

Problem A. Small Numbers

Input file: `standard input`
Output file: `standard output`
Time limit: 2 seconds
Memory limit: 256 megabytes

Little Vlad has two favorite numbers a and b . Recently, he was taught in school the division and multiplication operations and he immediately started dividing and multiplying his favorite numbers.

He wrote a and b in his notebook and came up with three cool operations which he wants to perform on his numbers:

- Divide both numbers by one of their common divisors;
- Divide a by one of his divisor g , multiply b by g ;
- Divide b by one of his divisor g , multiply a by g .

After performing every operation he erases his old numbers, replacing them with new ones. He may choose to continue performing operations or to stop.

Since Vlad is small, he wants numbers to be smaller. Thus, he is trying to minimize sum of a and b , but can't handle it on his own. Help him to determine minimum sum that he can obtain by performing these operations and give an example of final a and b with such sum.

Input

Input data contains multiple test cases. The first line of input contains integer t — the number of test cases ($1 \leq t \leq 500$).

Each test case is described by a single line containing integers a and b ($1 \leq a, b \leq 10^9$) — Vlad's favorite numbers.

Output

For each test case output one line — the pair with the minimum sum that can be obtained by performing operations from the list.

Example

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

Problem B. New Keyboard

Input file: **standard input**
Output file: **standard output**
Time limit: 2 seconds
Memory limit: 256 megabytes

Petya has bought a new keyboard. It supports n layouts, each layout allows to input some subset of characters of the English alphabet. Layouts are enumerated from 1 to n .

Petya wants to type some message now, it consists of m letters of the English alphabet. Initially the first layout is active. Petya can do one of the following two actions:

- Switch layout to the next one. If the current layout is i , the new layout will have number $i \bmod n + 1$, where $\bmod$ denotes the remainder of the division. If the previous action was also layout switch, this action takes b milliseconds, in the other case it takes a milliseconds.
- Type a character. Petya can add the typed character to the end of the current message. He spends c milliseconds for this action.

Help Petya to find out what minimal time he needs to type the message, or detect that it is impossible to type the message using his new keyboard. The final layout of the keyboard doesn't matter.

Input

The first line of the input data contains four integers n , a , b and c — the number of layouts, the number of milliseconds to switch the layout after a switch, to switch the layout in the other case, and to type a character, respectively ($1 \leq n \leq 2000$, $1 \leq b \leq a \leq 10^9$, $1 \leq c \leq 10^9$).

The following n lines describe the layouts. Each layout is described by a string that contains all characters of the English alphabet that can be typed using this layout. Each character is specified at most once for each layout. Characters in each string are ordered alphabetically.

The last line contains a string s — the message that the Petya wants to type (length of s is from 1 to 2000, inclusive). The string s consists of English letters.

Output

Output one integer — the minimum number of milliseconds needed to type the message. Output -1 if it is impossible to type the message.

Examples

standard input	standard output
5 3 2 1 abc d e f def abcdef	15
1 1 1 1 a z	-1

Problem C. Folding the Figure

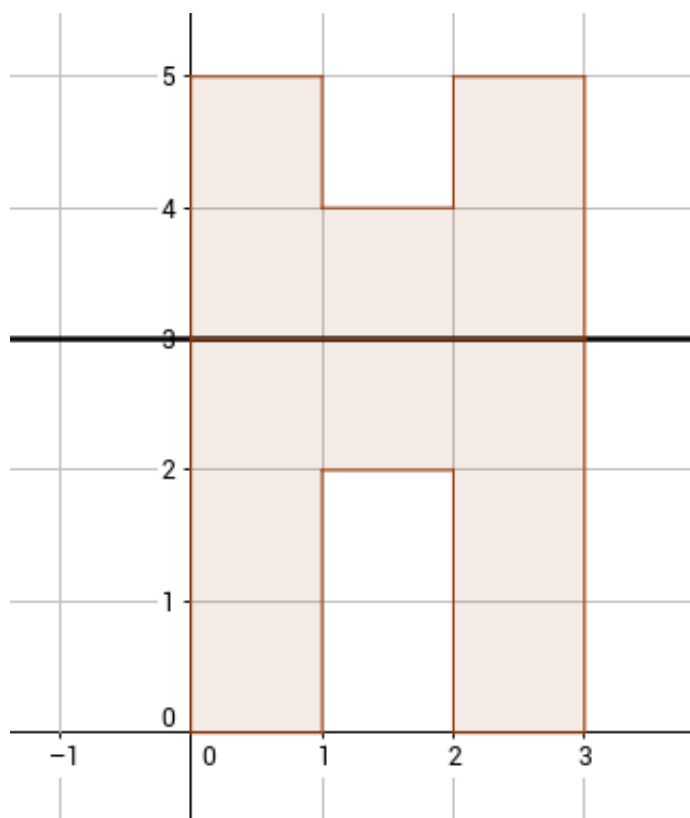
Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Petya was getting bored at his Math lesson, so he started painting squares at his grid paper. When he got bored of that action as well, he noticed that he now has a connected set of k painted squares — it is possible to get from any painted square to any other painted one by moving between painted squares sharing a common side.

