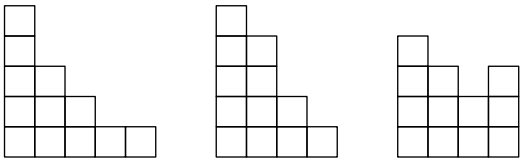
In the second test case, the phone is already in eco mode. The battery level will be enough for a little less than 11 minutes.

# Problem B. Building a Stair

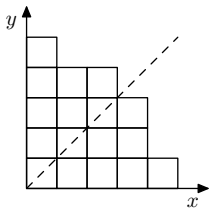
Time limit: 3 seconds
Memory limit: 512 megabytes

Little Barney just got a new toy cube set from his parents. His set contains $n$ identical cubes. Barney immediately started building various objects with these cubes.

The latest thing Barney built is a *stair*. A stair consists of one or more towers of cubes, where the heights of towers are *non-increasing* from left to right. In the following picture, you can see three different shapes with 12 cubes each. The first two are stairs and the third one is not a stair.



Barney noticed that for some stairs you can turn your head 90 degrees to the right and you will see the same stair, but reversed! He calls such stairs *symmetric*. For example, the first stair above is symmetric, but the second one is not. Formally, a stair is symmetric if and only if when you reflect the picture over the $x = y$ line, you get the same stair (where the $x$-axis is horizontal and oriented to the right, and the $y$-axis is vertical and oriented upwards).




Barney wants to build a symmetric stair using all of his $n$ cubes. Show him how to do it!

## Input

The single line of the input contains an integer $n$ — the number of cubes at Barney's disposal ($1 \le n \le 100$).

## Output

If there is no symmetric stair with $n$ cubes, output a single integer $-1$.

Otherwise, in the first line, output one integer $m$ — the number of rows and columns in the picture of the stair ($1 \le m \le 100$). Then, output $m$ lines describing the stair. Each line must contain exactly $m$ characters 'o' (a lowercase English letter) or '.', where 'o' describes a cell with a cube, and '.' describes an empty cell. There must be exactly $n$ 'o' characters in total. The cell in the bottom left corner must contain a cube. If there is more than one solution, output any of them.

## Examples

| standard input | standard output |
|---|---|
| 3 | 3<br>...<br>o..<br>oo. |
| 17 | 5<br>o....<br>ooo..<br>oooo.<br>oooo.<br>ooooo |

# Problem C. Counting Stairs

| | |
|---|---|
| Time limit: | 3 seconds |
| Memory limit: | 512 megabytes |

Remember Barney from problem B? Barney's older sister Cecilia often watches him play with his set of cubes. She also joins Barney in his games and prevails most of the time, shaking his confidence on a daily basis.

One day Cecilia noticed Barney struggling to build a *symmetric stair* with his $n$ cubes. She immediately told him she could not just build a symmetric stair, but even calculate the number of different symmetric stairs consisting of $n$ cubes! Can you?

Recall that a *symmetric stair* consists of one or more towers of cubes, where the heights of towers are *non-increasing* from left to right, and is symmetric with respect to the $x = y$ line (where the $x$-axis is horizontal and oriented to the right, and the $y$-axis is vertical and oriented upwards). For a more detailed explanation, please refer to problem B statement.

The number of different symmetric stairs can be quite large, so you need to calculate it modulo 998 244 353.

## Input

The input contains multiple test cases.

The first line of the input contains a single integer $t$ — the number of test cases ($1 \le t \le 10^4$). Each of the following $t$ lines contains a single integer $n_i$ — the number of cubes in the $i$-th test case ($1 \le n_i \le 2 \cdot 10^5$).

