Problem A. Archaeology

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

This is an interactive problem.

In one of the N^2 integer points of the square $\mathbb{D} = [1; N] \times [1; N]$, a mysterious artifact is hidden. Your task is to find this artifact. You have a radar, and its single use is arranged as follows:

- First, you place the radar at any integer point belonging to $\mathbb{D}$.
- If the artifact is exactly at that point, the radar lights up green, the coordinates of the point where the artifact and the radar are located will be displayed on the screen, and the archaeological excavation ends there.
- Otherwise, suppose you placed the radar at point R, and the artifact is at point $A \neq R$. Then the radar will light up yellow and display the coordinates of an integer point $B \in \mathbb{D}$ such that $\angle ARB < 90°$. Among all such integer points, the radar will display one uniformly at random; in particular, if $A \neq R$, there is always a chance that the radar will show the coordinates of point A (since $\angle ARA = 0° < 90°$), but it will never show point R.

You can assume that, except for the first five tests (described in the Notes section), among all possible points in the square $\mathbb{D}$, the artifact is hidden at a uniformly selected random point, which is fixed in each test and does not depend on the responses of your program. All the random choices described above are made independently.

Your radar is not fully charged, so you can use it no more than 90 times. Write a program that will indicate where to place the radar so that it eventually lights up green.

Interaction Protocol

The first line contains a single integer N: the side of the square in which the artifact is hidden ($1 \le N \le 10^9$). Then your program should print requests to use the radar. Each request consists of two integers, R_{xi} and R_{yi}: the coordinates of the next point where you want to place the radar ($1 \le R_{xi}, R_{yi} \le N$). After each request, you must print a newline and flush the output buffer. After that, you can read two integers, B_{xi} and B_{yi}: the coordinates of the point displayed on the radar screen ($1 \le B_{xi}, B_{yi} \le N$). If $B_{xi} = R_{xi}$ and $B_{yi} = R_{yi}$, you have successfully passed the test, and the program should terminate immediately to receive the `OK` verdict on this test. Otherwise, if you still have not made 90 requests, you should continue making requests.

Your program should terminate immediately if, after making 90 requests, the radar has not yet lit up green, to receive a `Wrong Answer` verdict; otherwise, the verdict may be unpredictable.

Examples

standard input	*standard output*
1 1 1	1 1
2 2 2 1 2 1 2 2 2 1 1 1 2 1 1 1 2	1 1 1 1 2 1 2 1 2 1 2 2 2 2 1 2

Note

In this problem, there are exactly 50 tests, including examples.

Test 1 coincides with the first example.

In tests 2–5, $N = 2$. The artifact in the second test is hidden in the point $(1, 2)$ as shown in the second example; however, it is not guaranteed that in the second test, the interactor will provide the same responses to requests as in the example. Tests 2–5 are guaranteed to feature all four possible artifact places.

In the next 28 tests, N consecutively goes through numbers $4, 8, 15, 30, \ldots, \frac{10^9}{4}, \frac{10^9}{2}$. Formally, for each $i \in \{6, \ldots, 33\}$, in the i-th test $N = \lceil 10^9 / 2^{34-i} \rceil$. In each of these tests, the artifact is hidden at a uniformly selected random point.

In the next 17 tests, N consecutively goes through numbers $10^9 - 5, \ldots, 10^9 - 1, 10^9, 10^9, \ldots, 10^9$. Formally, for each $i \in \{34, \ldots, 50\}$, in the i-th test $N = 10^9 + \min\{0, i - 39\}$. In these 17 tests, the artifact is also each time hidden at a uniformly selected random point.

If several submissions are made and in some test the corresponding programs make the same queries, they will receive identical answers from the interacting jury program.

Problem B. Glass Stepping Stones

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Alice and Bob are playing a game on a string of letters "L" and "R" (with ASCII codes 76 and 82, respectively). They take turns, starting with Alice. On their turn, a player removes one character from the string.

If the resulting string has no substring "LR", Alice wins; if the resulting string has no substring "RL", Bob wins. However, if both events occur simultaneously, the game is declared a draw.