He cut the figure away from the sheet of paper and folded it along some grid line (horizontal or vertical, he didn't remember). After that he painted squares of some other grid paper to create the copy of the folded figure. Now Petya has lost his original figure, so all he has is the copy of it after folding. Now Petya wants to restore the figure.

It is difficult to restore the figure exactly, but Petya has decided that he would like to have at least some figure of k squares that can be folded to get the same folded image. Help him, find some connected figure of k squares that can be folded the required way.

Consider for example the second sample test. It has a folded figure that is similar to an upside-down “U” letter, the original figure contains 12 squares. One way the original figure could look like is shown at the picture, it was folded along a line $y = 3$:



Input

Input data contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 200$).

Each of the t test cases is described in the following way: the first line of the description contains two integers n, k — the number of painted squares in the folded figure and the number of painted squares in the original figure ($1 \leq n < k \leq 10^5$).

Each of the following n lines contains two integers x_i, y_i — coordinates of the i -th painted square's bottom left corner ($-10^8 \leq x_i, y_i \leq 10^8$). It is guaranteed that all painted cells are distinct and form a connected

figure.

It is guaranteed that the sum of values of k in all test cases of one input data doesn't exceed 10^5 .

Output

For each test case output the answer to this test. Output the description of the figure and the way to fold it to get the input figure.

The first line must contain the description of the folding line. The following k lines must contain two integers (x'_i, y'_i) each — the coordinates of the squares of the connected figure that can be folded along the given line to get the figure from the input data.

The folding line description must be one of the following 4:

- L num — fold along the line $x = num$, put left side atop the right one;
- R num — fold along the line $x = num$, put the right side atop the left one;
- U num — fold along the line $y = num$, put the top side atop the bottom one;
- D num — fold along the line $y = num$, put the bottom side atop the top one.

All x'_i , y'_i , and the coordinate of the folding line must not exceed 10^9 by their absolute values. It is guaranteed that the required figure exists. If there are several possible answers, output any of them.

Example

standard input	standard output
2	L 0
7 14	0 0
0 0	0 1
0 1	0 2
0 2	1 2
1 2	2 0
2 2	2 1
2 1	2 2
2 0	-1 0
7 12	-1 1
0 0	-1 2
0 1	-2 2
0 2	-3 2
1 2	-3 1
2 2	-3 0
2 1	U 3
2 0	0 0
	0 1
	0 2
	1 2
	2 2
	2 1
	2 0
	0 3
	1 3
	2 3
	0 4
	2 4

Problem D. Acute Triangles

Input file: **standard input**
Output file: **standard output**
Time limit: 4 seconds
Memory limit: 256 megabytes

Recently Moscow high school student Dmitri Zakharov set a new record on the number of points in d -dimensional space, such that all triangles with vertices in those points are acute.

Tanya wants to compete with Dmitri. Of course she is planning to use the computer. To make a good start, she decided to solve the following problem. Given n points on a plane, find the number of acute triangles that have vertices in those points. The triangle is acute if all of its angles are less than 90 degrees.

Input

Input data contains multiple test cases. The first line of input contains integer t — the number of test cases ($1 \leq t \leq 666$).

Each test case is described by a line containing integer n ($3 \leq n \leq 2000$) — the number of points.

The following n lines contain two integers x_i, y_i each ($-10^9 \leq x, y \leq 10^9$) — coordinates of the points. It is guaranteed that in one test case all points are distinct.

