Problem A. Canonical Palindromes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

A palindrome is an array of integers that is equal to the reverse of the same sequence. Emmett finds palindromes too confusing, so he decided to make palindromes more "canonical."A palindrome is a canonical palindrome if the first half of the palindrome is non-decreasing and the second half of the palindrome is non-increasing.

Emmett now wants to determine how he might transform an arbitrary array a of length k into a canonical palindrome. To do this, he has decided to perform the following operation:

1. Choose an integer i such that $1 \leq i \leq k$ and let $j = k - i + 1$. Set $x = a_i$ and $y = a_j$, then remove a_i and a_j from a.
2. Choose whether or not to swap the values of x and y.
3. Partiton a into three subarrays: p, m, and s.
   - Let p be some prefix of the array and s be some suffix of the array, such that the lengths of p and s are equal.
   - Let m be the remaining subarray after removing p and s.
   - Any of these subarrays are allowed to be empty, as long as $p + m + s$ is equal to a.
4. Set a equal to $p + x + m + y + s$ where '+' represents the concatenation of two arrays.

All steps of the operation must be performed, and they must be performed in the order stated above. To avoid weird interactions with the middle element, Emmett has decided that the length of the array must always be even.

Emmett wants you to find the minimum number of operations to turn the array into a canonical palindrome, or state if it is impossible to ever do so; however, the data he wrote for this problem got corrupted! Several arrays got merged into a single input file, and instead of fixing the data, he has decided that you must solve this problem for queries on a given array of length n of the following form:

- Given two integers L and R ($1 \leq L < R \leq n$, $R - L + 1$ is even), determine the minimum number of operations to turn a_L, a_{L+1}, ... , a_{R-1}, a_R into a canonical palindrome.

Input

The first line of input consists of two integers n and q ($2 \leq n, q \leq 5 \cdot 10^5$) — the size of the array and the number of queries.

The second line of input consists of n integers a_1, a_2, ... , a_n ($1 \leq a_i \leq 10^9$) — the elements of array a.

The following q lines will consist of two integers corresponding to the query described above.

Output

For each query, output a line with a single integer denoting the minimum number of operations to turn the subarray into a canonical palindrome. If the subarray cannot be turned into a canonical palindrome, output -1.

Example

standard input	standard output
14 4 2 1 1 2 1 3 1 2 1 5 5 1 2 1 1 4 6 9 7 14 9 12	1 -1 1 0

Problem B. Cocktail Sort

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

Andrew has just learned about cocktail sort and has developed the following C++ code:

```
bool comp(int x, int y) {
    return x > y;
}
void sort(vector<int> &arr) {
    int n = arr.size();
    for(int iter = 0; iter < n-1; iter = iter + 1) {
        bool done_sorting = true;
        for(int i = 0; i < n-1; i = i + 1) {
            if(comp(arr[i], arr[i+1])) {
                done_sorting = false;
                swap(arr[i], arr[i+1]);
            }
        }

        for(int i = n-1; i > 0; i = i - 1) {
            if(comp(arr[i-1], arr[i])) {
                done_sorting = false;
                swap(arr[i-1], arr[i]);
            }
        }

        if(done_sorting) break;
    }

}
```

Andrew claims that this sorting algorithm is such a significant optimization to bubble sort, so it must have a time complexity of $O(n)$. To prove the foolishness of this claim, Ben created some arrays that might still take a large number of operations to sort the array. However, these cases are too big for Andrew to analyze, so he has asked you to write a program to count the total number of operations needed for this code to sort a given array. Andrew has defined an operation as a call to the "`comp`" or the "`swap`" functions.

Input

The first line of input consists of one integer n ($1 \le n \le 10^6$), the size of the array.

The second line of input consists of n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array.

Output

Output a single integer — the number of operations needed to sort the array.

