Problem A. Happy Birthday, Jedi Knight!

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

The Empire created a new version of its sinister vehicle — the Death Star version S3.14T (codename "Poetic Pig"). If nobody stops it — the galaxy is doomed! Fortunately, the retired Jedi Knight once again returns to the front line to prevent the tragedy.

First of all, Jedi Knight decided to explore the Death Star to find its vulnerable points. He wants to make as many explorations as possible, but he knows that if he chooses the same route of exploration more than once, then he will be caught. Help Jedi to make as many explorations as possible without being caught!

Let us describe the problem more formally. The Death Star contains 2^n rooms, each room is labeled by an unique binary string of length n. Two rooms are connected by a corridor iff their labels differ in exactly one position. Let us say that a sequence of rooms $(v_1, v_2, \ldots, v_k)$ is *a closed walk* if for every i rooms v_i and v_{i+1} are connected with a corridor and, moreover, v_1 and v_k are also connected. Each exploration route of Jedi Knight should be a closed walk. So, you need to find the number of different closed walks. Closed walks $(u_1, u_2, \ldots, u_k)$ and $(v_1, v_2, \ldots, v_k)$ are considered different if there exists such i $(1 \leqslant i \leqslant k)$ that $u_i \neq v_i$. For example, $(1, 2, 3, 4)$ and $(2, 3, 4, 1)$ are different closed walks.

Input

The first line contains two integers — n and k $(2 \leqslant n \leqslant 50,\ 1 \leqslant k \leqslant 10^9)$.

Output

Output one integer — answer modulo $10^9 + 7$.

Examples

stdin	stdout
2 69883628	845391189

Problem B. Positive Eigenvalues

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Let A be a matrix of size $n \times n$ that consists only of zeroes and ones. Let us call a complex number $\lambda \in \mathbb{C}$ *eigenvalue* of A if there exists a non-zero column-vector $v \in \mathbb{C}^n$ such that $Av = \lambda v$. You are to check if all eigenvalues of A are *real and positive*.

Input

The first line contains one positive integer — n ($1 \leqslant n \leqslant 50$). The next n lines contain A itself. Each of these lines contains n zeroes or ones.

Output

On the first line output "`YES`" if all eigenvalues of A are real and positive and "`NO`" otherwise (here quotes are for clarity only).

Examples

stdin	stdout

3
1 1 1
0 1 1
0 0 1
	YES

Problem C. SAT

Input file:	`stdin`
Output file:	`stdout`
Time limit:	3 seconds
Memory limit:	64 mebibytes

Consider n boolean variables $x_1, \ldots, x_n$ and m constraints that are disjunctions of distinct variables or their negations, for example $x_{i_1} \vee \neg x_{i_2} \vee x_{i_3}$. Let us call two constraints *neighbors* if they share some variable (doesn't matter with negation or without it). Each of these constraints contains exactly t distinct variables and each constraint has no more than $2^t/3$ neighbors. Find a set of variable values that satisfies all the constraints.

Input

First line of the input file contains three integer numbers n, m and t ($1 \leqslant n, m, t \leqslant 1000$). Each of the following m lines contains one constraint that consists of variables (like x40, x57, x218, index is an integer number from 1 to n), and operators OR and NOT. Variables and operators are separated with single space. All lines are non-empty.

Output

If it is impossible to satisfy all the constraints output "`IMPOSSIBLE`" (here quotes are for clarity only). Otherwise output n space separated numbers (0 or 1) — values of the variables.

Examples

stdin	stdout

`3 3 3`
`NOT x2 OR NOT x3 OR NOT x1`
`x2 OR x3 OR NOT x1`
`x2 OR NOT x3 OR NOT x1`
	`1 1 0`

Problem D. Permutations

Input file:	`stdin`
Output file:	`stdout`
Time limit:	10 seconds
Memory limit:	512 mebibytes

Let S_n be the set of all permutations that act on $\{1, 2, \ldots, n\}$. You are given a subset $P \subseteq S_n$, and you are to compute the size of the set $G(P) \subseteq S_n$, where $G(P)$ meets the following requirements:

1. $G(P)$ contains the identity permutation,
2. $P \subseteq G(P)$,
3. $G(P)$ is closed under taking inverses and multiplications (if $a, b \in G(P)$, then $ab \in G(P)$, if $a \in G(P)$, then $a^{-1} \in G(P)$),
4. $G(P)$ is a minimal possible set that meets requirements 1–3.

