Problem A. Blocks

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

Bytie has got a set of wooden blocks for his birthday. The blocks are indistinguishable from one another, as they are all cubes of the same size. Bytie forms piles by putting one block atop another. Soon he had a whole rank of such piles, one next to another in a straight line. Of course, the piles can have different heights.

Bytie's father, Byteasar, gave his son a puzzle. He gave him a number k and asked to rearrange the blocks in such a way that the number of successive piles of height at least k is maximised. However, Bytie is only ever allowed to pick the top block from a pile strictly higher than k and place it atop one of the piles next to it. Further, Bytie is not allowed to form new piles, he can only move blocks between those already existing.

Input

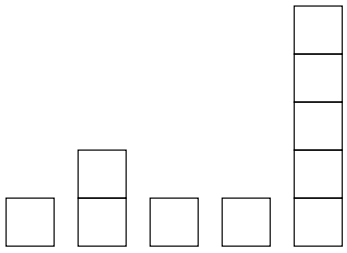
In the first line of the standard input there are two integers separated by a single space: n ($1 \leq n \leq 1\,000\,000$), denoting the number of piles, and m ($1 \leq m \leq 50$), denoting the number of Byteasar's requests. The piles are numbered from 1 to n. In the second line there are n integers $x_1, x_2, \ldots, x_n$ separated by single spaces ($1 \leq x_i \leq 1\,000\,000\,000$). The number x_i denotes the height of the i-th pile. The third line holds m integers $k_1, k_2, \ldots, k_m$ separated by single spaces ($1 \leq k_i \leq 1\,000\,000\,000$). These are the subsequent values of the parameter k for which the puzzle is to be solved. That is, the largest number of successive piles of height at least k that can be obtained by allowed moves is to be determined for each given value of the parameter k.

Output

Your program should print out m integers, separated by single spaces, to the standard output — the i-th of which should be the answer to the puzzle for the given initial piles set-up and the parameter k_i.

Examples

Standard input	Standard output
5 6 1 2 1 1 5 1 2 3 4 5 6	5 5 2 1 1 0



Problem B. Conspiracy

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

Hostile Bitotia launched a sneak attack on Byteotia and occupied a significant part of its territory. The King of Byteotia, Byteasar, intends to organise resistance movement in the occupied area. Byteasar naturally started with selecting the people who will form the skeleton of the movement. They are to be partitioned into two groups: *the conspirators* who will operate directly in the occupied territory, and *the support group* that will operate inside free Byteotia.

There is however one issue — the partition has to satisfy the following conditions:

- Every pair of people from the support group have to know each other — this will make the whole group cooperative and efficient.
- The conspirators must not know each other.
- None of the groups may be empty, i.e., there has to be at least one conspirator and at least one person in the support group.

Byteasar wonders how many ways there are of partitioning selected people into the two groups. And most of all, whether such partition is possible at all. As he has absolutely no idea how to approach this problem, he asks you for help.

Input

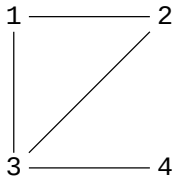
The first line of the input holds one integer n ($2 \leq n \leq 2\,000$), denoting the number of people engaged in forming the resistance movement. These people are numbered from 1 to n (for the sake of conspiracy!). The n lines that follow describe who knows who in the group. The i-th of these lines describes the acquaintances of the person i with a sequence of integers separated by single spaces. The first of those numbers, k_i ($0 \leq k_i \leq n-1$), denotes the number of acquaintances of the person i. Next in the line there are k_i integers $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i}$ — the numbers of i's acquaintances. The numbers $a_{i,j}$ are given in increasing order and satisfy $1 \leq a_{i,j} \leq n$, $a_{i,j} \neq i$. You may assume that if x occurs in the sequence a_i (i.e., among i's acquaintances), then also i occurs in the sequence a_x (i.e., among x's acquaintances).

Output

In the first and only line of the output your program should print out one integer: the number of ways to partition selected people into the conspirators and the support group. If there is no partition satisfying aforementioned conditions, then 0 is obviously the right answer.

Examples

Standard input	Standard output
4 2 2 3 2 1 3 3 1 2 4 1 3	3



Explanation of the example: There are three ways of partitioning these people into the groups. The group of conspirators can be formed by either those numbered 1 and 4, those numbered 2 and 4, or the one numbered 4 alone.

Problem C. Fire Brigade

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	8 seconds
Memory limit:	256 mebibytes

In the capital of Byteotia, Bytau, the layout of streets is highly regular. Every street leads either from north to south, or from east to west. Therefore every north-south street intersects every east-west street in exactly one spot. Furthermore, along every street its successive intersections are exactly 1 km apart.

