Problem A. Random Points on the Circle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.8 seconds
Memory limit:	512 mebibytes

Consider a circle of length L and pick a point on the circle which we will call the origin. In this problem, the coordinate of a point on the circle is the length of the counter-clockwise arc from the origin to this point. So, each point on the circle has a coordinate from 0 (inclusive) to L (exclusive). The distance between two points with coordinates a and b on the circle is the length of the smallest arc between them, that is, $\min(|a-b|, L-|a-b|)$.

You are given n ($1 \le n \le 1\,000\,000$) houses with integer coordinates on this circle. The coordinates of the houses are generated by a special pseudorandom generator. The code of the generator is given below. Note that there may be multiple houses at the same position.

You have to choose k ($1 \le k \le n$) points with integer coordinates on the circle and place collectors at these points. Again, there may be multiple collectors and/or houses at the same position.

After that, you have to assign a collector to each of the given n houses. Finally, for each collector, calculate the sum of all distances to the houses assigned to this collector. Your task is to place collectors and assign houses to them so that the maximum of these sums is as small as possible. Calculate and print this value.

Input

In this problem, $L = 2^{30}$.

The first line contains two integers n and k ($1 \le k \le n \le 1\,000\,000$). There is also a special condition: if $n \ge 100$, then $10 \le k \le \frac{n}{10}$ and $n \bmod k = 0$.

The second line contains two integers *seed* and *add* ($1 \le seed, add < L$). In all tests, these numbers are chosen uniformly at random. If we denote coordinate of the i-th point as a_i, the coordinates can be calculated using the following pseudocode:

```
1. for (i = 0; i < n; i++) {
2.     seed = (seed * 239017 + add) mod L;
3.     a_i = seed;
4. }
```

Output

Output a single integer: the smallest possible maximum over all collectors of the sum of all distances from this collector to the houses assigned to it.

Examples

standard input	standard output
10 2 13 123	626098570
10 3 13 123	302532222
10 10 13 123	0

Note

The generator produces the following points: `3107344, 752440587, 778714046, 266135273, 241409356, 201905063, 489905338, 937040197, 1024665608, 579507651`.

Be careful with your implementation. The time limit is quite tight.

Problem B. Lines

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	512 mebibytes

You are given n distinct points $a_1, \ldots, a_n$ on the plane. For each pair (i, j) $(i < j)$, consider the line passing through the points a_i and a_j (denote it as $L_{i,j}$). Let $A_{i,j}$ be the angle in radians from horizontal line to $L_{i,j}$ in counterclockwise direction. Note that, by definition, $0 \le A_{i,j} < \pi$.

Consider the array $p_1, p_2, \ldots, p_{\frac{n \cdot (n-1)}{2}}$ which contains the values $A_{i,j}$ in non-decreasing order. Your task is to find the median of p.

Recall that the median of the array of length x is its element with number $\lfloor \frac{x}{2} \rfloor + 1$ if x is odd, and the average of its elements with numbers $\lfloor \frac{x}{2} \rfloor$ and $\lfloor \frac{x}{2} \rfloor + 1$ otherwise.

Input

The first line of input contains a single integer n $(2 \le n \le 10^5)$, the number of points.

Next n lines contain the coordinates of the points: i-th of these lines contains two integers x_i and y_i $(-10^9 \le x_i, y_i \le 10^9)$, the coordinates of point a_i.

It is guaranteed that the points are distinct.

Output

Print the median of the angles with absolute or relative error at most 10^{-9}.

Examples

standard input	standard output
3 0 0 0 1 1 0	1.5707963268949
4 0 0 0 1 1 0 1 1	1.17809724517117
3 0 0 0 1000000000 1 0	1.5707963267949
3 0 0 1 0 2 0	0

Problem C. Fraction Factory

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3.5 seconds
Memory limit:	512 mebibytes

The famous Berland Fraction Factory works with different fractions: it can multiply them, break them into pieces, convert them between number systems.