Examples

standard input	standard output
9 3 1 4 1 5 9 2 6 5	58
7 2 7 1 8 2 8 1	45
4 1 6 1 8	13

Problem C. Corrupted Odyssey

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Thomas wanted to propose a string problem inspired by a famous story, but was uncertain of what story to use. Tyler, being a mythology nerd, proposed Homer's epic *The Odyssey*. Thomas's response to this was, "The Odyssey! What a lame piece of fiction. Percy Jackson did it better! In fact, I would go as far as to say that random gibberish would make a better story!" To try proving Thomas wrong, Tyler attempted to pull up a passage from *The Odyssey* on his computer, but to his dismay, his file was corrupted!

Tyler was able to prove that the copy of *The Odyssey* was corrupted in the following way: All non-alphabetical characters were removed, all capital letters were replaced with their lowercase counterparts, and then a substitution cipher was applied to the text. More formally, a permutation of the 26 lowercase English letters was generated; then, every letter was replaced with its corresponding new letter. That is, all 'a's were replaced with the first letter in the permutation, all 'b's were replaced with the second letter of the permutation, and so on. All substitutions are applied simultaneously.

Thomas still claims that it is indistinguishable from gibberish, so to settle their disagreement, Tyler has given you two strings. The following methods were used to generate the two strings:

1. All non-alphabetical characters in an English translation of Homer's *The Odyssey* are removed and all uppercase characters are replaced with the corresponding lowercase letters. Of all substrings of 1000 characters in the text, a substring is chosen uniformly randomly. A substitution cipher is then generated by choosing a permutation of the 26 lowercase letters uniformly randomly, and then it is applied to the chosen substring.
2. A string of 1000 characters was randomly generated. That is, each character of the string was generated by picking one of the 26 lowercase English letters with equal chance.

You are guaranteed that both methods were used to generate one string, and Tyler has asked you to determine which method was used to generate each string.

Input

The first line of input will consist of a single integer n ($n = 35$ or $n = 1000$) — the number of characters in the string. In all cases other than the sample, n will equal 1000.

The second and third lines of input will consist of a single string of n lowercase English letters, each of which were generated using one of the two methods described above.

Output

Output two lines, each denoting which of the two methods was used to format each string. If method 1 was used to generate the string, output "`Corrupted Odyssey`" (without quotes); otherwise, output "`Random Gibberish`" (without quotes). The output is case sensitive.

Example

standard input	standard output
35 ibukjacwauujxwqmfxwwpaoklajjuwaokkm yiewscmawvpsuekbuwbysrfvzpkwankdskr	Corrupted Odyssey Random Gibberish

Note

Before the substitution cipher was applied, the substring taken from *The Odyssey* was "ouldtakealltheirsheepandcattleanddr."

Problem D. Dots and Boxes?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Talia and Terra are playing a variant of dots and boxes. Just as in the original version, you are given an n by m grid of dots, and Talia and Terra will take turns connecting two cardinally adjacent dots with a segment. In this version, once a region is fully enclosed by a set of segments, all cells that aren't possessed yet in that region are possessed by the person who placed the last segment enclosing the region. Once a cell is possessed, neither player may place any more segments connecting two corners of that cell. Once a player cannot place a segment, they lose the game.

To make the game more interesting, some segments will already be added to the board before the game even begins. It is guaranteed that no set of segments initially on the game board will enclose a region. If Talia makes the first move, and both Talia and Terra play optimally, who will win the game?

Input

The first line of input will consist of two integers n and m ($2 \leq n, m \leq 500$) — the number of rows and columns of the grid of dots.

The following $2n - 1$ lines will each consist of $2m - 1$ characters denoting the initial state of the board. For each i ($1 \leq i \leq n$) and j ($1 \leq j \leq m$):

- The character on the $2i - 1$ row and $2j - 1$ column will be a '*', representing a dot.
- If $j < m$, the character on the $2i - 1$ row and $2j$ column will be a '-' if there is a segment connecting dots (i, j) and (i, j+1), and '.' otherwise.
- If $i < n$, the character on the $2i$ row and $2j - 1$ column will be a '|' if there is a segment connecting dots (i, j) and (i+1, j), and '.' otherwise.
- If $i < n$ and $j < m$, the character on the $2i$ row and $2j$ will be a '.', corresponding to the empty cell.