Bytau is not only the capital, but also one of the oldest cities in Byteotia. No wonder that there are as many as n historic buildings, each at one of the intersections. The City Council cares for their protection very much, and is now concerned with the risk of fire. Hence they have decided to establish two main fire stations in the city. Each monument is going to be protected by the nearest station; by both, should both fire stations be equally close.

Housing is very dense in Bytau, so Euclidean distance is not the measure of choice. The distance between a monument and fire station should rather be defined as the length of the shortest path along the streets between them.

The City Council has prepared several projects of the stations' location. And you have been asked to determine, for each of them, the number of monuments protected by: the first station only, the second station only, and both stations, respectively.

Input

In the first line of the input there are four integers n, m, z and p ($1 \le n, m \le 1\,000\,000\,000$, $1 \le z, p \le 100\,000$) separated by single spaces and denoting respectively: the number of streets leading from north to south, the number of streets leading from east to west, the number of historic buildings in Bytau, and the number of projects proposed by the City Council.

The north-south streets are numbered from 1 to n, west to east. The east-west streets are numbered from 1 to m, north to south. The intersection of x-th north-south and y-th east-west street will be denoted by the coordinates (x, y).

In each of the following z lines there are two integers x_i and y_i ($1 \le x_i \le n$, $1 \le y_i \le m$) separated by a single space and denoting the coordinates of the i-th monument. No pair of different monuments is located at the same intersection.

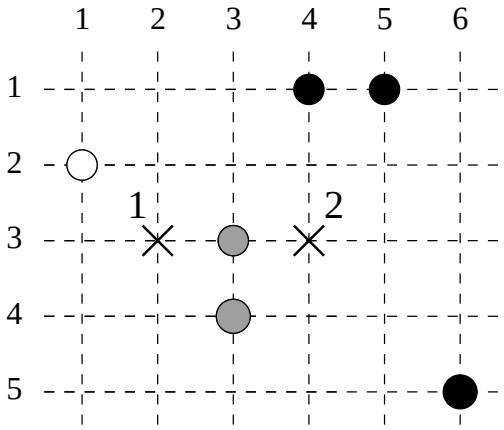
Each of the following p lines contains one proposal of the City Council — four integers $x_{j,1}$, $y_{j,1}$, $x_{j,2}$, $y_{j,2}$ separated by single spaces, $1 \le x_{j,1}, x_{j,2} \le n$, $1 \le y_{j,1}, y_{j,2} \le m$, $(x_{j,1}, y_{j,1}) \ne (x_{j,2}, y_{j,2})$. The coordinates $(x_{j,1}, y_{j,1})$ and $(x_{j,2}, y_{j,2})$ describe the intersections at which the fire stations are to be located according to the j-th proposal ($1 \le j \le p$).

Output

Your program should print out exactly p lines on the output. There should be three integers in the j-th line, denoting: the number of monuments protected by the first station of j-th proposal of the City Council only, the number of monuments protected by the second station only and the number of monuments protected by both stations, respectively. These numbers should be separated by single spaces.

Examples

Standard input	Standard output
6 5 6 1 1 2 6 5 5 1 3 3 3 4 4 1 2 3 4 3	1 3 2



The dashed lines in the figure represent streets, circles — locations of monuments, while crosses — proposed locations of fire stations. White circles depict monuments protected by the first station only, black circles — by second station only, while grey ones — by both stations.

Problem D. Hamsters

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Byteasar breeds hamsters. Each hamster has a unique name, consisting of lower case letters of the English alphabet. The hamsters have a vast and comfortable cage. Byteasar intends to place a display under the cage to visualize the names of his hamsters. This display is simply a sequence of letters, each of which can be either lit or not independently. Only one name will be displayed simultaneously. The lit letters forming the name have to stand next to each other, i.e., form a contiguous subsequence.

Byteasar wants to be able to display the names of the hamsters on at least m different positions. However, he allows displaying the same name on multiple different positions, and does not require to be able to display each and every hamster's name. Note that the occurrences of the names on the display can overlap. You can assume that no hamster's name occurs (as a contiguous fragment) in any other hamster's name. Bytesar asks your help in determining the minimum number of letters the display has to have.

In other words, you are to determine the minimum length of a string (consisting of non-capital letters of the English alphabet) that has at least m total occurrences of the hamsters' names (counting multiplicities). (We say that a string s occurs in the string t if s forms a contiguous fragment of t.)

