Problem A. Element-Wise Comparison

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.25 seconds
Memory limit:	1024 mebibytes

You are given a permutation p of length n: it contains numbers $1, 2, \ldots, n$ in some order. Your task is to find the number of pairs of subarrays of length m such that the left one is element-wise smaller than the right one.

Formally, consider two subarrays: $p_i, p_{i+1}, \ldots, p_{i+m-1}$ and $p_j, p_{j+1}, \ldots, p_{j+m-1}$. When $i < j$, we say the first subarray is the left one, and the second is the right one; the subarrays may intersect. The first one is *element-wise less* than the second when m inequalities hold simultaneously:

$$p_i < p_j, \quad p_{i+1} < p_{j+1}, \quad \ldots, \quad p_{i+m-1} < p_{j+m-1}.$$

Input

The first line contains two integers n and m ($1 \le m \le n \le 5 \cdot 10^4$). The second line contains n integers separated by spaces: the permutation p itself.

Output

Output the only integer: the answer to the problem.

Examples

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

Explanations

In the second example, the answer 2 is formed from two pairs of subarrays:

- $\{3, 1\}$ is element-wise less than $\{4, 2\}$,
- $\{1, 4\}$ is element-wise less than $\{2, 5\}$.

In the third example, the answer 3 is formed from three pairs of subarrays:

- $\{1, 2\}$ is element-wise less than $\{2, 3\}$,
- $\{2, 3\}$ is element-wise less than $\{3, 4\}$,
- $\{1, 2\}$ is element-wise less than $\{3, 4\}$.

Problem B. Schoolgirls

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

Schoolgirl Alice learned about hinged mechanisms in her technology class. She constructed a tool that allows her to extend a parallelogram to a fourth vertex using three of its vertices (which may coincide or be collinear). Formally, given three points A, B, C, she can construct a point D such that the vectors $\overrightarrow{AB}$ and $\overrightarrow{DC}$ are equal.

Schoolgirl Alina learned about the concept of regular polygons in her geometry class. In this problem, we will use the following definitions:

- we will say that the points $A_1, A_2, \ldots, A_n$ ($n \geq 3$) *form a degenerate regular polygon* if all these points coincide;
- we will say that the points $A_1, A_2, \ldots, A_n$ ($n \geq 3$) *form a non-degenerate regular polygon in counterclockwise order* if they are all distinct, lie on the same circle with some center O, and $\angle A_1OA_2 = \angle A_2OA_3 = \ldots = \angle A_nOA_1 = \frac{360^\circ}{n} = \frac{2\pi}{n}$, and in all these angles, a counterclockwise rotation around O by $\frac{2\pi}{n}$ maps $\overrightarrow{OA_i}$ to $\overrightarrow{OA_{(i \bmod n)+1}}$;
- we will say that the points $A_1, A_2, \ldots, A_n$ ($n \geq 3$) *form a non-degenerate regular polygon* if there exists a permutation $A_{(1)}, A_{(2)}, \ldots, A_{(n)}$ of these points that forms a non-degenerate regular polygon in counterclockwise order;
- we will say that the points $A_1, A_2, \ldots, A_n$ ($n \geq 3$) *form a regular polygon* if they form a degenerate regular polygon or a non-degenerate regular polygon.

Note that the last definition is independent of the order of the points: if a list of points forms a regular polygon, then any permutation of them also forms a regular polygon.

Headmistress Arina decided to test the girls' skills. First, she gave them a task to construct $n+m$ points on the plane. The first n points should form a non-degenerate regular polygon in counterclockwise order. Each of the next m points is constructed using Alice's tool based on three previous points.

The girls coped with this part of the task. Then Arina started naming certain sets of points and asking whether they form a regular polygon. This turned out to be quite difficult for the schoolgirls, so they turned to you for help. Write a program that can handle Arina's task.