The total number of points in all test cases of one input data doesn't exceed 2000.

Output

For each test case output one line — the number of acute triangles with vertices in the given points.

Example

standard input	standard output
2	3
5	4
1 1	
2 2	
3 3	
4 1	
6 4	
5	
0 0	
3 1	
5 1	
5 -1	
1 3	

Problem E. Joining Arrays

Input file: **standard input**
Output file: **standard output**
Time limit: 4 seconds
Memory limit: 256 megabytes

Consider two arrays of integers $A = [a_1, a_2, \dots, a_n]$ and $B = [b_1, b_2, \dots, b_m]$. Let us introduce their k -join as a lexicographically smallest array R of length k , that can be divided to two non-empty subsequences, first of which is a subsequence of A , and the second one is a subsequence of B .

Formally speaking, you need to find such array $R = [r_1, r_2, \dots, r_k]$, that there are two non-empty sequences of indices $1 \leq i_{1,1} < i_{1,2} < \dots < i_{1,t} \leq n$ and $1 < j_{1,1} < j_{1,2} < \dots < j_{1,k-t} \leq m$ in the original arrays, and two sequences of indices $1 \leq i_{2,1} < i_{2,2} < \dots < i_{2,t} \leq k$ and $1 \leq j_{2,1} < j_{2,2} < \dots < j_{2,k-t} \leq k$, such that:

- For each $1 \leq p \leq t$, $1 \leq q \leq k - t$ we have $i_{2,p} \neq j_{2,q}$;
- For each $1 \leq p \leq t$ we have $a_{i_{1,p}} = r_{i_{1,p}}$;
- For each $1 \leq p \leq k - t$ we have $b_{j_{1,p}} = r_{j_{1,p}}$.

For example if $A = [1, 2, 1, 3, 1, 2, 1]$, $B = [1, 2, 3, 1]$, and $k = 9$, their k -join is the array $R = [1, 1, 1, 1, 2, 1, 2, 3, 1]$ (subsequence from the first array — $[1, 1, 1, 2, 1]$, subsequence from the second array — $[1, 2, 3, 1]$).

For two given arrays A and B , and an integer k find their k -join R .

Input

The first line of input contains n — length of array A ($1 \leq n \leq 3000$).

The second line contains n integers a_i — array A ($1 \leq a_i \leq 3000$).

The third line contains m — length of array B ($1 \leq m \leq 3000$).

The fourth line contains m integers b_i — array B ($1 \leq b_i \leq 3000$).

The last line contains integer k ($2 \leq k \leq n + m$).

Output

Output k -join of the given arrays.

