Problem A. Brackets

Input file:	`brackets.in`
Output file:	`brackets.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Regular brackets sequence is the sequence of $2n$ characters each of which is either "`(`" (opening bracket), or "`)`" (closing bracket), such that each prefix contains no more closing brackets than opening ones, and the whole sequence contains n opening and n closing brackets. For example, "", "`((()))`", "`()()()()`", "`(())()`" are the regular brackets sequences, but "`())(()`" is not, because its prefix "`())`" contains more closing than opening brackets, neither is "`((`", because it contains more opening brackets than closing.

For each opening bracket you can find the *corresponding* closing bracket — the one following it, such that there is a regular brackets sequence between them.

The generalization of the regular brackets sequence if the regular brackets sequence with k types of brackets. Let each bracket have its *type* — an integer number from 1 to k. The sequence of opening and closing brackets with types is regular if it is a regular brackets sequence and the corresponding closing bracket for each opening bracket has the same type as the opening bracket itself. For example, "$(_1(_2)_2)_1(_1)_1$" is the regular brackets sequence with two types of brackets, but "$(_1(_2)_1)_2(_1)_1$" is not.

If you introduce some order on $2k$ typed brackets, you can consider *lexicographical* order on all regular brackets sequences of length $2n$. You are given n, k, the order on typed brackets and the regular brackets sequence. Find the next regular brackets sequence in the lexicographical order.

Input

The first line of the input file contains t — the number of tests ($1 \le t \le 10000$). The description of t test cases follows.

The first line of each description contains two integer numbers: n and k ($1 \le n \le 100\,000$, $1 \le k \le 10\,000$). The second line contains $2k$ integer numbers — the permutation of the set $\{-k, -(k-1), \ldots, -2, -1, 1, 2, \ldots, k\}$. It lists the brackets from the least to the greatest in the given order. Positive number i means the opening bracket of the i-th type, negative number $-i$ means the closing bracket of the i-th type. The third line contains $2n$ integer numbers from $-k$ to k (except 0) and describes the regular brackets sequence.

The sum of n and the sum of k in all test cases don't exceed 100 000.

Output

For each test case output $2n$ integer numbers — the next regular brackets sequence after the one from the input file in lexicographical order. If the sequence in the input file is the last one, output the first one.

Example

brackets.in	brackets.out
2 3 2 1 -1 2 -2 1 2 -2 -1 1 -1 3 2 1 -1 2 -2 1 2 -2 -1 2 -2	1 2 -2 -1 2 -2 1 2 -2 2 -2 -1

Problem B. Car Wash

Input file:	`carwash.in`
Output file:	`carwash.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Ben is the owner of a car wash. Ben offers car washing and dry-cleaning of the car compartment. His wash is located in the capital of Edgeland, and he often serves clients from government. Recently Ben has received an order of washing and dry-cleaning of n cars. And this order must be executed as fast as possible!

After preliminary investigation, Ben found out that the i-th car can be washed in a_i minutes and cleaned in b_i minutes. Washing and cleaning are performed at different buildings, so they cannot be performed simultaneously for the same car. The order of the two operations for a car is irrelevant, neither is the order of processing the cars. Ben's best workers Tom and Jerry were called to perform the order. Tom will wash the cars, and Jerry will dry-clean their compartments. Each worker can work with one car at a moment, and due to quality requirements, it is not allowed to switch from one car to another until the one is ready (this condition is independent for each worker, different workers can process cars in any order).

Help Ben to find out how the cars should be processed by the workers, so that all cars were finished as soon as possible.

Input

The first line of the input file contains n ($1 \le n \le 10\,000$). The following n lines contain two integer numbers each: a_i and b_i ($1 \le a_i, b_i \le 10^5$).

Output

The first line of the output file must contain t — the number of minutes after the process has started when all cars will be washed and dry-cleaned. You must minimize this number.

The following n lines must contain two integer numbers each — the number of minutes after the start of the process when the washing and the dry-cleaning of the corresponding car must start, respectively.

If there are several optimal solutions, output any one.