Input

The first line contains three integers n, m, k: the number of vertices in the original regular polygon, the number of additional points constructed using Alice's tool, and the number of polygons Arina will ask about ($3 \leq n \leq 10^4$, $0 \leq m \leq 3 \cdot 10^4$, $1 \leq k \leq 10^4$). The points $K_1, K_2, \ldots, K_n$ form a non-degenerate regular polygon in counterclockwise order.

The next m lines describe how the points $K_{n+1}, \ldots, K_{n+m}$ are constructed. The i-th line contains three integers a_i, b_i, c_i ($1 \leq a_i, b_i, c_i \leq n+i-1$): the numbers of the three points to which Alice's tool is applied. Point K_{n+i} is defined such that $\overrightarrow{K_{a_i}K_{b_i}} = \overrightarrow{K_{n+i}K_{c_i}}$. Some or all of the numbers a_i, b_i, c_i may coincide.

The next k lines describe Arina's sets of points. The i-th line describes the i-th set in the format "r_i $P_1^{(i)}$ $P_2^{(i)}$ $\ldots$ $P_{r_i}^{(i)}$". This means that the schoolgirls need to check whether the points $K_{P_1^{(i)}}, \ldots, K_{P_{r_i}^{(i)}}$ form a regular polygon ($3 \leq r_i \leq 3 \cdot 10^4$, $1 \leq P_j^{(i)} \leq n+m$). It is guaranteed that the sum of all r_i is at most $3 \cdot 10^4$. Some or all of the numbers $P_j^{(i)}$ may coincide.

Output

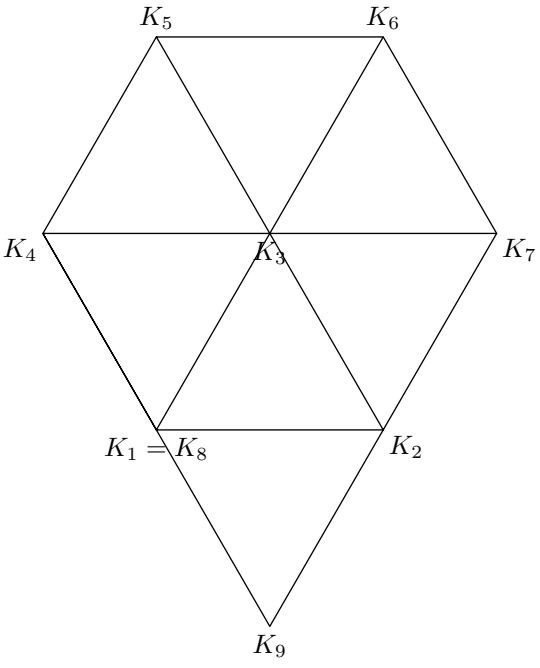
Output k lines. The i-th line should contain the word "`Yes`" if Arina's i-th set forms a regular polygon, and "`No`" otherwise. Each letter of the output can be in any case (uppercase or lowercase).

Example

standard input	*standard output*
3 6 8 1 2 3 3 1 4 5 4 3 3 1 2 4 5 3 4 5 2 6 4 7 6 5 1 2 3 1 3 2 3 1 1 8 4 2 5 6 7 3 2 1 4 3 6 5 9 3 4 7 9 4 1 3 2 8	Yes Yes Yes No No No Yes No

Note

Picture for the example:



Problem C. Cherry Picking

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

Statistics expert Vladimir Borisovich periodically analyzes games of different chess players for interesting series. He considers a series interesting if it consists of 40 consecutive wins against opponents with ratings of at least 2900. In between, the player may have played any number of games against opponents with lower ratings: the important part is that 40 games against high rated opponents ended in a win.

Every time Vladimir Borisovich finds an interesting series, he shares it on social networks. Unfortunately, Vladimir Borisovich performs all operations manually, so it takes him a lot of time to search for interesting series. Help Vladimir Borisovich automate the process of finding interesting series by solving a more general problem.