But today everyone forgets about work, because today is a special day: the official supplier brought an incredibly large fraction to the factory. Here it is:

$$Q = \frac{a_1 \cdot a_2 \cdot \ldots \cdot a_n}{b_1 \cdot b_2 \cdot \ldots \cdot b_m}$$

Employees became interested in Q very quickly. So much that they even decided to calculate its value! Unfortunately for them, they did not have enough computing power. Fortunately for them, today you came to an interview at the Berland Fraction Factory. So now it is your task, and after solving it, you are guaranteed to get a job.

For better accuracy, the employees ask you to calculate the value of Q just modulo M, but repeatedly. More precisely, for $1 \le l \le k$, you have to output Q modulo M_l, where M_l is an integer greater than 1.

Now, let us formalize the process of calculation $Q \bmod M$. First, we factorize (decompose into prime factors) all factors in numerator and denominator of Q. Then we "reduce" all repeating primes: while there exists a prime number p and two positive integers A and B such that $Q = \frac{p \cdot A}{p \cdot B}$, we divide both numerator and denominator by p. After all that, we "fairly" calculate the value of $Q \bmod M$. As usual, we assume that $\frac{1}{x} \bmod M$ equals to such y that $0 \le y < M$ and $x \cdot y \equiv 1$ modulo M. If no such y exists, we say that x is non-invertible.

If during calculations modulo M, you have to invert a non-invertible value, just output "`DIVISION BY ZERO`".

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 5000$): the total number of factors in the numerator and denominator of the fraction, respectively.

The second line contains n integers a_i ($1 \le a_i \le 10^{18}$).

The third line contains m integers b_j ($1 \le b_j \le 10^{18}$).

The next line contains one integer k ($1 \le k \le 50$): the total number of queries.

Each of the following k lines contains an integer M_l ($2 \le M_l \le 10^{18}$).

Output

Print k lines. The l-th line must contain one integer ($Q \bmod M_l$) if it is correctly defined for that l, or the string "`DIVISION BY ZERO`" without quotes otherwise.

Example

standard input	standard output
3 2 8 11 14 9 12 4 8 9 10 11	4 DIVISION BY ZERO 4 0

Problem D. Greedy Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

There are n items and two players. For each player and for each item, the value of the item for this player is known. Denote values of the i-th item for the first and the second player as a_i and b_i correspondingly.

Players take the items in turns. The first player starts the game. The first player is greedy: each turn, he chooses the item which has the maximal a_i among the remaining items. If there are several such items, he can take any one of them. What is the maximal possible sum of values b_i of items taken by the second player that he can guarantee regardless of the first player's moves?

Input

The first line contains a single integer $1 \le n \le 10^5$, the number of items.

The second line contains n numbers, i-th is equal to a_i, the value of the i-th item for the first player.

The third line contains n numbers, i-th is equal to b_i, the value of the i-th item for the second player.

All values are integers from 1 to 10^9.

Output

Output a single number: the maximal sum of values b_i of items taken by the second player that he can guarantee.

Example

standard input	standard output
5 1 2 3 4 5 2 3 4 5 6	8

Problem E. Odd Grammar

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

A *formal grammar* is a way of describing formal languages as $\Gamma = \langle \Sigma, N, S \in N, P \subset N^+ \times (\Sigma \cup N)^* \rangle$ where Σ is called an *alphabet* and its elements are called *terminals*, N is a set of *nonterminals*, S is the starting nonterminal, and P is a set of *production rules* of the form $\alpha \to \beta$.

Here, N^+ contains all strings of one or more elements of N (non-empty strings of nonterminals), and $(\Sigma \cup N)^*$ consists of all strings of zero, one or more elements of $(\Sigma \cup N)$ (strings of terminals and nonterminals, including the empty string).

A grammar is called *context-free* if the left side of each production rule consists of exactly one nonterminal, more formally, $P \subset N \times (\Sigma \cup N)^*$.

For example, let us consider a grammar from the second example test case with alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, set of nonterminals $N = \{S, A\}$ and two production rules:

1. $S \to \mathtt{b}A$
2. $A \to \mathtt{aa}$

One can easily see that it is a context-free grammar.