Input

On the first line there are two positive integers: n ($1 \leqslant n \leqslant 50$) and m ($1 \leqslant m \leqslant 100$) — the size of P. Then on the next m lines there are the descriptions of elements of P. A permutation is described as n integers: the first is the image of 1, the second is the image of 2, and so on.

Output

Output the required answer without leading zeroes. Note that it can be large, so consider using arbitrary-precision arithmetic in your solution.

Examples

stdin	stdout

3 2
2 1 3
3 1 2
	6

Problem E. Long Paths

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

A tourist has just arrived to a small town with beautiful name “Nsk”. There are n places of interest in this town, and there are m roads between them. A road can be travelled in both directions. The tourist wants to select his route in the following way: the route must be a node-disjoint path that goes through several places of interest. Moreover, the number of roads in the route must be exactly k. Since the town is small, it is quite well-connected, that is, the inequality $m \geqslant nk$ holds. Help the tourist to find such a route or to determine that it is impossible.

Input

The first line of the input contains three integer numbers — n, m, k ($1 \leqslant n, m, k \leqslant 10^5$, $m \geqslant nk$). The next m lines contain the descriptions of the roads. Each line contains two integers x and y — the indices of the places connected by the road ($1 \leqslant x, y \leqslant n$; $x \neq y$). There is at most one road between any pair of places.

Output

If there is no route which satisfies all the requirements, output exactly one line with the word “`No`” (quotes are for clarity only). Otherwise output a line with the word “`Yes`”. The next $k + 1$ lines must contain the sequence of indices of nodes in your route.

Examples

stdin	stdout

`3 3 1`
`1 2`
`1 3`
`2 3`
	`Yes`
	`1`
	`2`

Problem F. Candies

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Billy has a set of magic chests, each chest contains some magic boxes. For each box two numbers are known: a_i and b_i. Each box works as follows: after you open the box, you can extract x_i candies and y_i proteins from the box, in such a way that $0 \leqslant x_i \leqslant a_i$, $0 \leqslant y_i \leqslant b_i$ and $x_i : y_i = a_i : b_i$. Note that x_i and y_i can be fractional. Ratio $\lambda_i = x_i : a_i$ is called *openness* of the box. If some box is not opened, its openness is equal to zero.

Billy wants to open some boxes in such a way that the sum of openness of boxes in **each chest** is less than or equal to 1. The total amount of food (candies and proteins) extracted will be Billy's monthly food. Billy wants to maximize the number of candies, but there are some problems. The doctor says that Billy should eat at least P proteins per month, so the amount of candies should be maximized subject to this condition. Help Billy to choose an openness of each box!

Input

The first line of the input contains two integer numbers N and P, the number of chests and the minimum amount of proteins required ($0 \leqslant N \leqslant 10^5$, $0 \leqslant P \leqslant 10^6$). Each of the following N blocks describe chests in the following format: first line of the block contains the number of boxes in the chest — M ($0 \leqslant M \leqslant 10^5$); the next M lines contain real numbers A_i, B_i, $0 \leqslant A_i, B_i \leqslant 10^5$. The sum of all M's in the input will be less than or equal to 10^5.

Output

Output must contain the maximum number of candies (that is possibly fractional). Absolute or relative error must be less than 10^{-9}. In case that it is impossible to get enough proteins, just maximize the amount of candies.

Examples

stdin	stdout

```
2 7
3
1 1
1 20
5 6
3
7 6
5 20
11 12
```
	16.0000000000

Problem G. Projection

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Consider the following procedure that generates a random convex polygon. For a given set of points $P_1, \ldots, P_n$ in 3-dimensional space we uniformly choose a direction and find the projections $Q_1, \ldots, Q_n$ of these points along it. The resulting 2-dimensional polygon is the convex hull of $Q_1, \ldots, Q_n$. In order to generate small enough polygons it's useful to understand what is the average area of the resulting polygon. Find it!

Input

The first line of the input contains one integer number n: $1 \leqslant n \leqslant 50$. Each of the next n lines contains three integer numbers x_i, y_i, z_i. All these numbers don't exceed 50 by absolute value.

Output

Output one real number with six exact digits after the decimal point — the average area of the resulting polygon.