Example

carwash.in	carwash.out
6 10 6 7 9 3 8 1 2 12 7 6 6	39 11 26 4 17 0 7 3 15 27 0 21 32

Problem C. Painting Cottages

Input file:	`cottages.in`
Output file:	`cottages.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The new cottage settlement is organized near the capital of Flatland. The construction company that is building the settlement has decided to paint some cottages pink and others — light blue. However, they cannot decide which cottages must be painted which color.

The director of the company claims that the painting is *nice* if there is at least one pink cottage, at least one light blue cottage, and it is possible to draw a straight line in such a way that pink cottages are at one side of the line, and light blue cottages are at the other side of the line (and no cottage is on the line itself). The main architect objects that there are several possible nice paintings.

Help them to find out how many different nice paintings are there.

Input

The first line of the input file contains n — the number of the cottages ($1 \le n \le 300$). The following n lines contain the coordinates of the cottages — each line contains two integer numbers x_i and y_i ($-10^4 \le x_i, y_i \le 10^4$).

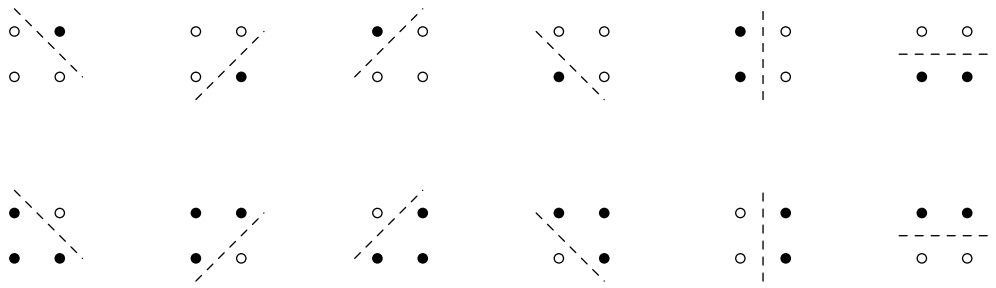
Output

Output one integer number — the number of different nice paintings of the cottages.

Example

cottages.in	cottages.out
4 0 0 1 0 1 1 0 1	12

The possible nice paintings are shown on the following picture.



Problem D. Dinner Problem

Input file:	`dinner.in`
Output file:	`dinner.out`
Time limit:	1 second
Memory limit:	256 megabytes

A group of k students from Cooking University living in the campus decided that each day of the semester one of them will prepare the dinner for the whole company. The semester lasts for n days.

In sake of fairness they decided that each of the students must prepare the dinner at least once during the semester. Now they wonder how many ways are there to plan the semester — to decide for each day which student would make a dinner that day. Help them to find that out.

Input

The input file contains two integer numbers k and n ($1 \le k \le n \le 100$).

Output

Output one number — the number of ways.

Example

dinner.in	dinner.out
2 3	6

There are six ways: $(1, 1, 2)$, $(1, 2, 1)$, $(2, 1, 1)$, $(1, 2, 2)$, $(2, 1, 2)$, $(2, 2, 1)$.

Problem E. Minima

Input file:	`minima.in`
Output file:	`minima.out`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are given an array $x[1 \ldots n]$ and a number m. For all i from 1 to $n-m+1$ find the minimum among $x[i], x[i+1], \ldots, x[i+m-1]$ and return the sum of those minima.

Input

The first line of the input file contains three integer numbers: n, m and k ($1 \le n \le 30\,000\,000$, $1 \le m \le n$, $2 \le k \le \min(n, 1000)$). The second line of the input file contains three integer numbers: a, b and c ($-2^{31} \le a, b, c \le 2^{31} - 1$). The third line of the input file contains k integer numbers: $x[1], x[2], \ldots, x[k]$ ($-2^{31} \le x[i] \le 2^{31} - 1$).

The rest of the array is calculated using the following formula: $x[i] = f(a \cdot x[i-2] + b \cdot x[i-1] + c)$. Here $f(y)$ returns such number $-2^{31} \le z \le 2^{31} - 1$ that $y - z$ is divisible by 2^{32}.

Output

Print one integer number — the sum of minima of all subarrays of length m of the given array.