## Output

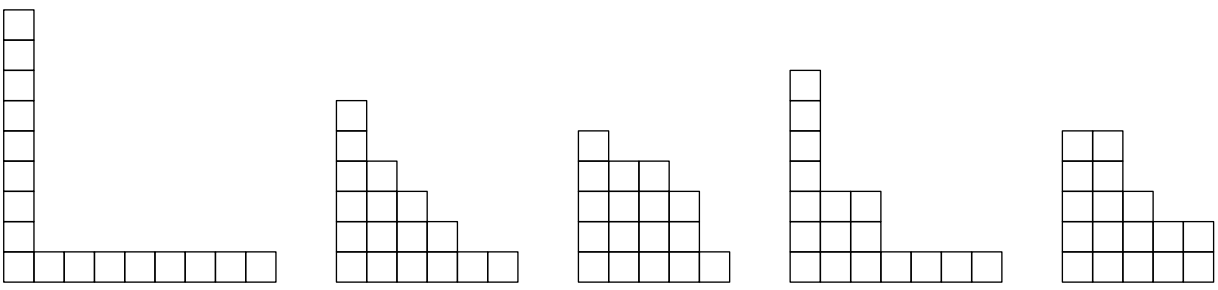
For each test case, output a line with a single integer — the number of symmetric stairs with exactly $n_i$ cubes, modulo 998 244 353.

## Example

| standard input | standard output |
|---|---|
| 4<br>3<br>5<br>17<br>25 | 1<br>1<br>5<br>12 |

## Note

All different symmetric stairs with $n = 17$ cubes are shown below:



# Problem D. Distinct Substrings

Time limit: 3 seconds
Memory limit: 512 megabytes

Diana bought a Long Random String Generator on some weird website. She planned to generate a long string $s$ of length $n$ and then use its contiguous substrings as passwords for other weird websites.

Soon she discovered that the generated string $s$ of length $n$ was not random at all, but rather a string $p$ of length $k$ repeated many times and then cut to length $n$. Thus, $s[i] = p[i \bmod k]$ for all $i$ from 0 to $n - 1$.

Diana wonders how many different passwords she can get from the generated string. Help her find the number of distinct non-empty substrings in string $s$.

## Input

The first line of the input contains a string $p$ consisting of $k$ lowercase English letters ($1 \le k \le 1000$).

The second line contains an integer $n$ ($k \le n \le 10^9$).

## Output

Output the number of distinct non-empty substrings in $s$.

## Examples

| standard input | standard output |
|---|---|
| abba<br>7 | 20 |
| a<br>42 | 42 |

## Note

In the first example, the generated string is `abbaabb`. It contains 20 distinct non-empty substrings: `a`, `b`, `aa`, `ab`, `ba`, `bb`, `aab`, `abb`, `baa`, `bba`, `aabb`, `abba`, `baab`, `bbaa`, `abbaa`, `baabb`, `bbaab`, `abbaab`, `bbaabb`, `abbaabb`.

# Problem E. Email Destruction

Time limit: 3 seconds
Memory limit: 512 megabytes

You have an account on ICPCorrespondence.com. This is an email service where emails are grouped into chains by their subject as follows.

The first email in every chain has a non-empty subject consisting of lowercase English letters. Every succeeding email in the chain has a subject consisting of "`Re: `" followed by the subject of the previous email.

For example, if the first email in the chain has subject "`subj`", then the second email has subject "`Re: subj`", the third one has subject "`Re: Re: subj`", and so on. Formally, the subject of the $k$-th email in the chain consists of "`Re: `" repeated $k-1$ times followed by the subject of the first email in the chain.

In your mailbox, you had one or more chains of emails with unique subjects. You never removed any emails from your mailbox.

Unfortunately, one day ICPCorrespondence.com was attacked by hackers. As a result of this attack, some emails were removed from the server, while the remaining emails were shuffled.

You are not sure how many emails you had in the mailbox before the attack, but you guess that this number is $n$. Can you check whether this guess can be correct?

## Input

The first line of the input contains two integers $n$ and $k$ — the number of emails that you think were in the mailbox before the attack, and the number of emails left after the attack, respectively ($1 \le k \le n \le 100$).

The following $k$ lines contain subjects of the emails left in your mailbox, one per line. The subject of each email consists of "`Re: `" repeated zero or more times, followed by at least one and no more than 10 lowercase English letters. The length of each subject does not exceed 500. All email subjects are pairwise distinct.

## Output

If your guess about the number of emails in your mailbox prior to the attack can be correct, output a single word "`YES`". Otherwise, output a single word "`NO`".