To create the language generated by a grammar, one needs to start from a string consisting of only start nonterminal S, and then apply production rules one or more times. Applying a production rule is the procedure of finding the left side of that rule somewhere in the current string and replacing it by the string from the right side of that rule. The *language* generated by Γ is the set of all strings consisting **only** of terminals that can be produced by applying production rules one or more times.

For example, there is a string `baa` in the language generated by the grammar described above. To produce it, one could apply productions $S \to \mathtt{b}A \to \mathtt{baa}$. There are no other strings in the language generated by this grammar.

Some grammars may even generate infinite languages, others may generate empty ones.

You are given a context-free grammar with an alphabet consisting of two terminals "`a`" and "`b`". Your task is to check whether the language generated by this grammar contains a string of **odd** length.

Nonterminals in this task are enumerated from 1 to n. The starting nonterminal always has number 1.

Input

The input consists of one or more test cases.

The first line of each test case contains two integers n and m: the number of nonterminals and the number of production rules ($1 \le n \le 50000$, $1 \le m \le 200\,000$).

Each of the next m lines describes one production rule in the following manner. At first, A_i and k_i are given: the number of left side nonterminal ($1 \le A_i \le n$) and the number of characters on the right side of the production rule ($0 \le k_i \le 5000$). Then k_i objects follow, each of them is either a nonterminal $B_{i,j}$ ($1 \le B_{i,j} \le n$) or a terminal "`a`" or "`b`". Consecutive characters are separated by single spaces.

The total sum of all n over all test cases does not exceed 50 000. The total sum of all m over all test cases does not exceed 200 000. The size of the input does not exceed 5 megabytes.

The input is terminated by a string of two zeroes.

Output

For each test case, output a separate line. It must contain "`YES`" if the language generated by the given grammar contains a string of **odd** length, otherwise the line must contain "`NO`".

Example

standard input	standard output
2 2 1 2 a 2 2 1 b 2 2 1 2 b 2 2 2 a a 2 2 1 2 b 2 2 3 a a 1 0 0	NO YES NO

Problem F. Colored Path

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	256 mebibytes

You have a board of size $n \times n$. Each cell of the board has weight and color. Both weight and color are positive integers. Rows and columns are enumerated from 1 to n. Let (i, j) be j-th cell of i-th row. In one step you can move from cell (i, j) to cells $(i, j + 1)$ and $(i + 1, j)$.

Consider all paths from $(1, 1)$ to (n, n) that obey the rule above. Obviously each such path contains exactly $2n - 1$ cells. Let's define the weight of the path as the sum of the weights of the cells on the path. Let's define the colorness of the path as the number of different colors among the colors of the cells on the path.

Given the weights and the colors of all cells, find the smallest colorness among all paths with weight no more than W or report that there are no such paths.

Input

The first line contains three integers: n ($1 \le n \le 400$), k ($1 \le k \le 10$) which is the number of possible colors, and W ($1 \le W \le 10^9$). Each of the next n lines contains n integers, j-th integer on i-th line is the weight of the cell (i, j) ($1 \le w_{ij} \le 10^6$). The last n lines contain n integers each, j-th integer on i-th line is the color of the cell (i, j) ($1 \le c_{ij} \le k$).

Output

On the first line output the minimal colorness of the path. On the second line output the path with the minimal colorness in the format $i_1\ j_1\ i_2\ j_2\ \ldots\ i_{2n-1}\ j_{2n-1}$. If there are several paths with minimal colorness, output any of them. If there are no paths with weight no more than W, output -1 on a single line.

Examples

standard input	standard output
3 3 10 1 1 1 5 3 1 5 3 1 1 2 3 2 2 1 3 3 2	2 1 1 1 2 2 2 2 3 3 3
1 1 1 2 1	-1
2 6 1000 10 10 1 10 1 1 2 1	1 1 1 1 2 2 2

Note

In the first sample the weight of the path $(1, 1) - (1, 2) - (2, 2) - (2, 3) - (3, 3)$ is $1+1+3+1+1 = 7 \le W$ and its colorness is 2. There are obviously no paths with colorness 1.