Examples

stdin	stdout

4
-2 -4 -5
3 0 -4
0 -4 -1
-2 -1 -4
	9.7067484771

Problem H. Subsequences

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Today Mika has learned an algorithm of finding the largest increasing subsequence of a given sequence in time $O(n \log n)$. However, the algorithm itself seemed to him quite trivial and straightforward. So in order to make his life more interesting he decided to solve another problem: given a sequence of n elements, determine if there are k disjoint increasing subsequences with the total length l (two subsequences $a_{c_1}, \ldots, a_{c_p}$ and $a_{c'_1}, \ldots, a_{c'_q}$ are called *disjoint* iff sets $\{c_1, c_2, \ldots, c_p\}$ and $\{c'_1, c'_2, \ldots, c'_q\}$ have empty intersection). Note that the subsequences can be empty. Of course, he has chosen the most interesting *on-line* version of the problem — elements of the sequence appear one by one, and on each step he is to find the greatest number that can be represented as a sum of lengths of k disjoint increasing subsequences.

Input

The first line of the input contains two integers — n and k ($1 \leqslant n, k \leqslant 10^5$, $n \cdot k \leqslant 10^7$). The second line contains n space-separated integers — the sequence itself. All integers are in range $[0, 10^9]$.

Output

Output must contain exactly n lines. The i'th line must contain the greatest number that can be represented as a sum of lengths of k disjoint strictly increasing subsequences in a prefix with length i of the original sequence, or `-1` if such a number does not exist.

Examples

stdin	stdout
4 1 16 11 1 5	1 1 1 2

Problem I. Yet another pattern matching

Input file:	`stdin`
Output file:	`stdout`
Time limit:	3 seconds
Memory limit:	64 mebibytes

You are given two strings S and T composed from small Latin letters. Find all occurrences of T in S as a substring. Oh, wait …Isn't this problem way too trivial? Let's make it harder!

Suppose that S and T can contain not only small Latin letters, but also wildcards. A wildcard can be matched against any symbol (including wildcard itself). Could you solve it now?

Input

The first line contains S and the second one contains T. Length of each string is not smaller than one and is not greater than 10^5. Both strings consist only of small Latin letters and question marks (which denote wildcards).

Output

On the first line output k — the number of occurrences of T in S as a substring. On the second line output k positive integers — positions where substring of S matches T. This list must be sorted.

Examples

stdin	stdout

`abac`
`a?`
	`2`
	`1 3`

Problem J. Transitive Closure

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Consider the following procedure for producing random acyclic graphs. Let us take n vertices identified with $\{1, 2, \ldots, n\}$. Then consider all pairs (i, j), where $1 \leqslant i < j \leqslant n$ independently and for every such a pair add an arc (i, j) to the resulting graph with probability p. What is the expected number of arcs in the transitive closure of a graph obtained with this procedure (an arc (u, v) belongs to the transitive closure of a graph G iff v is reachable from u in G)?

Input

The first line of input contains two numbers — n and p ($1 \leqslant n \leqslant 1000$, $0 < p < 1$).

Output

Print answer to the problem. Your output must not deviate from the correct answer by more than 10^{-6} (either in terms of absolute or relative error).

Examples

stdin	stdout
`4 0.5`	`3.4843750000`

Problem K. Communications

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 mebibytes

In this problem you need to help the famous Face Palm corporation. Face Palm consists of k departments and currently employs n workers. Each worker belongs to exactly one department. You know for some (unordered) pairs of workers that they are friends. Despite its size, Face Palm has only one office, so each pair of friends communicates extensively during working time. However, it can decrease the performance of workers: for example, if a manager communicates with a cleaning woman, it is not very likely that the subject has something to do with their jobs. On the other hand, the communication between two software engineers can be quite useful. In the simplified model, you can assume that the communication is "bad" iff the communicating workers are from different departments. You need to solve the following problem: select exactly one pair of workers from different departments and swap them between their departments in such a way that the number of bad communications is minimized.

Input

The first line of input contains three integer numbers n, m and k — the number of workers, the number of pairs of friends and the number of departments respectively ($1 \leqslant n, m, k \leqslant 100\,000$). The second line contains n space-separated integers — the indices of departments of the corresponding workers (each index is an integer between 1 and k). The next m lines contain the descriptions of friends — each line contains pair of integers x, y — indices of friends (numbered from 1). No person is a friend of himself, each pair of people can occur in the input at most once.

Output

Output should contain two numbers — the indices of workers. If there are several optimal solutions, print any of them. Note that in principle the optimal swap can increase the number of bad communications.

Examples

stdin	stdout

```
4 4 2
1 1 2 2
1 2
1 3
2 4
1 4
```
	`1 4`