It is guaranteed that no set of segments initially on the game board will enclose a region.

Output

Output "Talia" (without quotes) if Talia wins the game; otherwise, output "Terra" (without quotes). The output is case sensitive.

Examples

standard input	standard output
2 3 *.*.* ..... *.*.*	Talia
5 5 *-*-*-*-* |.|.....| *.*-*.*.* |...|.|.| *.*-*-*.* |...|...| *.*-*.*.* |.|...|.| *.*-*-*.*	Terra

Problem E. Just Tree Diameter

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

You are given a tree with n nodes where the edges are weighted. The diameter of the tree is defined as the maximum weighted distance between any two nodes of the tree, where the weighted distance between two nodes is the sum of the edge weights on the simple path between them. You must write a program that finds the diameter of the given tree. Easy enough, right?

But wait, there's more! You will be given a set of updates to the tree. In a single update, you will add x to the weights of all edges in the tree. Updates are persistent, meaning that once an update occurs, the next update will operate on the tree after all edge weights have been increased by x. After each update, you want to find the new diameter of the tree.

Input

The first line of input will consist of two integers n ($2 \leq n \leq 10^5$) and m ($1 \leq m \leq 10^5$) — the number of nodes in the tree and the number of updates.

The following $n-1$ lines will consist of three integers u, v ($1 \leq u, v \leq n$), and w ($1 \leq w \leq 10^{12}$), denoting that there is an edge between node u and node v with initial weight w.

The following m lines will consist of a single integer x ($1 \leq x \leq 10^{12}$) denoting that x gets added to every tree edge. It is guaranteed that the sum of x across all updates will be at most 10^{12}.

Output

Before the first update and after each update, output a line with a single integer corresponding to the current tree diameter.

Example

standard input	standard output
7 2 3 7 5 1 3 3 3 4 1 1 6 7 1 5 2 5 2 4 1 2	15 18 26

Problem F. Knapsack One Million

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

The Orlando City Plaza Casino has released a new game for their patrons to enjoy: Knapsack One Million! In this game, a million types of random chips will be dispensed. Each type of chip will have a random size and a random value, and a million of each type of chip will be dispensed. You are then given a Knapsack One Million, which is a bag that can hold a set of chips whose sum of sizes is at most one million; that is, the capacity of the knapsack is one million.

Before testing your luck in this game, you want to analyze the best-case performance you could have in a game; however, you worry the casino might attempt to cheat by giving you a bag with a more limited capacity than advertised. Thus, for a given set of chips, you want to compute the maximum value you can achieve for all knapsack capacities between 1 and 10^6, inclusive.

Input

The first line of input will consist of a single integer n ($n = 10$ or $n = 10^6$) — the number of chips. In all cases other than the sample, n will equal 10^6.

The following n lines will consist of two integers s_i and v_i — the size and value of the ith type of chip.

Each case (except for the sample) will be selected from a set of 10^6 generated cases. Each case was generated such that s_i and v_i are uniformly randomly generated from the range 1 to n, inclusive. It is guaranteed that each value is generated independently. Please read the note for more details.

Output

Output a single line consisting of n integers where for each i from 1 to n, inclusive, the ith integer corresponds to the maximum value of chips you can choose such that the sum of weights of the chips is at most i.

Example

standard input	standard output
10 7 2 1 3 8 5 4 6 9 10 5 4 7 9 6 1 2 10 8 3	3 10 13 20 23 30 33 40 43 50

Note

Each testcase, except for the sample, was generated using C++'s built-in "`std::mt19937`" (Mersenne Twister) in combination with "`std::uniform_int_distribution`". The random number generator was seeded with an integer between 1 and 10^6 (inclusive), though the seed was not necessarily chosen at random. Items were generated independently at random: the first $2 \cdot 10^6$ values produced were used as the sizes and values of each chip type. Specifically, for the i-th chip, the size is the $(2i - 1)$-th value and the value is the $(2i)$-th value generated.