Input

The first line of the input holds two integers n and m ($1 \leq n \leq 200$, $1 \leq m \leq 10^9$), separated by a single space, that denote the number of Byteasar's hamsters and the minimum number of occurrences of the hamsters' names on the display. Each of the following n lines contains a non-empty string of non-capital letters of the English alphabet that is the hamster's name. The total length of all names does not exceed 100 000 letters.

Output

The first and only line of the output should hold a single integer — the minimum number of letters the display has to have.

Examples

Standard input	Standard output
4 5 monika tomek szymon bernard	23

Explanation of the example: The shortest display could be, for example: `szymonikatomekszymonika`. It has 5 occurrences of the hamsters' names in total: `szymon` and `monika` occur twice each, `tomek` just once, and `bernard` does not occur in it at all.

Problem E. Lightning Conductor

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	5 seconds
Memory limit:	256 mebibytes

Progressive climate change has forced the Byteburg authorities to build a huge lightning conductor that would protect all the buildings within the city. These buildings form a row along a single street, and are numbered from 1 to n.

The heights of the buildings and the lightning conductor are non-negative integers. Byteburg's limited funds allow construction of only a single lightning conductor. Moreover, as you would expect, the higher it will be, the more expensive.

The lightning conductor of height p located on the roof of the building i (of height h_i) protects the building j (of height h_j) if the following inequality holds:

$$h_j \leq h_i + p - \sqrt{|i-j|},$$

where $|i-j|$ denotes the absolute value of the difference between i and j.

Byteasar, the mayor of Byteburg, asks your help. Write a program that, for every building i, determines the minimum height of a lightning conductor that would protect all the buildings if it were put on top of the building i.

Input

In the first line of the input there is a single integer n ($1 \leq n \leq 500\,000$) that denotes the number of buildings in Byteburg. Each of the following n lines holds a single integer h_i ($0 \leq h_i \leq 1\,000\,000\,000$) that denotes the height of the i-th building.

Output

Your program should print out exactly n lines to the output. The i-th line should give a non-negative integer p_i denoting the minimum height of the lightning conductor on the i-th building.

Examples

Standard input	Standard output
6 5 3 2 4 2 4	2 3 5 3 5 4

Problem F. Lollipop

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

Byteasar runs a confectionery in Byteburg. Strawberry-vanilla flavoured lollipops are the favourite of local children. These lollipops are composed of multiple segments of the same length, each segment of either strawberry or vanilla flavour. The price of the lollipop is the sum of the values of its segments, where a vanilla segment costs one bythaler, while a strawberry segments costs two.

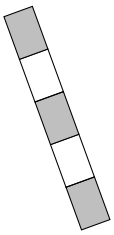


Рис. 1: An exemplary lollipop of five segments, three strawberry flavoured and two vanilla, alternately. The price of this lollipop is 8 bythalers.

Currently Byteasar is left with only one lollipop, though possibly very long. As a salesman, he knows only too well that probably no one will want to buy the whole lollipop. For this reason he thinks of breaking the lollipop at the joints of the segments in order to get a shorter lollipop. Each fragment for sale, of course, must stay in one piece.

Byteasar vast experience of a salesman, as well as his understanding of children psychology, tell him that his young customers will most likely want to spend all their money on a single lollipop. With this in mind, he wonders for which values of k the lollipop he has can be broken down in such a way that as a result one would get, among other pieces, a lollipop worth exactly k bythalers. Naturally, he is interested in the way of breaking the lollipop as well. As this task overwhelms him, he asks you for help.

Input

In the first line of the input there are two integers n and m ($1 \leq n, m \leq 1\,000\,000$), separated by a single space. These denote, respectively, the number of segments of the last lollipop left in store, and the number of values of k to consider. The lollipop's segments are numbered from 1 to n. The second line gives an n-letter description of the lollipop, consisting of letters `T` and `W`, where `T` denotes a strawberry flavoured segment, while `W` — vanilla flavoured; the i-th of these letters specifies the flavour of the i-th segment. In the following m lines successive values of k ($1 \leq k \leq 2\,000\,000$) to consider are given, one per line.

Output

Your program should print out exactly m lines, giving, one per line, the results for successive values of k, to the output. If for a given value of k it is impossible to break the lollipop in such a way that there is a contiguous fragment worth exactly k bythalers, then the word `NIE` (*no* in Polish) should be printed. Otherwise, two integers l and r ($1 \leq l \leq r \leq n$), separated by single spaces, should be printed, such that the fragment of the lollipop composed of the segments numbered from l to r inclusively is worth exactly k bythalers. If there are multiple such pairs, your program is free to choose one arbitrarily.