## Examples

| standard input | standard output |
|---|---|
| 7 3<br>Re: Re: Re: hello<br>Re: world<br>hello | YES |
| 3 2<br>Re: Re: pleasehelp<br>me | NO |

## Note

In the first example, the guess can be correct. For example, you could have emails with subjects "`hello`", "`Re: hello`", "`Re: Re: hello`", "`Re: Re: Re: hello`", "`Re: Re: Re: Re: hello`", "`world`", and "`Re: world`".

In the second example, the guess is incorrect since there had to be at least three emails in the chain of "`pleasehelp`" and at least one email in the chain of "`me`".

# Problem F. Forgotten Land

| | |
|---|---|
| Time limit: | 3 seconds |
| Memory limit: | 512 megabytes |

Fytelandian scientists are famous for their love for historical research. Recently they have discovered the remains of $n$ cities, which definitely belonged to a great past civilization.

All the discovered cities were connected by exactly $n - 1$ roads, and there was exactly one way to move between any pair of cities using these roads. The scientists found a lot of writings in each city and concluded that people of the civilization spoke $k$ different languages. In city $v$ people spoke language $a_v$.

It is known that the cities formed several alliances, and each city belonged to exactly one alliance. However, the exact formation of alliances remains unknown.

Let's say that an alliance is an arbitrary set of cities $c_1, c_2, \ldots, c_m$ that might or might not be connected by roads. To rule the alliance it was necessary to be able to make connections between different cities with people who speak different languages. The head of the alliance had to support all languages that were spoken either in some city $c_i$ of the alliance or in some city on the shortest path between some two cities $c_i$ and $c_j$ of the alliance.

The more languages were supported in the alliance, the harder was the translator's job — a good translator had to speak all of them! Luckily, many languages were similar to each other, and the more languages the translator had known, the easier it was for them to learn a new language. Every consecutive new language took twice less time for the translator to learn.

Let's say that the *language difficulty* of the alliance is the time the translator needed to spend to learn all languages supported in the alliance. The first language took $2^k$ units of time to learn, thus, the language difficulty of the alliance is equal to $2^k + 2^{k-1} + \ldots + 2^{k+1-t}$, where $t$ is the number of different languages supported in the alliance. Note that as there are $k$ languages in total, this sum is always an integer.

Let's say that an *alliance partition* is a partition of all cities into several alliances. Two alliance partitions are considered different if there exist two cities $u$ and $v$ such that they belong to the same alliance in one partition and to different alliances in the other partition.

Let's say that the *plausibility* of an alliance partition is the sum of the language difficulties of all alliances in the partition.

You need to calculate the sum of the plausibilities of all possible alliance partitions. As this number can be very big, you need to find only its remainder modulo 998 244 353.

## Input

The first line of the input contains two integers $n$ and $k$ — the number of cities and the number of languages, respectively ($1 \le n \le 5000$; $1 \le k \le 10$).

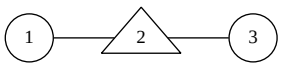
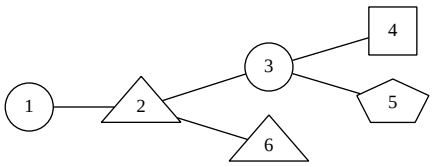
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ — the languages spoken in the cities ($1 \le a_i \le k$).

Each of the following $n - 1$ lines contains two integers $u_i$ and $v_i$ — cities connected by the $i$-th road ($1 \le u_i, v_i \le n$).

## Output

Output the sum of the plausibilities of all possible alliance partitions modulo 998 244 353.

## Examples

| standard input | standard output | Illustration |
|---|---|---|
| 3 2<br>1 2 1<br>1 2<br>2 3 | 48 |  |
| 6 4<br>1 2 1 3 4 2<br>1 2<br>2 3<br>3 5<br>3 4<br>2 6 | 14504 |  |

## Note

In the first example, there are three cities. Consider all possible alliance partitions:

- Three alliances:
  - $\{1\}$. One city, the only supported language is language 1. The language difficulty is $2^2 = 4$.
  - $\{2\}$. One city, the only supported language is language 2. The language difficulty is $2^2 = 4$.
  - $\{3\}$. One city, the only supported language is language 1. The language difficulty is $2^2 = 4$.