You will be given the results and ratings of opponents in n consecutive games of a chess player. Your task is to find the maximum possible rating x such that, if you remove all games with opponent ratings strictly less than x from the sequence, there exists a series of k consecutive wins.

Input

The first line of input contains two integers n and k ($1 \le k \le n \le 100\,000$). The second line contains n space-separated integers r_i: the ratings of the opponents ($1 \le r_i \le 100\,000$). The third line consists of n characters which are zeros and ones: the i-th character is equal to "1" if the i-th game was a win, or "0" otherwise.

Output

If the answer exists, output x. Otherwise, output 0.

Examples

standard input	*standard output*
5 2 1 2 3 4 5 01101	2
5 2 3 4 5 2 1 10101	0

Problem D. Dwarfs' Bedtime

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

This is an interactive problem.

On the edge of a forest, there is a small house. Snow White and n dwarfs live there.

It is known that each dwarf sleeps continuously for exactly one half of the day, and this half starts at the same time each day. For the other half of the day, the dwarf stays awake.

As the lady of the house, Snow White wants to know, for each dwarf, the exact minute when the dwarf goes to bed. At each of the 1440 minutes during the day, Snow White can check the beds of any dwarfs and learn whether they are asleep or awake. However, Snow White can check the bed of each dwarf no more than 50 times: otherwise, the dwarf will be outraged by the invasion of his privacy.

Please help Snow White to learn during one day, for each dwarf, the exact minute the dwarf goes to bed.

In each test, the dwarfs' schedule is fixed in advance and does not change during the interaction. In other words, the interactor in this problem is not adaptive.

Interaction Protocol

First, read a separate line containing an integer n: how many dwarfs live in the house ($1 \le n \le 100$).

Snow White uses a clock with 24-hour display. The interaction starts at `00:00` and ends at `23:59`. To check at moment `HH:MM` (hours from `00` to `23`, minutes from `00` to `59`) whether the i-th dwarf is sleeping, print a separate line formatted as "`at HH:MM check` i". In response, you will get a separate line: "`asleep`" if at this minute the i-th dwarf is sleeping, or "`awake`" if he is awake. Each next query must happen at a moment no earlier than the previous.

To output the answer, print a separate line "`answer`", followed by n separate lines: the times when the first, second, ..., n-th dwarf goes to sleep, formatted as `HH:MM`. After that, terminate your program.

If your solution performs too many checks for some dwarf or prints a wrong answer, and then terminates immediately, it will get `WA` (`Wrong Answer`). Remember to print the newline and flush the output buffer after each check and after the answer, or your solution will get `IL` (`Idleness Limit Exceeded`).

Example

standard input	*standard output*
2 asleep awake asleep awake awake	 at 01:40 check 1 at 01:40 check 2 at 07:59 check 1 at 08:00 check 1 at 13:41 check 2 answer 20:00 01:41

Explanation

In the example, $n = 2$ dwarfs live in the house.

The first dwarf sleeps from `20:00` to `07:59` inclusive. We know that because he was sleeping at `07:59`, but awake at `08:00`. This means the dwarf awoke at exactly `08:00`. Therefore, he goes to sleep exactly 12 hours after that.

The second dwarf sleeps from `01:41` to `13:40` inclusive. We know that because he was awake both at `01:40` and at `13:41`. The period from minute `13:41` to minute `01:40` inclusive spans exactly half a day. Therefore, these are the first and the last minute of being awake for him.

Please note that Snow White can check the beds of multiple dwarfs at the same minute (`01:40` in the example).

The empty lines are added only for readers' convenience. In the real input and output, there are no empty lines.

Problem E. Fake Coin and Lying Scales

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

Suppose you have c coins. One of them is fake, and is lighter than a real coin. All other coins are real, and have the same weight. You also have two-pan balance scales. In one weighing, you can put some coins on one pan and some coins on the other pan (and the remaining coins on neither pan), and see which pan is lighter, if any. Your task is to determine which coin is fake.