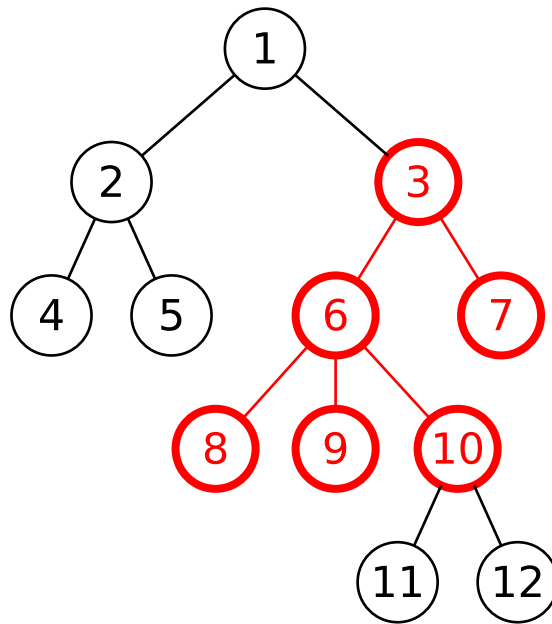
Example

standard input	standard output
7 1 2 1 3 1 2 1 4 1 2 3 1 9	1 1 1 1 2 1 2 3 1

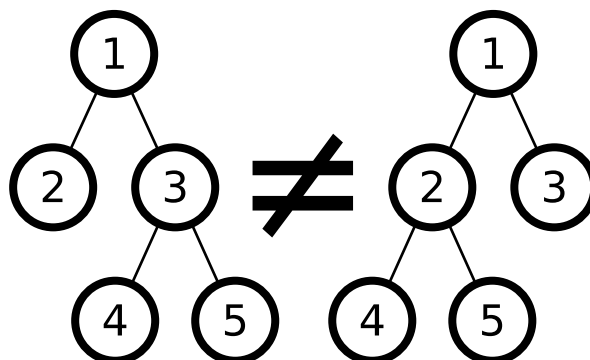
Problem F. Two Trees

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 256 megabytes

Consider a rooted tree. Let us a vertex v that has at least one vertex in its subtree that has distance k from v . Let us call a tree that we get from a subtree rooted at v by removing all vertices that are at distance greater than k from v its k -subtree rooted at v . For example, the picture below shows 2-subtree of the vertex 3 in red.

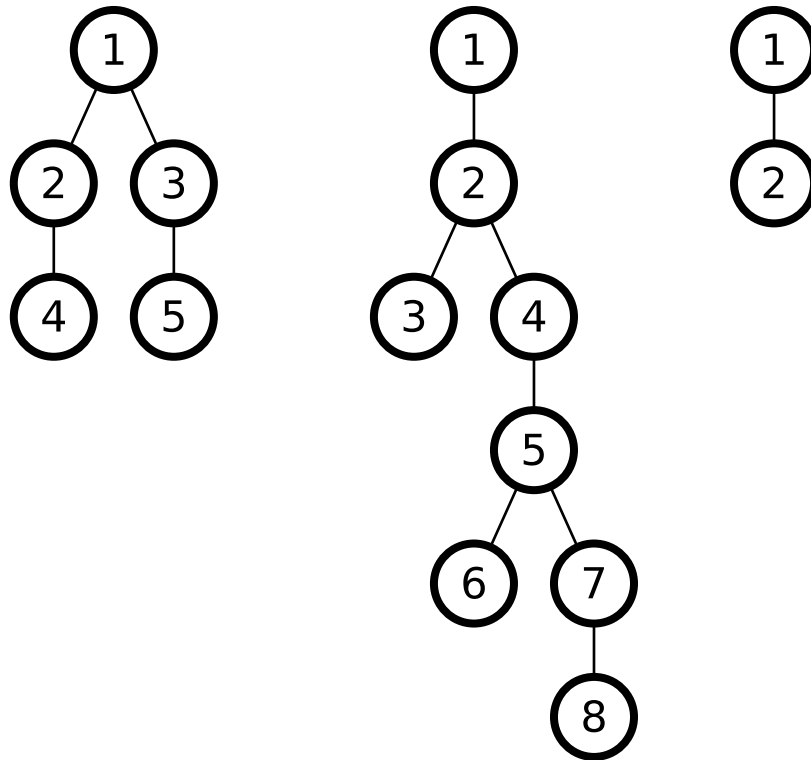


Let us call two subtrees equal, if it is possible to renumber vertices of one of them to get the other one. You are **not allowed** to change the order of children of a vertex. For example, the following two trees are **not equal**:



You are given a rooted tree. Find the greatest k such that the given tree has two equal k -subtrees with different roots. These trees are allowed to have vertices in common.

The picture shows trees from sample test.



The first example has equal 1-subtrees rooted at 2 and 3.

The second example has equal 3-subtrees rooted at 1 and 4.

The third example has equal 0-subtrees rooted at 1 and 2.

Input

Input data contains multiple test cases. The first line contains t — the number of tests ($1 \leq t \leq 10^4$).

Each of the following t test is described in the following way: the first line contains one integer n — the number of vertices ($2 \leq n \leq 10^5$).

The following n lines describe vertices, the i -th of them first contains an integer cnt_i — the number of vertices of the vertex i , followed by cnt_i integers — numbers of the children of the i -th vertex, in order. Vertices are numbered starting from 1. The root of the tree is the vertex number 1. It is guaranteed that the graph is the correct tree rooted at 1.

The sum of values of n for all tests in one input data doesn't exceed $2 \cdot 10^5$.

Output

For each test case output maximal k such that there are two equal k -subtrees with different roots.

Example

standard input	standard output
3	1
5	3
2 2 3	0
1 4	
1 5	
0	
0	
8	
1 2	
2 3 4	
0	
1 5	
2 6 7	
0	
1 8	
0	
2	
1 2	
0	