The previous paragraph also applies to the starting string: for example, if it contains the substring "RL", but does not contain the substring "LR", then no one will make any moves because Alice automatically wins.

What will be the outcome of the game if both play optimally? How should Alice play to achieve this outcome?

Input

The first line contains an integer T, the number of test cases ($1 \le T \le 10^6$). The following T lines describe the test cases themselves.

Each test case is represented by a string s consisting of the letters "L" and "R" ($1 \le |s| \le 10^6$).

The sum of lengths of the strings s across all test cases does not exceed 10^6.

Output

For each test case, print "Alice", "Bob" or "Draw" (case-insensitive) indicating either the winner of the game or that there will be no winner. On the same line, output an integer a describing any optimal move for Alice. The value a is the position of a character in string s to remove on the first turn ($1 \le a \le |s|$). The move is optimal if it does not change the outcome of the game when both players play optimally afterwards. If the game ends before the first move, print $a = 0$ instead.

Example

standard input	*standard output*
16	Draw 10
LRRRRLRRLRRRLLLRRL	Draw 0
L	Draw 0
R	Draw 0
RRRRRRRRR	Draw 0
LLLLLLLLLLLLLLL	Alice 1
LRLL	Draw 2
RLRR	Alice 2
RLRL	Bob 0
LLLLRRR	Alice 0
RL	Bob 0
LR	Bob 2
LRLR	Draw 5
LRLLR	Alice 1
LRLRL	Bob 2
LLLLLLLLLLRRRRRRRRLLLLLLRRRR	Draw 30
LLLLLLLLLLRRRRRRRRLLLLLLLLLLRRRR	

Problem C. Elegia's Mind

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	2048 mebibytes

The time and memory limits are quite strict.

There are n identical cubes, with each face painted in one of six different colors. Colors are numbered with integers from 0 to 5 inclusive. All the cubes are identical up to rotations in the three-dimensional space. Specifically, all cubes can be rotated in such a way that the front face is color 0, the top face is color 1, the right face is color 2, the left face is color 3, the bottom face is color 4, and the back face is color 5.

Suppose that we have arranged all these cubes in a line from left to right. Such an arrangement naturally produces the following six sequences of colors of length n: the colors on the front/top/right/left/bottom/back faces of the cubes, when viewed from left to right. How many ways are there to arrange the cubes so that all six said sequences are within the specified set of allowed sequences?

As the cubes are identical, it does not matter which cube is placed first in the sequence, which is placed second, and so on. Hence, there are 24^n arrangements in total: there are 24 ways to rotate each of the cubes. Of course, not all possible arrangements satisfy the condition on the colors of the faces.

The set of allowed sequences is given by an explicit string of 6^n characters "`0`" and "`1`". The sequence (from left to right) of colors $a_0, a_1, \ldots, a_{n-1}$ (where each a_i is an integer between 0 and 5 inclusive) is allowed if and only if the $\sum_{i=0}^{n-1}(a_i \cdot 6^i)$-th character (in 0-indexation) of the string is "`1`".

Despite the fact that the answer to the problem is not very large, output it modulo the prime number 998 244 353.

Input

The first line contains an integer n ($1 \le n \le 9$), the number of cubes. The second line contains a string of 6^n characters "`0`" and "`1`", with the k-th character (in 0-indexation) being "`1`" if and only if the base-6 notation of k is allowed when seen as a sequence of colors.

Output

One integer: the answer to the problem modulo 998 244 353.

Examples

standard input	*standard output*
1 111111	24
1 110111	0
2 000001001000010000000010000100100000	24
2 111111111111111111111111111111111111	576

Problem D. Preparation for the Exam

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Based on real events.

Misha sat down to prepare for the philosophy exam. The exam will take place in t hours and will consist of n parts. In the i-th part, there are q_i questions, each of which will take one hour to prepare for. Misha will receive n questions on the exam: one random question will be independently and uniformly selected from each part. To pass the exam, he must answer all n questions correctly. Misha can strategically prepare for all t hours and learn some subset of the questions. What is the maximum probability of passing the exam that he can ensure?

Input

The first line contains two integers t and n: the remaining time until the exam and the number of parts ($1 \le t \le 10^9$; $1 \le n \le 10^5$). The next line contains n integers $q_1, \ldots, q_n$: the number of questions in each part of the exam ($1 \le q_i \le 10^9$).