Problem G. Puzzle Boy

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Ben's favorite hobby is puzzles such as Heyawake or Nurikabe. Sadly, he has solved every puzzle that exists, so Nic has decided to create a new one for him. In this puzzle, Nic will lay out some cards, each with a positive or negative integer written on its top face (note that zero will not appear on any cards). Ben then must choose some (possible none) of the cards such that no non-empty subset of chosen cards has a product whose absolute value is a perfect square. Ben must also optimize the set of cards he chooses, first maximizing the number of cards he chooses, then maximizing the product of the integers on the cards. The product of the empty set is defined to be 1.

As Nic commonly claims, "surely puzzle boy can puzzle it out," and indeed he did. Ben solves these puzzles with such ease that Nic doesn't even have time to verify Ben's answer! Thus, Nic has asked you to write a program that computes the maximum product that Ben can achieve. To make the calculations easier, Nic has guaranteed that each integer written on a card can be expressed as the product of integers between -1000 and 1000, inclusive.

Input

The first line of input will consist of a single integer n ($1 \leq n \leq 10^5$) — the number of cards Nic will reveal to Ben.

The second line of input will consist of n integers a_i ($-10^9 \leq a_i \leq 10^9, a_i \neq 0$) — the integer on the ith card that Nic reveals. It is guaranteed that the integer written on each card can be expressed as the product of integers between -1000 and 1000, inclusive.

Output

Output a single line with a single integer denoting the maximum product that Ben can achieve. Since this answer is large, output a value in the range $[0, 998244353)$ which is congruent to the maximum product under modulo 998244353.

Examples

standard input	standard output
5 2 4 3 18 8	54
5 2 4 -3 18 -8	24
5 2 -3 5 -7 -11	998242043

Problem H. Shell Shock

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Luisa is playing in a Mario Kart championship, and one of her competitors released a flurry of green shells on her. Green shells are a power up that, when used, will launch a turtle shell in some direction indefinitely and will bounce off walls until it hits some other competitor. Luisa is unsure of the exact mechanics that define the path of a green shell; all she knows is that she wants to avoid them, and the more paths to the finish line there are, the less likely she is to encounter one.

Being a good friend of hers, you have hacked into the map and can now redesign it to her benefit. You know there are n junctions in the current map Luisa is racing in, and you will redesign the map by first removing all pre-existing roads, then by adding in roads connecting a pair of junctions. These roads can be traversed bidirectionally. You also know that Luisa will always take the least number of roads needed to reach the junction containing the finish line, regardless of the lengths of the roads. To minimize the chances that Luisa encounters a green shell, you want to redesign the map to maximize the number of routes Luisa can take. Before considering the new map design, you first want to answer the following question: what is the maximum number of shortest routes you can achieve with any new map design?

More formally, let the map Luisa is on be represented by a graph. If she is currently in node 1 and her target is in node n, determine the maximum number of shortest paths that you can create by placing edges between pairs of nodes. A shortest path from 1 to n is defined as a path with the minimal number of edges that starts in node 1 and ends in node n.

Input

The first and only line of input will consist of a single integer n ($2 \leq n \leq 10^9$) — the number of junctions in the map.

Output

Output a single integer: the maximum number of shortest paths you can create. Since this value can be large, output the answer under modulo 998244353.

Examples

standard input	standard output
4	2
6	4

Problem I. Stepping Stones

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

As you are traveling across the vast wilderness, you come across a puzzling challenge. There is a sign that states, "Danger: River Rapids Ahead."However, you see no river in sight. Trying to prepare for this obstacle, you collect some set of stones to use to cross the river. However, since you don't know what the river ahead looks like, you want to plan out several different scenarios of "If I take this subset of stones, can I cross the river?

To know if you can successfully cross the river, you model the problem in the 2D Cartesian Plane. Each stone will be a convex polygon, and each river will be modeled as two parallel lines representing the two banks of the river. You have laid the stones out in a line and have indexed them from left to right, starting with index 1. You have decided that for each scenario you will use each stone from L to R, inclusive.

