Problem A. Byteland

Input file:	a.in
Output file:	a.out
Time limit:	10 seconds
Memory limit:	256 Mebibytes

There are n towns in Byteland. King Byteman would like to improve the infrastructure in Byteland and has ordered the royal architects to prepare plans for building highways across the country. He has received m proposals, each of which consists of three numbers p, k, w, where p and k are the towns being the endpoints of the highway, and w denotes the cost of building this highway. Each highway is bidirectional.

The king would like to choose the highways in such a way that there exists at least one way of travelling between any pair of towns, possibly visiting some other towns on the way. Byteman would like the highway network to be as cheap to build as possible.

Write a program which:

- reads from the standard input the number of towns, the number of proposed highways and the descriptions of these highways,
- finds out, for every highway, whether there exists a highway network containing this highway and satisfying the Byteman's requirements,
- writes the results to the standard output.

Input

The first line of the input contains two integers: the number of towns, n, and the number of proposed highways, m, separated with a single space and satisfying the conditions: $2 \leqslant n \leqslant 7\,000$, $1 \leqslant m \leqslant 300\,000$. Each of the following m lines contains three space-separated integers p, k, w, describing the proposed highway. p and k denote the towns which are endpoints of the highway, while w is the price of the highway ($1 \leqslant p, k \leqslant n$, $1 \leqslant w \leqslant 100\,000$).

Output

The output should consist of m lines. The ith of those lines should contain a word "YES" or "NO", depending on whether there exists a highway network satisfying the Byteman's requirements (the cheapest network such that each pair of towns is connected by a path) containing the ith highway from the input. You can assume that there exists at least one highway network satisfying Byteman's requirements.

Example

a.in	a.out
6 10 1 2 2 1 6 1 1 5 3 4 1 5 2 6 2 2 3 5 4 3 4 3 5 4 4 5 4 5 6 3	YES YES YES NO YES NO YES YES YES YES

Problem B. Telephone exchange

Input file:	`b.in`
Output file:	`b.out`
Time limit:	3 seconds
Memory limit:	256 Mebibytes

Bytel — a new telecommunication company in Byteland — enters the market with new wireless home phone offer.

In order to introduce the service Bytel is going to build a central office with a tower, which will provide coverage for its users. The location of the tower is known, but has not yet been determined how high the tower will be. The higher the tower, the bigger its range — which means more customers and more money. On the other hand, increasing height of the tower increases the cost of its maintenance.

There are several houses in Byteland. Each house is a simple polygon — closed polyline without self-intersections, multiple vertices or overlapping edges. In each house lives a family which is willing to accept Bytel's offer and pay a fixed monthly fee if and only if the company will cover the whole area of the house.

The monthly maintenance cost of the tower depends on its height. The coverage area of the tower has a form of a circle, and each meter of the tower increases the radius by k meters. However the maintenance of a single meter of a tower costs as many dollars as the number of meters of the tower that are above. This is due to technical reasons — part of the tower which is lower must hold the weight of everything above, hence it has to be wider and stronger, thus more expensive to maintain. The height of the tower expressed in meters has to be an integer.

Having the location and shape of all houses in Byteland, the amount of monthly fee each family is willing to pay and the integer k, find the maximum possible profit that the Bytel company can obtain.