Sounds familiar, right? Well, there are good news and bad news. The bad news is that the scales may lie to you up to k times, and you have no way to know which weighing results are true. The good news is that you can make up to $3k$ mistakes. That is, you can make up to $3k + 1$ guesses, and you will win if at least one of them is correct.

Let $f(n, k)$ be the maximum number c such that, if you have c coins and the scales may lie up to k times, and you can make up to $3k$ mistakes, there exists a weighing strategy such that you can win by making no more than n weighings, whatever results you get.

You should find $f(n, k)$ **approximately**. In particular, output $\ln f(n, k)$, where ln is the natural logarithm. Your answer will be considered correct if the **absolute** difference between your answer and the correct one is no more than **10**.

Input

The first line contains an integer t, the number of test cases ($1 \le t \le 10^5$).

Each of the next t lines contains two integers n and k ($1 \le n \le 10^9$; $0 \le k \le 10^9$).

Output

For each test case, print a line with a single real number: the answer $\ln f(n, k)$ for that test case. The answer will be accepted if it differs from the right one by at most 10.

Example

standard input	*standard output*
2 100 0 100 1	109.8612289 106.1174552

Note

In the first test case, $f(100, 0) = 3^{100}$. Here, the answer is $\ln(3^{100}) = 100 \cdot \ln(3) = 109.8612289\ldots$.

Natural logarithm (logarithm with base $e = 2.71828182845904523536\ldots$) can be calculated using `log` in C++, `math.log` in Python, and `Math.log` in Java.

Problem F. The Whole World

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

We will call a polynomial with real coefficients *whole* if it takes integer values at all integer points.

You are given n pairs of integers (x_i, y_i). You should find the minimal possible degree of a *whole* polynomial f such that $f(x_i) = y_i$ for $i = 1, 2, \ldots, n$.

In this problem, consider the polynomial $f(x) = 0$ to have degree 0.

Input

The first line of the input contains an integer T ($1 \le T \le 100$) denoting the number of test cases.

Each test case starts with a line containing an integer n ($1 \le n \le 30$) denoting the number of points. Then n lines follow, each one containing a pair of integers (x_i, y_i) ($1 \le x_i \le 30$, $-10^9 \le y_i \le 10^9$).

Output

For each test case, print a line with a single integer: the answer for that test case.

Example

standard input	*standard output*
2 2 1 0 4 1 3 1 1 4 4 6 6	3 1

Problem G. Unusual Case

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Sir Hamilton loves long walks...

You are given a random undirected graph with n vertices and m edges: this graph is chosen randomly and uniformly from all graphs with this number of vertices and edges. You are also given an integer k. Your task is to find k non-intersecting Hamiltonian paths in the given graph.

A path is called Hamiltonian if it passes through all vertices of the graph exactly once.

In this problem, there is one example and exactly 30 more tests. In all tests except the example, $n = 10\,000$, $m = 200\,000$, and $k = 8$.

Input

The first line contains three integers: the number of vertices n, the number of edges m, and the number of paths k to be found.

Then m lines contain the description of the graph's edges. Each edge is described by a pair of integers from 1 to n: the numbers of the vertices it connects. The graph has no loops and no multiple edges.

Output

Output k lines. In each line, output a Hamiltonian path: a sequence of n vertices in the order of traversal. Each edge of the graph can be used in at most one of the output paths. If there are several possible answers, output any one of them. It is guaranteed that an answer exists for each of the given tests.

Example

standard input	*standard output*
5 9 2 1 3 1 4 1 5 2 3 2 4 2 5 3 5 4 3 5 4	1 3 5 2 4 5 1 4 3 2

Note

The example in the statement is the only test where $n \neq 10\,000$, $m \neq 200\,000$, $k \neq 8$. It is used to demonstrate the input and output format. The example is test 1 in the testing system.