  The plausibility of this partition is $4 + 4 + 4 = 12$.
- Two alliances:
  - $\{1, 2\}$. Two cities, languages 1 and 2 are supported. The language difficulty is $2^2 + 2^1 = 6$.
  - $\{3\}$. One city, the only supported language is language 1. The language difficulty is $2^2 = 4$.

  The plausibility of this partition is $6 + 4 = 10$.
- Two alliances:
  - $\{1, 3\}$. Two cities, languages 1 and 2 are supported. Note that even though the only language spoken in the alliance is language 1, language 2 is supported as well because it's spoken in city 2 that lies on the shortest path between cities 1 and 3. The language difficulty is $2^2 + 2^1 = 6$.
  - $\{2\}$. One city, the only supported language is language 2. The language difficulty is $2^2 = 4$.

  The plausibility of this partition is $6 + 4 = 10$.
- Two alliances:
  - $\{1\}$. One city, the only supported language is language 1. The language difficulty is $2^2 = 4$.
  - $\{2, 3\}$. Two cities, languages 1 and 2 are supported. The language difficulty is $2^2 + 2^1 = 6$.

  The plausibility of this partition is $4 + 6 = 10$.
- One alliance:
  - $\{1, 2, 3\}$. Three cities, languages 1 and 2 are supported. The language difficulty is $2^2 + 2^1 = 6$.

  The plausibility of this partition is 6.

The required sum is equal to $12 + 10 + 10 + 10 + 6 = 48$.

# Problem G. Generalized German Quotation

Time limit: 3 seconds
Memory limit: 512 megabytes

German language uses conventional angular quote marks ('«' and '»'), so one may quote text in a «conventional» way. What is unconventional in German is that one may also quote text in a »reversed« way. In normal life these styles do not mix, since they are used in different German-speaking countries. But let us have some fun! If we merge these two typographical traditions and forget about rules for nested quotes (that is, if we allow unlimited nesting), we will receive Generalized German rules that allow us to write small quotation masterpieces like this:

«»Anführungszeichen« means «quote marks» in German»

Informally we will say that a string is a correct quotation if it can be obtained by removing all non-quote characters from a correctly formed Generalized German text. Formally:

$$\langle G\rangle ::= \varepsilon \mid \langle G\rangle\langle G\rangle \mid \text{`«'}\langle G\rangle\text{`»'} \mid \text{`»'}\langle G\rangle\text{`«'}$$

Thus, a correct quotation is an empty string, a concatenation of two correct quotations, or a correct quotation quoted in either a conventional or a reversed way. In the latter case, we will say that the quote mark to the left of $\langle G\rangle$ is a *starting* quote, and the quote mark to the right of $\langle G\rangle$ is an *ending* quote. For example, in quotation string '«»' the quote mark '«' is a starting quote, while in string '»«' the same quote mark '«' is an ending quote.

Your task is to check whether the given string is a correct quotation, and if it is, restore its structure — that is, replace all starting quote marks with '`[`' and all ending quote marks with '`]`'.

## Input

The first and only line of the input contains a single string with a sequence of quote marks. To limit ourselves to plain ASCII, the quote marks '«' and '»' are encoded as '`<<`' and '`>>`', respectively. The string does not contain any other characters. The string is not empty and is not longer than 254 ASCII characters.

## Output

If the input string is a correct quotation, replace all starting quote marks with '`[`', all ending quote marks with '`]`', and output the result. If there is more than one possible solution, output any of them.

If the string is not a correct quotation, output "`Keine Loesung`".

## Examples

| standard input | standard output |
|---|---|
| `<<>><<<<>>>>` | `[][[]]` |
| `>><<>>>><<<<<<>>>>>><<` | `[][[]][][]` |
| `<<<<>>` | `Keine Loesung` |

# Problem H. Halves Not Equal

| | |
|---|---|
| Time limit: | 3 seconds |
| Memory limit: | 512 megabytes |

The king died and his gold had to be divided among his $n$ wives. He had not left his will about the parts of his wives, so they started arguing. The $i$-th wife claimed that she should get $a_i$ dinars.