In the second sample the only path has weight $2 > W$, so the answer is -1.

Problem G. Pencil of Wishing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.9 seconds
Memory limit:	512 mebibytes

Andrew likes playing a role-playing game "WebDDoS". At the moment he's got a valuable artifact called "Pencil of Wishing". With its help one can obtain any game item writing a special spell.

A spell is a string that consists of lowercase English letters and characters `?` and `*`. Every game item has its unique codename that consists of lowercase English letters. An item *matches* a spell if it's possible to replace each `?` with a lowercase English letter and each `*` with several (maybe zero) lowercase English letters in such a way that the resulting string is the item's codename.

The artifact is so powerful that it gives its owner all items that match the written spell. Andrew needs item A but he absolutely doesn't want item B. In order to save magical energy he asks you to find a spell of minimal length such that item A matches it but B does not.

Input

The first line contains the item A's codename.

The second line contains the item B's codename.

Codenames are different and non-empty, they consist only of lowercase English letters and contain no more than 700 characters.

Output

Output a spell which satisfies Andrew's request. If there are several possible spells output any of them.

Examples

standard input	standard output
aabb ab	*bb
abaabaaabbbaabbb abaabbbaabaaabbb	*b?????
amuletofyendor amuletofshmendor	*y*

Problem H. Points

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

You are given n triples of non-negative integers a_i, b_i, c_i ($1 \le i \le n$) and a positive integer $k \le n$. Your task is to find the set of indices $1 \le i_1 < i_2 < \ldots < i_k \le n$ such that $\left(\sum_{j=1}^{k} a_{i_j}\right)^2 + \left(\sum_{j=1}^{k} b_{i_j}\right)^2 + \left(\sum_{j=1}^{k} c_{i_j}\right)^2$ is maximized. Print the maximum possible value of this sum.

Input

The first line of the input contains a positive integer T, the number of test cases.

The first line of each test case consists of two integers n and k ($1 \le k \le n \le 40$). Each of the next n lines contains one triple of non-negative integers a_i, b_i and c_i ($0 \le a_i, b_i, c_i \le 10^6$).

It is guaranteed that the sum of all values of n in the input does not exceed 200.

Output

For each test case, on the first line, write one integer: the answer to the problem. On the next line, print k integers $1 \le i_1 < i_2 < \ldots < i_k \le n$ representing the set of indices on which the answer is achieved.

Example

standard input	standard output
2 3 1 4 2 6 4 9 7 6 4 2 3 2 7 3 2 8 2 4 4 7 9	146 2 394 2 3

Problem I. Set Intersection

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given $(n + 1)$ sets. Sets consist of integer elements between 1 and $2n$. The sizes of all sets are exactly n. The number n is **even**.

Proposition: there are always two sets, intersection of which has at least $\frac{n}{2}$ elements.

The task: find such two sets.

Input

The first line contains an integer n ($1 \le n \le 6000$, n is **even**). The next $(n + 1)$ lines contain $\lceil \frac{2n}{6} \rceil$ characters each. Each line contains encoded sequence of $2n$ zeroes and ones. There is a 1 on j-th position of i-th sequence if i-th set contains element j, or 0 otherwise. Thus, there are exactly n ones in each sequence.

Let us describe the encoding process. Consider a sequence $a_0, a_1, a_2, \ldots, a_{2n-1}$ of zeroes and ones. Let us append some zeroes to the end of the sequence to make its length divisible by 6. Now let us create a new sequence: $b_0 = \sum_{j=0}^{5} a_j \cdot 2^j$, $b_1 = \sum_{j=0}^{5} a_{j+6} \cdot 2^j$, $b_2 = \sum_{j=0}^{5} a_{j+12} \cdot 2^j$, $\ldots$

The characters with ASCII codes $33 + b_0, 33 + b_1, 33 + b_2, \ldots$ form the encoded sequence.

Output

Sets are enumerated from 1 to $(n + 1)$ in the order they are given in the input. Output two different integers: the numbers of sets, intersection of which has at least $\frac{n}{2}$ elements. If there are several possible answers, output any one of them.