Problem H. Page on vdome.com

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

On the website `vdome.com`, each new user is assigned a page with an address of the form `vdome.com/id`K, where K is the sequential number of this user, starting from the creation of the site. For example, the site creator has the address `vdome.com/id1`, and his assistants and friends have addresses like `vdome.com/id2`, `vdome.com/id6`, and so on.

The site quickly gained popularity, and now, years later, it has about 10^9 users. Therefore, owners of pages with small numbers are periodically contacted by strangers asking to buy their page, the smaller the number, the more attractive the address. Moreover, some have managed to change `vdome.com/id`K to a letter address of the form `vdome.com/namelogin`. Therefore, there are some "gaps" in the row of numeric addresses.

John decided to find himself an address with the smallest possible number among those currently available. To do this, he obtained a database of all active user addresses of the form `vdome.com/id`K from the darknet, and is trying to find the smallest number missing from it. Since there are too many records in the received database, he wrote a simple program for searching. But John is just a beginner programmer, and therefore, when he took a slice from the string-address of each record, he accidentally reversed the string, and only then read it as an integer. As a result, for example, from the string `vdome.com/id`12345 he got 54321, from `vdome.com/id`67500 he got 576, and so on. That is, all numbers were written backwards, leading zeros disappeared, and among what was obtained, John is looking for the smallest number.

He could have avoided all this if he knew that, even if a user has a letter address, any page on `vdome.com` can still be accessed by its number, and therefore all the numbers from the first to the N-th were in the database, without gaps. But John didn't know any of this. Therefore, for a given N (the number of address records in the database), answer what is the smallest missing number his program will output.

Input

The first line specifies an integer N — which represents the number of records in the database ($1 \le N \le 10^9$).

Output

Print the page number that John's program will output.

Examples

standard input	*standard output*
5	6
10	10

Explanations

In the first example, the database contained records `id1`, `id2`, `id3`, `id4`, `id5`. After processing, they turned into a list of numbers 1, 2, 3, 4, 5: single-digit numbers remained unchanged when reversed. The smallest missing number among them is 6.

In the second example, the database contained records `id1`, `id2`, `id3`, `id4`, `id5`, `id6`, `id7`, `id8`, `id9`, `id10`. After processing, they turned into a list of numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 1: `id10` first transformed into `01`, and when converted into an integer, it became the number 1. The smallest missing number among them is 10.

Problem I. Spin & Rotate!

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

There is a large square $ABCD$ (vertices are listed in counterclockwise order). In each of its vertices, there is a peg. There are also two long ropes, one connecting vertices A and B and the other one connecting vertices C and D. Each peg pins its rope end to the ground. The ropes can stretch and contract indefinitely, but they should never leave the area above the square. Initially, the ropes are *unentangled*: that is, if one shrinks them enough, then one of them lies along one of the sides of the square while the other one lies along the opposite side of the square.

There is a robot called Ka-BAN which can execute two instructions:

- spin, denoted as `S`: all four ropes are detached from the pegs, spun 90° counterclockwise around the vertical axis, keeping all the entanglement between the ropes, and reattached to the new pegs. Alternatively, one can also illustrate this operation as if the square, pegs, and ropes stay still, but instead the names of the vertices spin 90° clockwise.
  - the new name of vertex A is B;
  - the new name of vertex B is C;
  - the new name of vertex C is D;
  - the new name of vertex D is A;

  For the purposes of this problem, these two operations are equivalent.
- rotate, denoted as `R`: Ka-BAN detaches ropes from pegs A and D, swaps them, and reattaches them back. During the swap, the rope end initially attached to A is passed over the rope end initially attached to D. If one stands outside the square near the side AD and observes this process, it looks like the end A and end D rotate 180° around each other counterclockwise.