However, it turned out that the total property of the king was only $s$ dinars, and $s \le a_1 + a_2 + \ldots + a_n$. A wise man was called to help divide the king's inheritance. But he said that he only knew a fair way to divide gold between two persons.

The *fair way* is the following. Without loss of generality, let the claims of the two persons be $a_1 \le a_2$, and let there be $b$ dinars of gold to be divided, $0 \le b \le a_1 + a_2$. If $b \le a_1$, each of the persons would get $b/2$ dinars. If $a_1 < b < a_2$, the first one would get $a_1/2$ dinars and the second one would get $b - a_1/2$ dinars. Finally, if $a_2 \le b$, the first one would get $a_1/2 + (b - a_2)/2$ and the second one would get $a_2/2 + (b - a_1)/2$. Gold can be divided to any fractional part, so the amount one gets can be fractional. Note that the amount each one would get is a monotonic and continuous function of $b$.

Now you have been called as an even wiser person to help divide the gold among the $n$ wives. Each wife should get no more than she claims. The division is called fair if for any two wives who claim $a_i$ and $a_j$ dinars of the inheritance and get $c_i$ and $c_j$ dinars, correspondingly, these values are the *fair way* to divide $c_i + c_j$ dinars between them.

Help the wives of the late king divide his inheritance.

## Input

The first line of the input contains $n$ — the number of wives of the king ($2 \le n \le 5000$).

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 5000$).

The third line contains an integer $s$ ($0 \le s \le a_1 + a_2 + \ldots + a_n$).

## Output

Output $n$ floating point numbers $c_1, c_2, \ldots, c_n$ — the amounts of gold each wife should get in a fair division.

For each pair of wives $i$ and $j$ the absolute or relative difference between their parts and their parts in the *fair way* to divide $c_i + c_j$ between them must not exceed $10^{-9}$. The sum of $c_i$ must be equal to $s$ with an absolute or relative error of at most $10^{-9}$.

It can be proved that a fair division always exists. If there is more than one solution, output any of them.

## Examples

| standard input | standard output |
|---|---|
| 3<br>10 20 30<br>10 | 3.33333333333333<br>3.33333333333333<br>3.33333333333333 |
| 3<br>10 20 30<br>20 | 5<br>7.5<br>7.5 |
| 3<br>10 20 30<br>30 | 5<br>10<br>15 |

# Problem I. Interactive Array Guessing

Time limit: 3 seconds
Memory limit: 512 megabytes

This is an interactive problem.

Jury has prepared $n$ non-empty integer arrays $a_1, a_2, \ldots, a_n$. Each array consists of at most 10 positive integers not exceeding 1000. The elements of each array are pairwise distinct.

You can make queries of the following kind: choose several indices $q_1, q_2, \ldots, q_m$, where $1 \le m \le n$ and $1 \le q_i \le n$. These indices do not have to be distinct. The testing system tells you the contents of arrays $a_{q_1}, a_{q_2}, \ldots, a_{q_m}$ in the same order. However, these contents are concatenated without any delimeters.

Your task is to find the contents of all $n$ arrays.

## Interaction Protocol

First, the testing system writes the integer $n$ — the number of arrays ($1 \le n \le 1000$).

Your solution shall print requests of two types:

- "`?` $m\ q_1\ q_2\ \ldots\ q_m$" corresponds to a query about the contents of arrays $a_{q_1}, a_{q_2}, \ldots, a_{q_m}$. The testing system responds with the total length of these arrays, followed by the contents of array $a_{q_1}$, followed by the contents of array $a_{q_2}$, ..., followed by the contents of array $a_{q_m}$.
- "`!` $k_1\ a_{1,1}\ \ldots\ a_{1,k_1}\ \ldots\ k_n\ a_{n,1}\ \ldots\ a_{n,k_n}$" tells that your program has determined the contents of all the arrays, where $k_i$ is the length of array $a_i$. The request is formed by the length of array $a_1$, followed by the contents of array $a_1$, followed by the length of array $a_2$, followed by the contents of array $a_2$, ..., followed by the length of array $a_n$, followed by the contents of array $a_n$.