Input

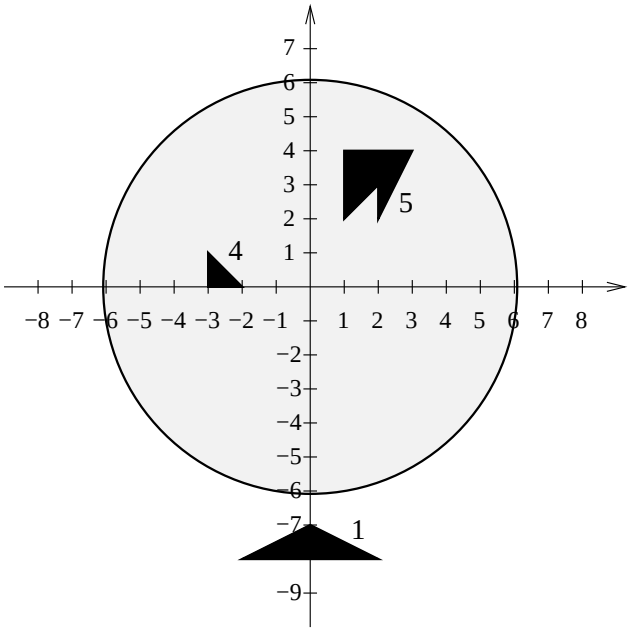
The first line of the input contains two space-separated integers n and k ($1 \leqslant n \leqslant 100\,000$, $1 \leqslant k \leqslant 1\,000\,000$) – the number of houses in Byteland and the increase of the range of the tower per meter of its height.

Each of the next n lines contains a description of a house. Each description starts with two integers m_i and p_i ($3 \leqslant m_i \leqslant 10$, $1 \leqslant p_i \leqslant 1\,000\,000\,000$) denoting the number of vertices of the i-th polygon-shaped house and the monthly fee in dollars that the i-th family is willing to pay. The description of the polygon follows. It consists of a list of coordinates x_{ij}, y_{ij} ($-10^9 \leqslant x_{ij}, y_{ij} \leqslant 10^9$) of m_i consecutive vertices of the polygon. All numbers of the description of a single house are space-separated integers.

It is possible that two houses overlap. Moreover it can happen that the tower (with coordinates $(0, 0)$) is located inside a house.

Output

You should output a single line containing a single integer — the maximum possible monthly profit of the Bytel company. You may assume, that for each input data there exists an integer height of the tower resulting in a positive monthly profit.



Example

b.in	b.out
3 2 3 4 -2 0 -3 0 -3 1 6 5 1 2 2 3 2 2 3 4 1 4 1 3 3 1 -2 -8 0 -7 2 -8	6

Explanation of the example Bytel will have the greatest profit, when the tower will be 3 meters high. Then the radius of the coverage area will be $3 \cdot 2 = 6$ meters. The income from monthly fees will be equal to $4 + 5 = 9$, whereas the maintenance cost of the tower will be equal to $0 + 1 + 2 = 3$, resulting in 6 dollars of monthly profit.

Problem C. Sequence

Input file:	`c.in`
Output file:	`c.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Little Johnny has just learned about number sequences. He likes them very much and has started to create his own, very long sequences. His latest invention is the sequence a — a **non-decreasing** sequence of positive integers with the following property: for every positive integer k it contains exactly a_k occurrences of the number k. In other words, for every k, exactly a_k amongst the numbers $a_1, a_2, a_3, \ldots$ are equal to k.

Johnny managed to write down several first terms of the sequence, however he found a small piece of paper insufficient to generate longer initial fragments of the sequence. Please help Johnny with this problem.

Write a program which:

- reads from the standard input an integer n,
- computes the nth term of the sequence a,
- writes the result to the standard output.

Input

The first and only line of the input contains a single integer n ($1 \leqslant n \leqslant 10^{18}$) — the index of a term of the sequence a.

Output

The first and only line of the output should contain a single integer a_n.

Example

c.in	c.out
5	3

Explanation of the example: $a_5 = 3$ means that the number 5 occurs in the sequence a exactly 3 times; more precisely, $a_9 = a_{10} = a_{11} = 5$.

Problem D. Keyboard

Input file:	d.in
Output file:	d.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Byteman has received an extraordinary keyboard as a gift. There are $n \cdot m$ keys on it, placed in n rows with m columns each. Moreover, all keys except the one in the top left corner are covered with domino tiles of dimensions 1×2, so that there are $\frac{n \cdot m - 1}{2}$ domino tiles in total. At any time, Byteman can move onto the free key one of the domino tiles adjacent to it by the shorter side. He can also press keys, but only if they are not covered.