Examples

Standard input	Standard output
5 3 TWTWT 5 1 7	1 3 2 2 NIE

Explanation of the example: The example considers the lollipop from Fig. 1. The segments numbered from 1 to 3 form a `TWT` lollipop, worth 5 bythalers. Segment number 2 has vanilla flavour and costs 1 bythaler. There is no way of getting a lollipop worth 7 bythalers out of the one given.

Problem G. Minima Game

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Alice and Bob learned the minima game, which they like very much, recently. The rules of the game are as follows. A certain number of cards lies on a table, each inscribed with a positive integer. The players make alternate moves, Alice making the first one. A move consists in picking an arbitrary positive number of cards from the table. For such move the player receives a number of points equal to the minimum of the numbers inscribed on the cards he collected. The game ends when the last card is removed from the table. The goal of each player is maximizing the difference between their and their opponent's score.

Alice and Bob have duly noted that there is an optimal strategy in the game. Thus they are asking you to write a program that, for a given set of cards, determines the outcome of the game when both players play optimally.

Input

In the first line of the input there is one integer n ($1 \leq n \leq 1\,000\,000$) given, denoting the number of cards. The second line holds n positive integers $k_1, k_2, \ldots, k_n$ ($1 \leq k_i \leq 10^9$), separated by single spaces, that are inscribed on the cards.

Output

Your program should print out a single line with a single integer to the output — the number of points by which Alice wins over Bob, assuming they both play optimally; if it is Bob who has more points, the result should be negative.

Examples

Standard input	Standard output
3 1 3 1	2

Explanation of the example: Alice picks a single card, the one with 3, which grants her three points. Bob takes both the remaining cards, for which he receives one point. The games ends with three to one score, hence Alice wins by two points.

Problem H. Nasty sets

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Let p and q be positive integers, assume that p and q are coprime. A set $S \subseteq \{0, 1, \dots\}$ is called *nasty* if both the following conditions hold:

1. $0 \in S$
2. if $x \in S$ then $x + p \in S$ and $x + q \in S$.

Write a program which computes the number of different nasty sets for the given p and q, modulo m.

Input

The input consists of three integers p, q and m ($2 \le p, q \le 500\,000$, $2 \le m \le 1\,000\,000\,000$), separated by single spaces.

Output

Your program should output the number of different nasty sets for p and q modulo m.

Examples

Standard input	Standard output
2 3 10	2

Problem I. Ones

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	7 seconds
Memory limit:	256 mebibytes

Let x be a sequence of zeros and ones. An *utterly forlorn one* (UFO) in x is the extreme (either first or last) one that additionally does not neighbour with any other one. For instance, the sequence `10001010` has two UFOs, while the sequence `1101011000` has no UFO, and the sequence `1000` has only one UFO.

Let us denote the total number of UFOs in the binary representations of the numbers from 1 to n with $ufo(n)$. For example, $ufo(5) = 5$, $ufo(64) = 59$, $ufo(128) = 122$, $ufo(256) = 249$.

We will be working with very large numbers. Therefore, we shall represent them in a *succinct* way. Suppose x is a positive integer and x_2 is its binary representation (starting with 1). Then the succinct representation of x is the sequence $REP(x)$ consisting of positive integers denoting the lengths of successive blocks of the same digits. For example:

$$\begin{aligned} REP(460\,288) &= REP(1110000011000000000_2) &&= (3,5,2,9) \\ REP(408) &= REP(110011000_2) &&= (2,2,2,3) \end{aligned}$$

Your task is to write a program that finds the sequence $REP(ufo(n))$ given $REP(n)$.

Input

The first line of the input holds one integer k ($1 \le k \le 1\,000\,000$) denoting the length of the succinct representation of a positive integer n. The second line of the standard input holds k integers $x_1, x_2, \ldots, x_k$, ($0 < x_i \le 1\,000\,000\,000$), separated by single spaces. The sequence $x_1, x_2, \ldots, x_k$ forms the succinct representation of the number n. You may assume that $x_1 + x_2 + \ldots + x_k \le 1\,000\,000\,000$, i.e., $0 < n < 2^{1\,000\,000\,000}$.

Output

Your program is to print out two lines to the standard output. The first one should contain a single positive integer l. The second line should hold l positive integers $y_1, y_2, \ldots, y_l$, separated by single spaces. The sequence $y_1, y_2, \ldots, y_l$ is to form the succinct representation of $ufo(n)$.