You have a string of letters "`S`" and "`R`" which is a program executed by Ka-BAN: the instructions were performed one by one from left to right. Unfortunately, after this process, the square and ropes seem like an incomprehensible mess. To fix that, you bought a new robot Iz-BAN. When you read the manual, you got unpleasantly surprised: Iz-BAN had the same set of two instructions as Ka-BAN! That meant that the unentanglement promised to be difficult: even though any spin can be undone with three more spins, there seemed to be no easy way to undo a rotate.

Nevertheless, it can be proven that these two operations are enough to fix the situation! Given the program executed by Ka-BAN, write the shortest program for Iz-BAN such that, after the execution of this program, the ropes again become unentangled.

Input

The first line contains an integer T, the number of test cases ($1 \le T \le 300\,000$).

Each of the next T lines contains one string s consisting of letters "`S`" and "`R`": the program executed by Ka-BAN ($1 \le |s| \le 300\,000$). The sum of the lengths of strings among all test cases does not exceed 300 000.

Output

For each test case, print one line consisting of letters "`S`" and "`R`": the shortest program that can be given to Iz-BAN to unentangle the ropes. If the ropes are already unentangled, print one letter "`S`" instead. It can be proven that the answer always exists and is unique.

Examples

standard input	*standard output*
14 R S RR SR RS SS RRR SRR RSR SSR RRS SRS RSS SSS	SR S SRSRR S R S SRSRRSRR S S SR RSRR S SR S
5 SRRSRRSRR SRRRSRRR RRRSRRRSRRR SRRSRRSRRSRRSRRSRRSRRSRRSRR SRRRSRRRSRRRSRRRSRRRSRRRSRRRS	SRSRRR SRSRRSRR SRSRRSRRRSRRRSRR SRRRRRRRRR RSRRSRRRSRRRSRRRSRRRSRRRSRR

Problem J. First Billion

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

You are given N positive integers. Your task is to select some of the given numbers so that the sum of the selected numbers is exactly 10^9.

To make it a little easier for you to solve the problem, the jury promises that the tests will be organized as follows.

1. Two random sets of exactly n positive integers with the same sum 10^9 are generated (a uniform distribution is taken over all sets with sum 10^9).
2. You are given $N = 2n$ generated elements in random order. The value $N = 2n \leq 100$.

Input

The first line contains a positive integer $N = 2n \leq 100$.

The second line contains N positive integers $a_1, \ldots, a_N$.

There are exactly 100 tests in the problem.

Output

Output a line describing a set with sum equal to 10^9: first the number of indices k, then k different indices $i_1, \ldots, i_k$ for which $\sum_j a_{i_j} = 10^9$ is satisfied.

It is not necessary to find a set of size exactly n.

Example

standard input
10 386413329 88494216 245947398 316438989 192751270 204627269 65749456 3938400 150458676 345180997
standard output
5 2 3 4 8 10

Problem K. Tasks and Bugs

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

Nikolai, a responsible head of the development department, wants to complete tasks as quickly as possible, but without any bugs. To achieve this, he keeps tasks and bugs in the Jira tracking system. The identifiers for tasks and bugs are represented in the format `CS-`X where X is a number containing from 1 to 5 decimal digits. To expedite task finalization, Nikolai needs to understand how many unresolved bugs are associated with each task.

Using a special filter in the tracking system, he compiles a list of unresolved bugs, and for each bug a non-empty list of tasks linked to this bug. Nikolai runs the filter and obtains a list of unresolved bugs and their associated tasks:

```
CS-20: CS-1
CS-100: CS-239
CS-300: CS-239, CS-11111
```

Bug `CS-20` is linked to task `CS-1`, bug `CS-100` is linked to task `CS-239`, and bug `CS-300` is linked to both tasks `CS-239` and `CS-11111`. The bugs, as well as the tasks associated with each bug, are listed in ascending **numerical** order of identifiers (identifiers are sorted as numbers). Bug identifiers are different from task identifiers.