Byteman would like to test (i.e., press) all keys with vowels, that is, letters `a`, `e`, `i`, `o`, `u` or `y`. What is the minimum number of tile moves neccessary to do that?

Input

The first line of the input contains two integers n and m ($1 \leqslant n, m < 70$) — the dimensions of the keyboard. The next n lines contain m lowercase letters of the English alphabet each, describing the rows of the keyboard. Each of the next n lines contains m characters describing placement of the domino tiles: `.` (ASCII code 46) denotes an uncovered key, `-` (ASCII code 45) denotes a key covered by a domino tile placed horizontally and `|` (ASCII code 124) — a key covered by a tile placed vertically.

Output

If it's not possible for Byteman to press all keys with vowels, your program should output just the single word `"NIE"`. Otherwise output the minimum number of tile moves that Byteman must make in order to press all keys with vowels.

Example

d.in	d.out
3 3 ytr hgf dsa .- ||| |||	2

Problem E. Magic square

Input file:	`e.in`
Output file:	`e.out`
Time limit:	4 seconds
Memory limit:	256 Mebibytes

Byteman has found an old box in his basement. After opening it he saw a big stack of mysterious boards with numbers written on them. Here you can find one of those boards:

1	35	34	4
32	6	7	29
8	30	31	5
33	3	2	36

A moment later Byteman realized that this is a magic square! A magic square is an arrangement of n^2 distinct positive integers in a square grid of size $n \times n$, such that the sum of n numbers in each row, column and in both diagonals is exactly the same. In the above square all rows, columns and diagonals sum up to 74.

Pleased by this discovery, Byteman began browsing other boards. Unfortunately, the numbers on several fields of the boards were blurred. After a detailed visual inspection he noticed that each board misses exactly n numbers, such that no pair shares a row or a column. Byteman loves math puzzles, but is not good at solving them. Could you help him fill in the missing fields, so that the board becomes a magic square once again?

Input

The first line of the input contains a single integer n ($2 \leqslant n \leqslant 1\,000$) — the size of the magic square. Each of the next n lines contains n space-separated integers a_{ij} ($0 \leqslant a_{ij} \leqslant 10^9$); numbers $a_{11}, a_{12}, \ldots, a_{1n}$ represent the first row of the board, numbers $a_{21}, a_{22}, \ldots, a_{2n}$ represent the second row of the board, etc. Positive values of a_{ij} represent the actual content of respective fields, whereas zeros denote fields which are blurred.

Output

Your program should output n lines, each containing n space-separated positive integers not greater than 10^{18}. These n^2 numbers should represent the filled magic square from the input.

You may assume, that for the input data it is always possible to obtain a magic square. If there exists more than one solution, you may output any of them.

Example

e.in	e.out
4 0 35 34 4 32 0 7 29 8 30 31 0 33 3 0 36	1 35 34 4 32 6 7 29 8 30 31 5 33 3 2 36

Problem F. Palindromic equivalence

Input file:	`f.in`
Output file:	`f.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

We will call two words s and t of length n *palindromically equivalent*, if for every pair of numbers i and j such that $1 \leqslant i \leqslant j \leqslant n$, the subword of s consisting of letters from positions i to j, inclusively, is a palindrome if and only if the subword of t consisting of letters from the same positions is a palindrome.

For a given word compute the number of words over the English alphabet that are palindromically equivalent to it, modulo $10^9 + 7$.

Input

The first line of the input contains a non-empty word consisting of lowercase letters of the English alphabet, of length not exceeding 10^6.

Output

Your program should output a single integer — the number of words palindromically equivalent to the one given in the input, modulo $10^9 + 7$.

Example

f.in	f.out
abba	650

Explanation of the example: Only words of the form `xyyx` are palindromically equivalent to `abba`, where `x` and `y` are distinct letters. Since the English alphabet has 26 letters, there are $26 \cdot 25 = 650$ such words in total.