Since stones are heavy, you can only move them such that they are translated to another position in the coordinate plane. You cannot rotate or flip the stones. You can successfully cross the river if there are some set of translations you can apply to each stone in the range such that if you start on one bank of the river, you can walk to the other bank while staying within or on the boundary of at least one stone. Note that stones are allowed to overlap with other stones and the boundaries of the river.

Input

The first line of input will consist of two integers n and r ($1 \leq n, r \leq 10^5$) — the number of stones and the number of river scenarios.

Following will be the description of each stone. Each description will start with a line consisting of a single integer k_i ($3 \leq k_i \leq 500$) — the number of points that make up the stone. The following k_i lines will each consist of two integers x_j and y_j ($-10^4 \leq x_j, y_j \leq 10^4$) - - - the jth point of the stone. It is guaranteed that the points form a convex polygon and the points will be given in counter clockwise order. The sum of all k_i will be less than $5 \cdot 10^5$. For each polygon, it is guaranteed that no three adjacent points are collinear.

The last r lines will consist of 8 integers x_1, y_1, x_2, y_2, ($-2 \cdot 10^9 \leq x_1, y_1, x_2, y_2 \leq 2 \cdot 10^9$), dx, dy, ($-10^9 \leq dx, dy \leq 10^9, |\langle dx, dy \rangle| > 0$), L, and R ($1 \leq L \leq R \leq n$). The first boundary of the river is the line with the direction vector $\langle dx, dy \rangle$ and that goes through point (x_1, y_1). The second boundary of the river is the line with the same direction vector, but goes through point (x_2, y_2). It is guaranteed that these two lines are not coincident. Lastly, the range of stones that can be used is denoted by L and R.

Output

For each river scenario, output `"YES"`if the river can be crossed with the range of stones or output `"NO"`if the river cannot be crossed with the range of stones. The output is case sensitive.

Example

standard input	standard output
5 5 4 0 -1 1 0 0 1 -1 0 3 -3 -1 3 -1 0 2 4 0 0 -3 0 -3 -3 0 -3 8 1 1 3 1 4 2 4 4 3 5 1 5 0 4 0 2 3 0 0 4 0 4 4 0 0 1 15 1 0 2 5 0 0 1 15 1 0 1 5 12 12 1 3 2 -2 2 4 12 12 1 3 -2 2 1 5 5 5 7 2 -1 1 5 5	NO YES NO YES YES

Problem J. Techno-Trains

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

You have been put in charge of designing a train network to connect several cities, each containing a single train station, together. To do this, you can add a railroad connecting two different cities, as long as no railroad exists between those two cities already; however, to ensure traveling between cities is as simple as possible, any pair of cities should have at most two different simple paths between them. Also, to guarantee any two cities can reach each other in the train network, each pair of cities must have at least one path between them.

Each railroad can also be upgraded to a techno-road. A techno-road contains additional technology that supports a techno-train: a train that has additional technological features such as more luxurious passenger cars or faster arrival times. There are k different types of techno-trains, and each techno-road can only support one type of techno-train. The train station in each city also has limitations on how it may support techno-trains, mainly due to budget. Each city's train station may only support one type of techno-train entering it and at most two techno-roads. In addition, no subset of techno-roads can make a cycle. There is no limit on how many of each type of techno-road you may use.

Given that there are n cities, you want to determine the number of different train networks you may construct. Two train networks differ if for some pair of cities u and v, they have a different type of connection, where a connection is either:

1. No railroad connects the two cities.
2. A standard railroad connects the two cities.
3. A techno-road supporting type i connects the cities.

Note that if in the first train network, the techno-road spanning u and v supports type i, but in the second train network it supports type j, such that $i \neq j$, the two train networks are different.

Input

The input will consist of a single line consisting of two integers n and k ($1 \leq n, k \leq 50000$) — the number of cities and the number of types of techno-trains.

Output

Output a single integer — the number of train networks you can construct with n cities. Since this number can be large, output the answer modulo 998244353.