Now, Nikolai regroups the tasks and bugs by sorting them in numerical order. After the reorganization, the result looks as follows:

```
CS-1: CS-20
CS-239: CS-100, CS-300
CS-11111: CS-300
```

Task `CS-1` is linked to bug `CS-20`, task `CS-239` is linked to bugs `CS-100` and `CS-300`, and task `CS-11111` is linked to bug `CS-300`. Tasks, as well as the bugs associated with each task, are listed in ascending **numerical** order of identifiers (identifiers are sorted as numbers).

However, Nikolai doesn't want to spend time doing this reorganization. Write a program that will do it for him.

Input

The input consists of one or more lines. Each line contains a name of a bug and a list of tasks associated with that bug. The bug is separated from tasks by a colon and a space. Every two tasks in the list are separated from each other by a comma and a space.

Bugs, as well as the tasks associated with each bug, are listed in ascending numerical identifier order.

The number of bugs ranges from 1 to 100, and the number of tasks for each bug ranges from 1 to 10.

Output

Print one or more lines: for each task, list the bugs associated with it. Follow the same format and use the same order as in the input.

Example

standard input	*standard output*
CS-20: CS-1 CS-100: CS-239 CS-300: CS-239, CS-11111	CS-1: CS-20 CS-239: CS-100, CS-300 CS-11111: CS-300

Problem L. Candies

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Suppose you have numbers x_1, x_2, and x_3. Initially, all of them are zeroes. In one step, you may increase one of the numbers by 1. The only condition is that, at any time, x_1 should be the greatest among x_1, x_2, and x_3 (formally, $x_1 \geq x_2$ and $x_1 \geq x_3$). In how many ways can you reach $x_1 = a$, $x_2 = b$, and $x_3 = c$ in the end? Two ways are different if they differ in at least one step.

Since the answer can be very large, output it modulo the prime number 998 244 353.

Input

The first line contains three integers a, b, and c, where $1 \leq b, c \leq a \leq 10\,000$.

Output

The output should contain one number, which is equal to the number of ways modulo the prime number 998 244 353.

Example

standard input	*standard output*
4 3 2	368

Problem M. Toilets

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

There is an office in the form of a circle of length L. There are several toilets located in different places in the office. Each employee has a predetermined time t_i when they get up from their workplace with coordinate p_i on the circle and start walking around the office at a unit speed in a predetermined direction until they find an available toilet, after which they will occupy it for a duration of d_i. If an employee encounters an occupied toilet, they will pass by it. In the event that multiple employees simultaneously reach an available toilet, it will be occupied by the employee who started their journey earlier (it is guaranteed that t_i are monotonically increasing). A toilet can be occupied by the next employee at the same time when the previous one vacates it.

For each employee, find when and to which toilet they will get.

Input

The first line contains three numbers n, m, and L ($1 \le n, m, \le 2 \cdot 10^5$, $1 \le L \le 10^{12}$): the number of employees, the number of toilets, and the length of the office, respectively.

The second line contains m numbers x_i ($0 \le x_i < L$): the coordinates of the toilets. It is guaranteed that all x_i are distinct.

The following n lines describe the employees, one per line. Each employee is described by four values t_i, p_i, s_i, d_i ($0 \le t_i < 10^{18}$, $0 \le p_i < L$, $1 \le d_i \le 10^{12}$, $t_i < t_{i+1}$). The character s_i is either "+" or "-" (without quotes) depending on whether the i-th employee moves in the direction of increasing or decreasing coordinates. It is guaranteed that all input numbers are integers.

Output

Output n lines, each containing two numbers. The i-th line should contain the coordinate of the toilet that the i-th employee will occupy and the time when it will happen.

Example

standard input	*standard output*
5 3 45 10 20 30 0 0 + 200 2 5 + 10 20 40 - 100 21 16 + 10 50 0 + 22	20 20 10 7 30 30 10 60 10 105

Problem N. (Un)labeled graphs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

This is a run-twice problem.