Output

Let $p = k/\ell$ be the desired probability represented as an irreducible fraction. Output p as a fraction modulo $M = 10^9 + 7$ as two integers: numerator and denominator.

Formally, output any two integers x and y ($-2^{63} \le x, y \le 2^{63} - 1$) such that $x\ell - yk$ is divisible by M, but y is not divisible by M. It is guaranteed that such numbers always exist.

For example, if $0 \le k \le \ell \le 2^{63} - 1$, you can just print "$k$ ℓ".

Examples

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

Note

If you are more accustomed to problems where the answer requires finding a residue x of a rational number modulo a prime, you can simply output "x 1", and this answer will be accepted. In particular, in the second example, the answers "1 2", "-1 -2", "1000000006 -1000000009", "500000004 1", and many others will be accepted.

Problem E. Fractal Maze

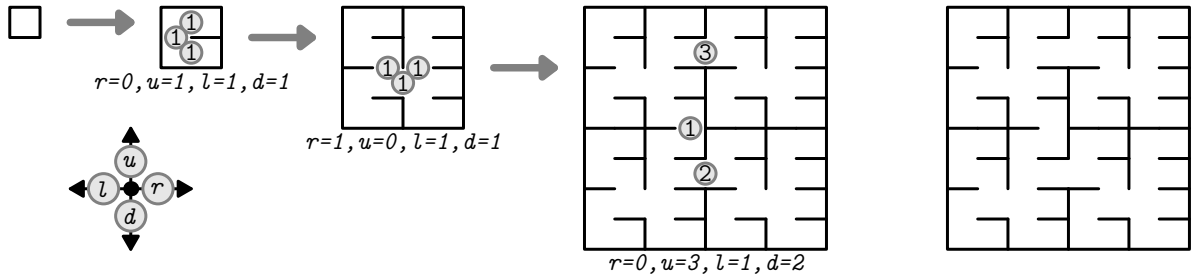
Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Let us construct a maze consisting of squares and walls between them. Start with a single square surrounded by walls. Then expand the maze several times.

An expansion goes as follows. Glue together four copies of the maze, forming a 2×2 square of copies. Now, imagine standing at the very center of the new maze, in a point between four squares. Consider the four long walls which run from the center, separating the neighboring copies. In three of these walls, we make a passage, one square wide. The fourth wall is left intact.

Let us write down a specific expansion. Assign four numbers to the walls from the center: r to the wall going right, u to the wall going up, l to the wall going left, and d to the wall going down. One of these numbers will be zero. The other three determine where exactly we make a passage: the number of the unit segment of the wall, counting from the center. For example, 1 means a passage next to the center, 2 denotes the second unit segment from the center, and so on.

The figure shows the construction of the maze from the example, along with the result.



The maze we got has exactly one simple path between any two squares; recall that a path is simple if it visits each square at most once. Answer q questions of the following form: given the coordinates of squares A and B, find the length of the simple path between them. The length of a path is the number of steps to the neighboring square.

Input

The first line contains an integer n ($1 \le n \le 30$). The next n lines define how the maze is built. Each line describes an expansion and contains four integers: r, u, l, and d. One of them is a zero, and the other three are strictly positive.

The next line contains an integer q ($1 \le q \le 1000$). Then follow q lines with questions. Each question consists of four integers: row_A, col_A, row_B, and col_B. These are coordinates of two squares in the maze (from 1 to 2^n). Here, *row*s are numbered from top to bottom, and *col*umns from left to right.

Output

For each question, print an integer: the length of the simple path between the given squares.

Example

standard input	*standard output*
3 0 1 1 1 1 0 1 1 0 3 1 2 4 4 5 4 5 5 4 8 1 5 8 1 8 5 5 4 5	0 6 22 15

Problem F. Interactive Primality

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

This is an interactive problem.

The jury has chosen $T \leqslant 10$ integers from 1 to 10^{18} inclusive. You need to guess all these numbers one by one.