Don't forget to flush the output after each request!

Your solution must issue exactly one request of the second type. It must be the last request, and the solution must terminate gracefully after issuing it.

Your solution is allowed to issue at most 20 requests of the first type.

## Example

| standard input | standard output |
|---|---|
| 3<br><br>5 1 1 2 2 1<br><br>4 1 2 1 1<br><br>2 1 2 | <br>? 3 1 2 3<br><br>? 3 1 3 1<br><br>? 1 2<br><br>! 1 1 2 1 2 2 2 1 |

# Problem J. Joined Vessels

Time limit: 3 seconds
Memory limit: 512 megabytes

John is doing physics practice at school. Today he is studying the law of communicating vessels. This law states that if we have a set of communicating containers with a homogeneous liquid, when the liquid settles, it balances out to the same level in all of the containers regardless of their shape and volume.

In the lab, John has a set of $n$ cylindrical vessels with a base area of one square decimeter and a height that we consider to be infinite. The vessels are numbered from 1 to $n$, and vessels $i$ and $i+1$ are communicating via a very thin bridge at a height of $h_i$ decimeters. All these heights are pairwise distinct.

The practice work contains $t$ independent experiments. In each experiment, all vessels are initially empty. In the $i$-th experiment, water is slowly put into vessel $a_i$, and the experiment finishes when any amount of water appears in vessel $b_i$. The result of the experiment is the total volume of water put into vessel $a_i$, measured in liters (or, equivalently, cubic decimeters).

Note that the law of communicating vessels can only be applied to vessels $i$ and $i+1$ when the water level is at least $h_i$ in both of them. Until then, if the water level reaches $h_i$ in just one of them, it stays constant and any excess water coming into this vessel flows through the bridge into the other one.

Help John check his results!

## Input

The first line of the input contains an integer $n$ — the number of vessels ($2 \le n \le 2 \cdot 10^5$).

The second line contains $n-1$ integers $h_1, h_2, \ldots, h_{n-1}$ — the heights of communication bridges between consecutive vessels, in decimeters ($1 \le h_i \le 10^9$). These heights are pairwise distinct.

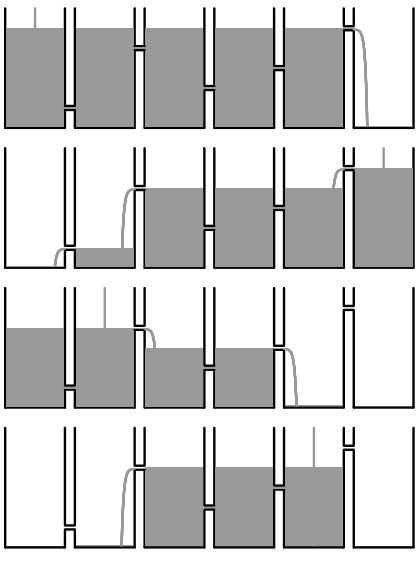
The third line contains an integer $t$ — the number of experiments ($1 \le t \le 2 \cdot 10^5$).

Each of the following $t$ lines contains two integers $a_i$ and $b_i$ — the numbers of the starting vessel and the target vessel in the $i$-th experiment ($1 \le a_i \le n$; $1 \le b_i \le n$; $a_i \ne b_i$).

## Output

For each experiment, in the order of input, output a single integer — the required volume of water, in liters.

## Example

| standard input | standard output | Illustration |
|---|---|---|
| 6<br>1 4 2 3 5<br>4<br>1 6<br>6 1<br>2 5<br>5 2 | 25<br>18<br>14<br>12 |  |

# Problem K. Keyboard Consensus

Time limit: 3 seconds
Memory limit: 512 megabytes

Famous young programmers Kolya and Kostya are preparing for the upcoming team contest. One of the important decisions they need to make is to pick a keyboard they both are comfortable with. To reach a consensus they have decided to use the following procedure.

There are $n$ keyboards numbered from 1 to $n$. Initially, the set of candidates contains all $n$ keyboards. The programmers take turns, Kolya goes first. On his turn, the programmer removes one keyboard from the set. The last keyboard remaining in the set will be chosen for the contest.