Example

standard input	standard output
4 7" *$ D# M" ;"	2 3

Note

Decoded sequences:

1. 01101010
2. 10010011
3. 11000101
4. 00110110
5. 01011010

Problem J. Sort It!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

You are given a permutation of length n: p_1, p_2, ..., p_n. Consider some array of length n consisting of integers between 1 and n (equal elements are allowed). Let us transform the array in the following manner: at first, let us take all elements equal to p_1 from it and write them on a piece of paper (if there are no such elements, just do not write anything). Then write all elements equal to p_2, then equal to p_3 and so on, finishing by all elements equal to p_n, thus obtaining a new array of length n. For example, if the permutation is 2 1 3 and the array is 2 3 2, the resulting array will be 2 2 3. If after this transformation we get a sorted array, let us call the original array *sortable by* p. Calculate the total number of arrays that are sortable by p.

As the answer can be very large, output it modulo $10^9 + 7$.

Input

The first line contains a single integer n ($1 \le n \le 2000$): the length of the permutation. The second line contains n distinct integers p_1, p_2, ..., p_n ($1 \le p_i \le n$): the permutation itself.

Output

Output a single integer: the answer to the problem modulo $10^9 + 7$.

Examples

standard input	standard output
2 2 1	2
3 2 1 3	15

Problem K. Two Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	768 mebibytes

You are given two strings $S = S_0S_1 \cdots S_{|S|-1}$ and $T = T_0T_1 \cdots T_{|T|-1}$ consisting of lowercase letters. Here $|S|$ is the length of the string S.

The substring $S[l, r]$ $(0 \le l \le r < |S|)$ of the string $S = S_0S_1 \cdots S_{|S|-1}$ is the string $S_lS_{l+1} \cdots S_r$.

Define the function $F(S, l, r)$ for the string S and two integers l, r as follows:

$$F(S, l, r) = r - l - \max(l, |S| - r - 1) + 1.$$

In other words, F is the length of the substring minus the maximum distance from borders of S to the substring.

Your task is to find a substring $S[l, r]$ such that it occurs in T as substring and the value $F(S, l, r)$ is maximum among all pairs (l, r) $(0 \le l \le r < |S|)$.

Input

The first two lines contain strings S and T, respectively $(1 \le |S|, |T| \le 10^6)$.

Strings S and T consist of lowercase English letters.

Output

If no substring of string S occurs in the string T, print a single string "-1 -1" (without quotes). Otherwise, print two integers l and r such that $F(S, l, r)$ is maximum among all possible pairs (l, r) $(0 \le l \le r < |S|)$ and $S[l, r]$ is a substring of T. If there are several possible pairs, print the lexicographically smallest one.

Examples

standard input	standard output
riveragesmalir toaxernaturaln	4 5
aaaaa aaaaa	0 4
amkar zenit	-1 -1

Note

Pair (l_1, r_1) is lexicographically less than pair (l_2, r_2) if either $l_1 < l_2$, or $l_1 = l_2$ and $r_1 < r_2$.

Problem L. Triangle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are given n segments of different integer lengths from 1 to c. Construct a nondegenerate triangle using three of them. Among all the triangles that can be constructed, choose one with the minimum area.

Input

The first line of input contains one integer T, the number of test cases.

Each test case consists of two lines.

The first line of each test case contains two integers n and c: the number of segments and the maximum possible length of the segment ($1 \le n \le 50\,000$, $n \le c \le 100\,000$).

The next line contains n different integers a_i, the lengths of the segments ($1 \le a_i \le c$).

The total sum of all n over all test cases does not exceed 50 000.

The total sum of all c over all test cases does not exceed 100 000.

Output

For each test case, print the minimum possible area of the triangle on a single line. If no nondegenerate triangles could be constructed, print -1. Your answer must have an absolute error no more than 10^{-9}.

Example

standard input	standard output
4 3 3 1 2 3 4 4 1 2 3 4 3 11 5 7 11 6 11 5 7 8 9 10 11	-1 2.904737509655563 12.968712349342937 12.968712349342937