Let us say the jury has chosen the number x. You can repeatedly call out an integer y from 1 to 10^{18} inclusive, and in response, you will be informed whether the number $x + y$ is prime or composite. When you guess the number, state it, and then start guessing the next one.

In all tests except for the example, all T numbers are chosen independently and uniformly at random from the interval $[1; 10^{18}]$. You need to guess all of them using a total of no more than 8750 queries.

Interaction Protocol

First, read a line with the integer T: the number of integers chosen by the jury ($1 \le T \le 10$). The jury chose T secret integers $x_1, \ldots, x_T$, and you need to guess them all in turn ($1 \le x_i \le 10^{18}$).

Assume you are trying to guess the number x_i. To make a query, print a line formatted as "`?` y" ($1 \le y \le 10^{18}$). Then read the next line:

- if $x_i + y$ is prime, you will read the word "`Prime`";
- if $x_i + y$ is composite, you will read the word "`Composite`";
- if you made an incorrect query (the 8751st during the entire program run; or one that does not fit the specified format; or one in which y is not a positive integer within the required limits), you will read the word "`Busted`".

When you think you know that $x_i = z$, print a line formatted as "`!` z" ($1 \le z \le 10^{18}$). This line is not counted towards the query limit. Then read the next line:

- if indeed $x_i = z$, you will read the word "`Correct`". If $i = T$, the program should terminate; otherwise, proceed to guess the number x_{i+1}.
- if $x_i \neq z$ or if you violated the output format, you will read the word "`Busted`".

After reading "`Busted`", your program must immediately terminate to get `Wrong Answer` verdict. Otherwise, the verdict will be unpredictable.

Example

standard input	*standard output*
1 Composite Prime Prime Prime Correct	? 6 ? 5 ? 3 ? 1 ! 2

Note

There are exactly 30 tests. In the i-th test, $T = \min\{i, 10\}$. In each test, the T chosen integers are fixed in advance: they are the same for each run of each solution and don't change during the interaction.

Problem G. Jump the Frog

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Have you played "Tap the Frog"? It is a large collection of mini-games where you help a frog with its daily tasks. Today, we will discuss the second mini-game called "Jump the Frog."

The game can be visualized as a long grid strip on a lake, where each cell either contains a lily pad or nothing but the water surface. In the leftmost cell of the strip, there is a lily pad, and a frog is sitting on it. The game interface has k buttons: "jump right by one cell," "jump right by two cells," ..., "jump right by k cells." For simplicity, unlike the original game, we will assume that the frog can only jump on lily pads; also, if there are $r < k$ cells remaining to the right of the frog, you cannot press the button "jump right by i cells" where $i > r$.

You want to create a level for this game. You already have several templates, each of which can be represented as a string of characters "~" and "O": the character "~" with ASCII code 126 represents a cell without a lily pad, and the Latin letter "O" with ASCII code 79 represents a cell with a lily pad. Next, you create several new templates: in one operation, you take two templates s_i and s_j (either two different ones or the same one), concatenate them, and add the resulting string $s_i s_j$ to the collection of templates.

For each template, find out how many ways there are to complete the level, that is, to press some buttons in some order to get from the leftmost cell to the rightmost cell without ever visiting cells without a lily pad. Two ways are considered different if they either have a different number of button presses, or they have the same number of presses but for some i, the i-th pressed button in the first way was not the same as in the second. Since these counts can be quite large, find them modulo 998 244 353. Assume that if there is no lily pad in the left or right cell (or in both), the number of ways is zero, even if the strip has a length of one.

Input

The first line contains three integers, n, k, and a: the number of existing templates, the maximum jump length of the frog, and the number of templates you will add ($1 \le n, a \le 10^5$; $1 \le k \le 20$).

In the next n lines, there are strings $s_1, s_2, \ldots, s_n$: the descriptions of the existing templates ($|s_1| + \ldots + |s_n| \le 10^5$). All strings s_i are non-empty and consist of characters "~" and "O".

The next a lines describe how to construct the templates $n+1, \ldots, n+a$. In the i-th line, there are two integers, ℓ_i and r_i, which mean that the $n+i$-th template is obtained by concatenating the ℓ_i-th and r_i-th templates. Formally, $s_{n+i} = s_{\ell_i} s_{r_i}$ ($1 \le \ell_i, r_i < n + i$).