Each programmer has prepared a list containing all $n$ keyboards, sorted from the one he likes the most to the one he likes the least. Both Kolya and Kostya want to minimize the position of the chosen keyboard in their list. They both know the lists of each other.

Find the keyboard that will be chosen if both programmers play optimally, and all optimal first moves for Kolya — keyboards he can remove on his first turn to guarantee the best possible result for himself.

## Input

The first line of the input contains an integer $n$ — the number of keyboards ($2 \le n \le 100$). The second line contains $n$ distinct integers between 1 and $n$, inclusive — the numbers of keyboards in Kolya's list, in the order from the one he likes the most to the one he likes the least. The third line contains Kostya's list in the same format.

## Output

In the first line output one integer — the number of the chosen keyboard. In the second line output the number of optimal first moves for Kolya. In the third line output these moves in increasing order.

## Examples

| standard input | standard output |
|---|---|
| 5<br>1 2 3 4 5<br>5 4 3 2 1 | 3<br>2<br>4 5 |
| 4<br>1 2 3 4<br>1 3 2 4 | 1<br>3<br>2 3 4 |
| 3<br>3 1 2<br>1 3 2 | 3<br>1<br>1 |
| 4<br>4 1 3 2<br>1 3 4 2 | 1<br>3<br>2 3 4 |

## Note

In the first example, Kolya will remove keyboards 4 and 5 in any order, and Kostya will remove keyboards 1 and 2 in any order. Thus, keyboard 3 will be chosen.

In the second example, both of them like keyboard 1, so they will remove all other keyboards in any order.

In the third example, the only optimal move for Kolya is to remove keyboard 1 on his turn. Kostya will choose between keyboards 2 and 3, and since keyboard 3 is better for him, he will remove keyboard 2.

In the fourth example, a simple search through all possible moves shows that Kolya cannot force keyboard 4 to be chosen, and the answer is 1.

# Problem L. LED-led Paths

Time limit: 3 seconds
Memory limit: 512 megabytes

In a large city there are $n$ junctions and $m$ one-way streets connecting those junctions. It is known that one can't walk along the streets infinitely, as there is no directed cycle of streets in the city.

The tourism department has decided to install colored illumination using the LED (light-emitting diode) technology. Each street is to be illuminated with either red (`R`), green (`G`), or blue (`B`) color.

The official city maps for tourists will indicate the selected colors. All continuous paths illuminated by the same color will be suggested for walking and sightseeing. As LEDs will lead people along beautiful ways, these paths will be called LED-led paths.

The health department says that if there is a very long single-color LED-led path, some tourists might overestimate their strength, walk for too long, get too tired, and dislike the city.

Propose a three-color illumination plan in which no single-color LED-led path consists of more than 42 streets.

## Input

The first line of the input contains two integers $n$ and $m$ — the number of junctions and streets, respectively ($2 \le n \le 50\,000$; $1 \le m \le 200\,000$).

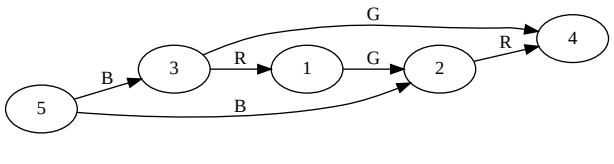
Each of the following $m$ lines contains two integers $u_i$ and $v_i$, denoting a one-way street from junction $u_i$ to junction $v_i$ ($1 \le u_i, v_i \le n$; $u_i \ne v_i$).

The given city is guaranteed to be acyclic. Each pair of junctions is connected by at most one street.

## Output

For each street, in the order of input, output its color on a separate line — a single letter `R`, `G`, or `B`.

## Example

| standard input | standard output | Illustration |
|---|---|---|
| 5 6<br>5 3<br>3 1<br>1 2<br>2 4<br>5 2<br>3 4 | B<br>R<br>G<br>R<br>B<br>G |  |

## Note

In this example, all single-color LED-led paths are not longer than one street (and therefore not longer than 42 streets), which is absolutely OK according to the health department.