Example

minima.in	minima.out
10 3 2 1 1 0 0 1	33
1000000 15 5 283471207 23947205 3 17625384 939393931 1838388 912740247 290470294	-1879262596173354

Problem F. Move to Front

Input file:	`mtf.in`
Output file:	`mtf.out`
Time limit:	1 second
Memory limit:	256 megabytes

Move-to-Front is a method of transforming sequences of positive integer numbers, that is used in some compression algorithms, such as Burrows-Wheeler transform.

Initially all positive integer numbers are organized as an ordered list L in their natural order. Consider a sequence $a_1, a_2, \ldots, a_n$ of positive integer numbers. It is encoded as a sequence $b_1, b_2, \ldots, b_n$ in the following way. Let the part of the sequence from a_1 to a_{i-1} be encoded. The position of a_i in the current list L is considered. It is assigned to b_i, and a_i is moved to the beginning of the list L.

For example, the sequence $3, 3, 3, 2, 2, 2, 2, 2, 3, 1, 3, 3, 2$ is encoded as $3, 1, 1, 3, 1, 1, 1, 1, 2, 3, 2, 1, 3$.

You are given a sequence $a_1, a_2, \ldots, a_n$, you must encode it using Move-to-Front method, and output the resulting sequence $b_1, b_2, \ldots, b_n$.

Input

The first line of the input file contains integer number n ($1 \le n \le 100\,000$). The second line contains n integer numbers a_i, ranging from 1 to 10^9.

Output

Output n integer numbers — the sequence $b_1, b_2, \ldots, b_n$.

Example

mtf.in	mtf.out
13 3 3 3 2 2 2 2 2 3 1 3 3 2	3 1 1 3 1 1 1 1 2 3 2 1 3

Problem G. TV Show

Input file:	`show.in`
Output file:	`show.out`
Time limit:	1 second
Memory limit:	256 megabytes

Recently various TV shows started to gain popularity among young people who like to test their luck. One such show is running on ZhTV channel in China.

The show proceeds as follows. There is a special game board which has the form of a rectangle with n rows and n columns. The cells on the main diagonal of the board contain one hieroglyph each. There are also m phrases each of which contains n hieroglyphs. For each i there are at most two phrases such that their i-th hieroglyph is equal to the hieroglyph written in the i-th row of the board.

The player has to put phrases to the rows of the board in such way that if the phrase is put to the i-th row, the i-th hieroglyph of the phrase must coincide with the hieroglyph written in the corresponding cell. Also hieroglyphs in each column must be distinct.

For each phrase put to the board the player gets some small prize, and she gets a super prize if she puts a phrase to every row of the board. Help the player to determine whether she can win the super prize.

Input

We will denote hieroglyphs with integer numbers from 1 to 10^9.

The first line of the input file contains n — the number of rows on the board ($1 \le n \le 200$). The next line contains n integer numbers — hieroglyphs written on the diagonal.

The third line contains an integer number m ($n \le m \le 2n$) followed by m lines. Each line contains n integer numbers — hieroglyphs of the corresponding phrase. For each i there are at most two phrases such that their i-th hieroglyph is equal to the hieroglyph written in the i-th row of the board.

Output

If the super prize can be won, print "`YES`" at the first line of the output file. The second line must contain n integer numbers — the phrases to put to the board, from top to bottom. Phrases are numbered from 1 to m in order they are given in the input file.

If the super prize cannot be won, print "`NO`" at the first line of the output file.

Example

show.in	show.out
4 1 2 3 4 7 1 5 2 5 1 5 4 1 2 3 3 1 3 2 4 1 2 2 2 2 3 3 3 3 5 4 5 4	YES 2 5 6 7
4 1 2 3 4 7 1 5 2 5 1 3 4 1 2 3 3 1 3 2 4 1 2 2 2 2 3 3 3 3 5 4 5 4	NO

Problem H. Hard Test

Input file:	`test.in`
Output file:	`test.out`
Time limit:	1 second
Memory limit:	256 megabytes

Andrew is having a hard time preparing his 239-th contest for Petrozavodsk. This time the solution to the problem is based on Dijkstra algorithm and Andrew wants to prepare the hard test for the algorithm.