Examples

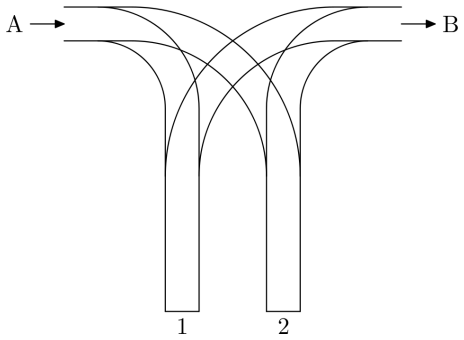
Standard input	Standard output
6 1 1 1 1 1 1	5 1 1 2 1 1

Explanation of the example: The sequence $1, 1, 1, 1, 1, 1$ forms the succinct representation of $101010_2 = 42$, $ufo(42) = 45$, while $45 = 101101_2$ is succinctly represented by $1, 1, 2, 1, 1$.

Problem J. Railway

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

A railroad siding consists of two (dead-end) sidetracks 1 and 2. The siding is entered by track A, and left by track B (see figure below).



There are n cars on track A, numbered from 1 to n. They are arranged in such a way that they enter the siding in the order $a_1, a_2, \ldots, a_n$. The cars are to be transferred to the siding, so that they leave it by track B in the order $1, 2, \ldots, n$. Each car is to be transferred once from track A to one of the sidetracks 1 or 2, and later (possibly after some transfers of the remaining cars) once from that sidetrack to the track B. The sidetracks are long enough to store even the longest trains, so there is no need to worry about their capacity.

Input

The first line of the input holds one integer n ($1 \leq n \leq 100\,000$) that denotes the number of cars for transfer. The second line stores the numbers $a_1, a_2, \ldots, a_n$ that are a permutation of $1, 2, \ldots, n$ (i.e., each a_i belongs to $\{1, 2, \ldots, n\}$, and all these numbers are unique), separated by single spaces.

Output

The first line of the output should contain the word `TAK` (*yes* in Polish) if there is a way of transferring the cars so that they enter track B in the order $1, 2, \ldots, n$, or the word `NIE` (*no* in Polish) if it is impossible. If the answer is `TAK`, the second line should give, separated by single spaces, the numbers of sidetracks (1 or 2) to which successive cars $a_1, a_2, \ldots, a_n$ are moved in a correct transfer. If there are several ways of making the transfer, choose one arbitrarily.

Examples

Standard input	Standard output
4 1 3 4 2	TAK 1 1 2 1
4 2 3 4 1	NIE

Explanation of the example: In the first example we start by moving car 1 to the first sidetrack, and move it to track B instantly. Then the car 3 is moved to the first sidetrack, and car 4 to the other one. Finally, the car 2 is moved to the first sidetrack, after which the cars 2 and 3 leave it to enter track B, followed by the car 4, which enters track B from the second sidetrack.

Problem K. Shift

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Byteasar bought his son Bytie a set of blocks numbered from 1 to n and arranged them in a row in a certain order. Bytie's goal is to rearrange the blocks so that they are ordered naturally, from the smallest number to the largest. However, the only moves Bytie is allowed to make are:

- putting the last block at the very beginning (move `a`), and
- putting the third block at the very beginning (move `b`).

Help Bytie by writing a program that tells whether a given arrangement of blocks can be properly reordered, and tells the right sequence of moves if it is.

Input

In the first line of the input there is a single integer n, $1 \leq n \leq 2\,000$. In the second line there are n integers from the range 1 to n, separated by single spaces. No number appears twice, and thus they represent the initial arrangement of the blocks.

Output

If there is no sequence of moves leading to an arrangement with increasing blocks' numbers, your program should print out "`NIE DA SIE`" (*there is no way* in Polish), without the quotation marks.

Otherwise there should be a single integer m ($m \leq n^2$), denoting the number of *operations*, in the first line. An *operation* is a k-fold execution of either `a` or `b` move.

If $m > 0$, then there should be a sequence of m integers with either `a` or `b` appended in the second line. Thus k`a` (for $0 < k < n$) denotes the k-fold execution of the move `a`. Analogously, k`b` (for $0 < k < n$) denotes the k-fold execution of the move `b`.

Furthermore, the characters appended to the numbers in the second line have to alternate.

Should there be more than one solution, your program is free to pick one arbitrarily.

Examples

Standard input	Standard output
4 1 3 2 4	4 3a 2b 2a 2b
7 1 3 2 4 5 6 7	NIE DA SIE
3 1 2 3	0