Problem G. Salary redistribution

Input file:	`g.in`
Output file:	`g.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

The Bytesoft company has n employees. Because of the secrecy of its projects, not all of them know each other. Only employees who deal with consecutive phases of a project know each other.

Each project is developed in a sequential manner: first, the employee number 1 (the team leader) creates the first draft of the project, then passes it on to the employee number 2, who performs his task and provides the employee number 3 with the result of his work, and so on. Finally the project arrives at the employee number n, who finishes the project and then forwards it back to the team leader (the employee number 1).

Each employee has a contracted salary. However, the company sometimes does not fulfill the contracts. While the total sum paid to the employees is correct, it often happens that a person gets less money than he or she is guaranteed in the contract, while someone else gets more than they should.

Fortunately, the employees are honest and they agreed that after each payment they will adjust the salary received among them, so that eventually everyone gets exactly the amount guaranteed by the contract.

They have only one problem. Money may only be transferred between a pair of employees that know each other. Consequently, for each $1 < i < n$ employee number i can pass (or receive) money only to (and from) employees $i - 1$ and $i + 1$, whereas the employee number 1 can make transactions with employees 2 and n, while the employee number n can make transactions with employees $n - 1$ and 1.

Since every such redistribution requires many transactions, Bytesoft's employees asked you — an independent programmer — to write a program which calculates the minimum number of transactions needed to adjust salaries.

We assume that employees can not trade more money than they currently have.

Input

The first line of the input contains a single integer n ($1 \leqslant n \leqslant 1\,000\,000$) — the number of Bytesoft's employees.

Each of the next n lines contains two space-separated integers a_i and b_i ($1 \leqslant a_i, b_i \leqslant 1\,000$), representing the contracted salary and the actual amount of money of the i-th employee respectively (expressed in bytedollars). You may assume that $a_1 + a_2 + \ldots + a_n = b_1 + b_2 + \ldots + b_n$.

Output

Your program should output a single line containing a single integer — the minimum number of transactions needed to finish the adjustment process.

If no set of transactions leads to the state where each employee has the amount of money equal to his or her contracted salary your program should output a single word `"NIE"`.

Example

g.in	g.out
4 10 13 5 4 5 6 8 5	2

Explanation of the example: The adjustment can be performed in two steps. In the first transaction the employee number 3 gives one bytedollar to the employee number 2. In the second transaction the employee number 1 gives three bytedollars to the employee number 4.

Problem H. Vacation

Input file:	h.in
Output file:	h.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Byteman wants to pick days in which he will take a vacation. He bases his choices on the weather forecast for the next $3n$ days. He is only interested in the highest temperature anticipated for a given day.

Byteman is additionally constrained by terms set by his boss. During any consecutive n days he can take no more than k days of absence. What is the best way to plan the vacation, so that the sum of temperatures in the chosen days is maximized?

Input

The first line of the input contains two integers n and k ($1 \leqslant n \leqslant 200, 1 \leqslant k \leqslant 10, k < n$). In the second line there are $3n$ positive integers not exceeding 10^6, describing temperatures anticipated for each of the next $3n$ days.

Output

Your program should output a single integer — the maximum possible sum of temperatures during vacation days picked with respect to boss' terms.

Example

h.in	h.out
5 3 14 21 9 30 11 8 1 20 29 23 17 27 7 8 35	195

Problem I. Garages

Input file:	`i.in`
Output file:	`i.out`
Time limit:	5 seconds (8 for Java)
Memory limit:	256 Mebibytes

In Byteland there is exactly one road going from east to west. It leads through exactly m towns. In each of the towns there is a garage which offers painting cars. Each of the garages has set up a promotion, in which cars can be painted from the colour a_j to the colour b_j for free (here j represents the number of town, $j = 1, 2, \ldots, m$). These colours are selected independently for every garage. Each garage offers exactly one such promotion.

There are n cars travelling along the road, from east to west. The drivers in Byteland do not like spending money, but they really like having their cars freshly painted. Therefore each driver tries to paint his car in every single town, but does so only if this operation is free for him.