The Dijkstra algorithm is used to find the shortest path from a source vertex to all other vertices in a graph. The algorithm acts as follows. Let G be a weight directed graph with vertex set V, edge set E and weight function $w : E \to \mathbb{R}^+$. Let all vertices be reachable from vertex s. The algorithm uses a set of vertices U, first initialized as empty. Each vertex is labeled with either an integer number, or with $+\infty$. Initially all vertices are labeled with $+\infty$, and the vertex s is labeled with 0. Denote the label of vertex v as $d[v]$.

A step of the algorithm is the following: the vertex with the minimal label that doesn't belong to U is selected. Let this vertex be u. The vertex u is added to the set U, and each edge $uv \in E$ is *relaxed*. The relaxation replaces $d[v]$ with $\min(d[v], d[u] + w(uv))$. The algorithm is over when all vertices belong to U. If the label of the vertex v has changed, the relaxation is said to be *active*.

Now Andrew would like to create a graph with n vertices and m edges, such that the Dijkstra algorithm makes as many active relaxations as possible. Help him to create such graph. To avoid nondeterminism, each time when selecting a vertex with minimal label among vertices that are not in U there must be exactly one vertex with the minimal label.

Input

The first line of the input file contains two integer numbers: n and m — the number of vertices and the number of edges in the graph Andrew would like to create ($4 \le n \le 1000$, $n - 1 \le m \le n^2/5$).

Output

Output m lines — the edges of the graph. Each line must contain three integer numbers: the beginning of the edge, the end of the edge and the weight of the edge. All weights must be non-negative and must not exceed 10^6. All vertices must be reachable from vertex 1. If Dijkstra algorithm is run with $s = 1$ there must be maximal possible number of active relaxations among all graphs with n vertices and m edges. There must be no loops and no parallel edges.

Example

test.in	test.out
4 3	1 2 0 1 3 1 1 4 2
5 5	1 2 0 1 3 1 2 4 4 3 4 2 1 5 2

Problem I. Travel Agency

Input file:	`travel.in`
Output file:	`travel.out`
Time limit:	1 second
Memory limit:	256 megabytes

Anthony is working in the intergalaxy travel agency. He often meets the requests to find a path from one planet to another using available spaceship routes. Unfortunately, spaceships have only limited number of routes, so passengers usually have to make connections at intermediate planets.

Anthony noticed that some planets are used as intermediate more often than other. He decided to make an investigation — for each planet A he would like to know how many pairs of distinct planets (B, C) are there, such that any path from planet B to planet C visits planet A. Help him!

Input

The first line of the input file contains two integer numbers: n and m — the number of planets and the number of spaceship routes, respectively ($2 \le n \le 20\,000$, $1 \le m \le 200\,000$). The following m lines describe spaceship routes. Each route is bidirectional, and is described with the numbers of planets it connects. It is possible to get from any planet to any other.

Output

Output n integer numbers — for each planet A output the number of pairs of distinct planets such that any path from one to another goes through A.

Example

travel.in	travel.out
7 9 1 2 1 3 1 4 1 5 1 6 1 7 2 3 4 5 6 7	18 6 6 6 6 6 6

Problem J. Triatrip

Input file:	`triatrip.in`
Output file:	`triatrip.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

The travel agency "Four Russians" is offering the new service for their clients. Unlike other agencies that only suggest *one-way* or *roundtrip* for airline tickets to their customers, "Four Russians" offers the brand new idea — *triatrip*. Triatrip traveler starts in some city A, flies to some city B, then flies to some city C, and returns to the city A.

Now the managers of the agency started to wonder, how many different triatrips they can offer to their customers. Given a map of all possible flights, help them to find that out.

Input

The first line of the input file contains two integer numbers n — the number of cities that are served by airlines that agree to sell their tickets via the agency ($3 \le n \le 1500$). The following n lines contain a sequence of n characters each — the j-th character of the i-th line is '`+`' if it is possible to fly from the i-th city to the j-th one, and '`-`' if it is not. The i-th character of the i-th line is '`-`'.

Output

Output one integer number — the number of triatrips that the agency can offer to its customers.

Example

triatrip.in	triatrip.out
4 --+- +--+ -+-- --+-	2