Examples

standard input	standard output
2 60	61
3 1	19
3 2	34
4 2	679
4 1000	30264031

Problem K. Urban Horizons

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Theo recently downloaded a new game: Urban Horizons! In this game, he is tasked with designing a city's building layout, public transport, infrastructure, and much more. He's designed several hubs for his city, but he wants to add a highway system that connects all of these hubs. A highway connects two hubs of the city and can be traversed bidirectionally. Note that two hubs can have multiple highways between them, and a highway can connect a hub back to itself. Typically, people try to design their city's infrastructure to be as efficient as possible, but Theo is quite bad at the game. He decides that the best approach is to randomly generate highways and add them to the city. Before he enacts his plan, he first writes a simulation to test how feasible it is.

```
void build(Graph g) {
    int n = g.size();
    while(g.not_connected()) {
        int u = rand(1, n);
        int v = rand(1, n);
        g.add_edge(u, v);
    }
}
```

The code above denotes the strategy Theo uses to build the graph of highways for his city. The "`not_connected`" function returns true if there exists some pair of nodes that are not yet connected through some subset of edges, and the "`add_edge`" function adds a new edge between nodes u and v. The function "`rand(x, y)`" returns a uniformly random number between x and y, inclusive.

Theo has several different initial highway configurations he has already created on the same layout of hubs. For each initial configuration, Theo has asked you to compute the expected number of highways that need to be added to the city such that all hubs are connected directly or indirectly to each other.

Input

The first line of input consists of two integers c ($1 \leq c \leq 100$) and n ($1 \leq n \leq 22$) — the number of initial highway configurations to process and the number of hubs in the city.

The first line of each initial configuration consists of a single integer m ($0 \leq m \leq 250$) — the number of highways that already exist in the city.

The following m lines consist of two integers u_i and v_i ($1 \leq u_i, v_i \leq n$) — denoting the ith highway which connects hub u_i and v_i.

Output

Output a line consisting of the expected number of highways that need to be added such that every hub is connected. It can be shown that the answer can be represented as $\frac{p}{q}$ where p and q are integers, so print the answer in the form of $p \cdot q^{-1} \mod 998244353$.

Example

standard input	standard output
2 6 6 1 2 2 3 3 1 4 5 5 6 6 5 1 3 3	2 891588784

Note

In the first configuration, there are two disjoint cycles of highways, each with 3 hubs. Of the 36 potential choices for u and v, 18 of them connect the two cycles, making the graph connected. Any other choice either corresponds to a self-loop or an edge that already exists. Thus, the probability we first connect the graph after $k-1$ edges have been added is equal to $\frac{1}{2^k}$. Therefore, the expected number of edges we must add to connect the graph is equal to $\sum_{k=1}^{\infty} \frac{k}{2^k} = 2$.

Problem L. Yet Another XOR Problem

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

Polarized lenses are a great feat in the field of optics and have many fascinating properties. As you are messing around with a set of polarized lenses, you notice that the region of light they block is equivalent to the XOR of the regions that each lens covers. More formally, let's consider the 2D Cartesian Plane. Each lens will take the form of a circle in the plane. A point is considered to be blocked if there is an odd number of circles that contain the point within its border or on its border.

Now you want to explore the results of several updates. In each update, a new polarized lens of radius r is added into the Cartesian Plane at (x, y). After each update, you want to determine the total area of the set of blocked points in the entire plane.

Input

The first line of input will consist of a single integer n ($1 \leq n \leq 1000$) — the number of polarized lenses that are being added to the plane.

The following n lines will consist of three integers x, y ($-10000 \leq x, y \leq 10000$), and r ($1 \leq r \leq 10000$) — the x and y coordinate of the center of the lens and the radius of the lens.

Output

After each update, output a single real number – the area of the set of blocked points. Your answer will be accepted if it has an absolute or relative error of 10^{-4} with the judge's solution.

Example

standard input	standard output
3 -1 1 3 1 -1 3 0 0 1	28.2743338823 32.6384059655 35.7799986191