Output

Output $n + a$ integers $A'_1, A'_2, \ldots, A'_{n+a}$. The number A'_i must be within the range from -2^{63} to $2^{63} - 1$ and must be congruent modulo 998 244 353 to the number of ways A_i to get from the left to the right cell of the level, for which the template s_i is chosen (that is, $A_i - A'_i$ must be divisible by 998 244 353).

Example

standard input	*standard output*
4 3 6 0 ~ 000~~ ~000 4 1 1 4 3 1 3 2 8 1 7 7	1 0 0 0 0 3 2 0 0 8

Problem H. Slot Machine

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	1024 mebibytes

Little Misha is sitting in front of a slot machine and wants to win the jackpot. The machine has k slots forming a string of length k. Each slot can show any decimal digit or a question mark. Initially, each slot shows a question mark.

The machine also keeps a secret string consisting of k decimal digits. It is possible that not any decimal string of length k can be stored by it: Misha found a manual for this slot machine, and in chapter 007, all possible secret strings are listed. To win the jackpot, the player needs to make the k slots exactly match the secret string and press a big red button afterwards. However, when the button is pressed, if the displayed string is different from the secret one, the machine converts both the secret string and the string formed by the k slots into two integers (possibly with leading zeros) and tells the player which one of them is greater than the other. Note that if at least one of the slots shows a question mark when the big red button is pressed, then the machine says nothing instead.

Misha can press the button any number of times. Before the first press and between any two presses, he can also choose any number of slots and change the symbols in those slots (question marks or digits) into any other symbols. There is one caveat: for each replacement of a symbol, Misha has to pay one ruble. He doesn't want to anger his mom, so he tries to spend as little as possible. How much is guaranteed to suffice?

Input

The first input line contains an integer T, the number of test cases ($1 \le T \le 10^4$). Then descriptions of T test cases follow.

The first line of a test case description contains k, the number of slots ($1 \le k \le 5$). The second line contains a binary sequence s of length 10^k. For each $i \in \{0, 1, \ldots, 10^k - 1\}$, the value $s_i = 0$ means that number i could not be the secret string, while $s_i = 1$ means that number i is allowed by the manual and could be stored by the machine. There is at least one i such that $s_i = 1$.

The total length of all strings does not exceed 10^5.

Output

Print one integer: the smallest amount of money (in rubles) that is enough for Misha's victory.

Example

standard input	*standard output*
2 1 1110001010 1 1111111111	3 4

Problem I. Product

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	1024 mebibytes

We will be short here.

For a given sequence of integers $(a_0, a_1, \ldots a_{mk-1})$ of length mk, define its weight as the product $\prod_{i=0}^{m-1} a_{ik}$. Calculate the sum of all weights of sequences $(a_0, a_1, \ldots, a_{mk-1})$ such that $1 \le a_0 \le a_1 \le \ldots \le a_{mk-1} \le n_0$ modulo 998 244 353 for all n_0 from 1 to n, inclusive.

Input

The only input line contains three integers: n, m, and k ($1 \le n, k \le 250\,000$; $1 \le m \le 10^{18}$).

Output

Output n lines: answers for $n_0 = 1, 2, \ldots, n$ modulo 998 244 353.

Example

standard input	*standard output*
2 2 2	1 10

Problem J. Generating Random Trees

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Suppose that you need to generate a uniformly distributed random tree from the set of all labeled trees on n vertices. Consider the following algorithm: while the graph is still not connected, pick two uniformly random different vertices from the graph and, if they are not in the same connected component, add an edge between them.

Performing the above process is a rather simple task, requiring only a simple DSU (disjoint set union) data structure. Unfortunately, it turns out that this process does not generate uniformly random labeled trees! Your task is to discern the trees generated by the above procedure from the uniformly random labeled trees.

More precisely, you are given $2k$ labeled trees with n vertices. Exactly k of said trees were uniformly chosen from the set of all labeled trees on n vertices, and exactly k of them were generated by the above procedure. The order in which $2k$ trees are given is also chosen uniformly at random. All random choices are independent from each other.