Knowing the initial colours of all cars, we would like to find out what will the colours of the cars be when they all reach the western end of the road.

Assume that the road is unidirectional, i.e., the cars only travel from east to west.

Input

The first line of the standard input contains three integers n, m and k ($1 \leqslant n, m, k \leqslant 1\,000\,000$) separated by single spaces, denoting the number of cars, the number of towns and the number of possible colours respectively. The colours are numbered from 1 to k.

The second line contains n space-separated integers k_i ($1 \leqslant k_i \leqslant k$) which denote the initial colours of subsequent cars.

The following m lines contain descriptions of garages in the order in which they are visited along the road. Each such description contains two space-separated integers a_j and b_j ($1 \leqslant a_j, b_j \leqslant k$) meaning that the garage in the jth town offers free painting of cars from the colour a_j to the colour b_j.

Output

The first and only line of the standard output should contain n space-separated integers, denoting the colours of subsequent cars after reaching the western end of the road.

Example

i.in	i.out
5 3 4 1 2 3 4 2 2 4 1 2 4 1	2 1 3 1 1

Explanation of the example: In the first garage, the second and fifth cars get painted, so the sequence of colours becomes: `1 4 3 4 4`. In the second garage, only the first car gets painted — the new sequence of colours is: `2 4 3 4 4`. Finally, in the third garage, the second, fourth and fifth cars get painted. Thus the final sequence of colours is: `2 1 3 1 1`.

Problem J. Towers

Input file:	`j.in`
Output file:	`j.out`
Time limit:	3 seconds
Memory limit:	256 Mebibytes

Little Johnny has built a number of towers of various heights using building bricks. Each brick has an integer number written on it. Johnny would like to construct a tower with the greatest possible sum of numbers written on bricks this tower is composed of; however, he does not want to completely destroy towers already constructed. Therefore, he decided that the only thing he will do is performing the following operation on pairs of towers of different heights:

- remove from the higher tower the l top blocks, where l is the height of the shorter tower, and create from them a new tower of height l (without changing the order of the blocks);
- afterwards, take the shorter tower and put it on top of the one we removed blocks from.

In this way, the two newly constructed towers have the same heights as the towers before the operation, but numbers written on corresponding blocks may be different.

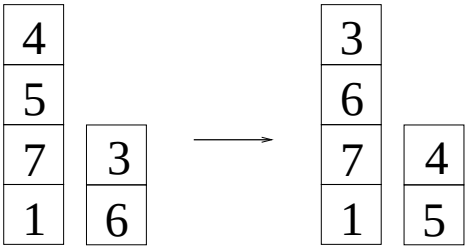


Figure: One operation on towers (4, 5, 7, 1) and (3, 6).

Johnny would like to maximize the value of the most valuable tower, where the value of a tower is the sum of numbers written on bricks it is composed of. He can make as many operations as he wants to.

Input

The first line of the input contains a single integer n ($1 \leqslant n \leqslant 5 \cdot 10^5$) — the number of towers built by Johnny. Each of the next n lines contains a description of one tower. The $(i+1)$-th line contains an integer w_i ($1 \leqslant w_i \leqslant 10^6$) — the height of the i-th tower — followed by w_i space-delimited integers $x_1, x_2, \ldots, x_{w_i}$ ($-10^6 \leqslant x_k \leqslant 10^6$). The number x_k is the number written on the k-th block (counting from the top) of the i-th tower. You may assume that the total number of blocks in all towers does not exceed 10^6.

Output

Your program should output a single line containing a single integer — the maximum possible value of a tower that Johnny can build.

Example

j.in	j.out
4 4 4 5 7 1 2 3 6 1 4 1 1	18

Explanation of the example: Johnny can use towers 2 and 3 to build the tower (4, 6). Afterwards, he can use this tower together with tower number 1 to build the tower (4, 6, 7, 1), which has value equal to 18.