Old man Pafnutiy wants to send a *labeled graph* G on n vertices to old man Infinitiy. Two labeled graphs on the set of vertices $[n] = \{1, 2, \ldots, n\}$ are different if and only if there are two distinct integers $i, j \in [n]$ such that vertices i and j are adjacent in one of the graphs but not in the other one.

Unfortunately, old man Pafnutiy's old computer is only capable of storing unlabeled graphs. Two unlabeled graphs on the set of vertices $[m] = \{1, 2, \ldots, m\}$ are different if and only if there is no permutation $\pi \in \mathbb{S}_m$ such that, if one renumbers the vertices of one of the graphs according to this permutation, then the two graphs become equal as labeled graphs. Therefore, if one tries to enter a labeled graph into old man Pafnutiy's old computer, the latter somehow shuffles all its vertices.

Old man Pafnutiy decided to send a bigger graph to old man Infinitiy: that is, if G has n vertices, old man Pafnutiy will send some graph H on $m = f(n) = n + \lceil \log_2 n \rceil + 3$ vertices. Then old man Infinitiy will only have to somehow decode the received graph H' back into G, given that H and H' are equal as unlabeled graphs. But how can they organize this transmission?

Input

The first line contains two integers v_1 and v_2: the number of vertices in the input and output graphs ($3 \le \min\{n_1, n_2\} \le 2024$). The next v_1 lines contain binary strings of length v_1: the adjacency matrix of the input graph.

The graph is undirected, so the adjacency matrix is symmetric. Additionally, it is guaranteed that there are no self-loops.

Output

Print v_2 binary strings of length v_2: the adjacency matrix of the output graph.

If $v_1 < v_2$, then $v_1 = n$, $v_2 = m = n + \lceil \log_2 n \rceil + 3$. In this case, you should act as old man Pafnutiy: you receive graph G and should output graph H.

Otherwise, $v_2 = n$, $v_1 = m = n + \lceil \log_2 n \rceil + 3$. In this case, you should act as old man Infinitiy: you receive graph H' (which is obtained from H after applying some permutation to its vertices) and should output the initial graph G.

The graph should be undirected, so the adjacency matrix should be symmetric. Additionally, there can be no self-loops in the graph.

Examples

standard input	*standard output*
5 11 01110 10110 11001 11001 00110	0001000000 0000000000 0000000000 1000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
11 5 01000000000 10000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000	01110 10110 11001 11001 00110

Problem O. Mysterious Sequence

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

In the ancient kingdom of numbers X, a mysterious formula ruled, connecting each number with the previous ones:

$$X_{i+2} = A \cdot X_{i+1} + B \cdot X_i$$

This formula was so powerful that even the wisest magician could not decipher all the numbers in the sequence without knowing the first two numbers.

However, over time, all members of the sequence were lost except for the first and the last numbers.

Once upon a time, a wizard of the highest category was asked to find all the numbers X_i in this mysterious sequence. He embarked on a long journey, deeply contemplating incantations and mathematical formulas. With his knowledge and magic, he delved into the secrets of numbers and finally revealed all the hidden X_i contained in this mystical formula.

And what about you? Can you reconstruct all the members of the sequence knowing only the first and last numbers?

Input

The first line contains real numbers A and B ($0.25 \le A, B \le 1$, up to two digits after the decimal point) and integers n, X_1, X_n ($2 \le n \le 10$, $1 \le X_1, X_n \le 100$).

Output

Output n lines: the numbers X_1, X_2, ..., X_n, one per line. The answers will be considered correct if absolute or relative error is at most 10^{-6}.

Example

standard input	standard output
1.0 1.0 10 1 10	1 -0.3235294118 0.6764705882 0.3529411765 1.029411765 1.382352941 2.411764706 3.794117647 6.205882353 10