For each of the $2k$ trees, you need to say whether it was chosen uniformly at random or generated by the above procedure. **However, you are allowed to make some mistakes.** On each test, your solution will be accepted if and only if it has an accuracy of 80% or more. While it is guaranteed that exactly k trees in the input were generated in one way and exactly k trees were generated the other way, there **are no restrictions** on the number of trees that you can identify as truly uniform.

There are exactly 6 tests in this problem: one sample and five main tests. In all main tests, $n = 10^4$ and $k = 50$. In the sample, $n = 4$ and $k = 2$. The sample is here to clarify the input and output formats. **Any answer that satisfies the output format will be accepted for the sample**. It is guaranteed that the sample is the first test in the testing system.

Input

The first line contains two integers n and k ($n = 4$ and $k = 2$ in the sample; $n = 10^4$ and $k = 50$ in the five main tests).

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n$; $u \ne v$): the vertices connected by an edge in the first given tree.

The remaining $(2k - 1)(n - 1)$ lines describe the remaining $2k - 1$ trees in the same format.

Output

Output $2k$ lines: the i-th line should be "`DSU`" or "`Uniform`" depending on whether your solution thinks that the i-th tree was generated by the procedure given in the statement or was chosen uniformly at random from the set of all labeled trees on n vertices. For the sample, any answer that satisfies the output format will be accepted; for the main tests, you are allowed to make at most 20 mistakes.

Example

standard input	*standard output*
4 2 3 2 1 4 1 3 2 1 3 2 4 1 4 1 3 2 2 4 1 4 3 1 2 1	DSU Uniform DSU Uniform

Problem K. Vortex

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

This is a run-twice problem.

Honestly, the author turns 27 the day after writing the legend and does not expect to get a randomly shuffled tree as a present from anyone, so you will have to live without yet another story about weird birthday presents.

Instead, here is what happens in the problem. On each run, you will be given a randomly shuffled tree. As an unlabeled tree, it is the same tree on both runs. It is guaranteed to be randomly shuffled, and it may be shuffled differently on both runs (the trees may be different as labeled trees). Your task is to permute the vertices of two trees in such a way that the trees become equal as labeled trees.

Input

The first line contains a single integer n: the number of vertices in the tree ($2 \le n \le 2 \cdot 10^5$).

Each of the next $n - 1$ lines contains two integers, v and u ($1 \le v, u \le n$): the endpoints of an edge.

Output

Output a permutation of vertices on a single line separated by spaces.

Your answer will be considered correct if and only if for all i and j there is an edge between the i-th and the j-th vertices in your output either in both runs or in none.

Formally, let your outputs be $p_1, p_2, \ldots, p_n$ for the first run and $q_1, q_2, \ldots, q_n$ for the second run. For all i and j, the edge p_i–p_j should exist in the first input if and only if the edge q_i–q_j exists in the second input.

Examples

standard input	*standard output*
7 2 1 7 1 3 7 4 7 6 5 5 2	1 2 3 4 5 6 7
7 1 5 6 7 3 6 2 3 2 1 1 4	2 3 4 5 6 7 1

Problem L. Random Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	14 seconds
Memory limit:	1024 mebibytes

You have a number x, initially equal to 0, as well as a **prime** number p. Next, for $i = 1, 2, \ldots, m$, you increase x by a_i with probability q_i. What is the probability that x will be divisible by p at the end of the process?

The probability can be represented as a rational number. Output it modulo 998 244 353.

Input

The first input line contains two integers, p and m ($2 < p < 30\,000$; $1 \le m \le 10^6$; p is prime).

Each of the next m lines contains two integers, a_i and r_i ($0 \le a_i < p$; $0 \le r_i \le 10^8$). The actual probability q_i is equal to $r_i/10^8$.

Output

Output a single integer: the answer modulo 998 244 353.

Formally, if the answer is a rational number x/y, print the integer $x \cdot y^{-1} \bmod 998\,244\,353$. Here, y^{-1} is an integer such that $y \cdot y^{-1} \bmod 998\,244\,353 = 1$.

Example

standard input	*standard output*
2 3 0 100000000 1 100000000 1 50000000	499122177