Problem A. Alien Homophones

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Okarun is obsessed with the idea that aliens exist. In his childhood, he tried various ways to contact them, but to no avail. This is not surprising, as aliens speak a completely different language! One day, he stumbled upon an article stating that aliens actually use lowercase Latin letters for writing, but read words in a completely different way.

In the alien language, there is a set of $n + 26$ different sounds, each represented by a string of Latin letters s_i. It is known that for any $1 \le i \le 26$, the i-th sound is represented by a string of length one, consisting of the i-th letter of the Latin alphabet. For any $i > 26$, the sound with number i is represented by a string consisting of at least two Latin letters.

When an alien wants to read the word x, he starts reading it from the first position. When the alien is at position i, he looks for a sound s_j in the set of sounds such that it appears as a substring† in x, starting from position i. If there are multiple such sounds, he chooses the sound s_j with the **maximum** length. Then he pronounces that sound and moves to position $i + |s_j|$ and continues reading the word until he has read it completely. Note that we can always read any word because all Latin letters are sounds.

It also turned out that some alien sounds actually sound the same but are written differently. This means that there are some pairs of sound strings s_i and s_j $(i \ne j)$ such that $s_i \ne s_j$, but the sounds represented by these strings are the same. Note that if s_i and s_j are considered the same sound, and s_j and s_l are considered the same sound, then s_i and s_l are also considered the same sound.

Aliens call some words *homophones* — words that sound the same, and their spelling may either match or differ. In other words, if we have two words v and w, aliens call them homophones if the sequence of sounds pronounced by aliens while reading word v matches the sequence of sounds pronounced by aliens while reading word w.

Okarun wrote some text t, consisting of lowercase Latin letters. At one point, he became interested in examining q pairs of substrings of the text. In each such pair of substrings, the first substring is from the a_i-th to the b_i-th character in the text t, inclusive, and the second is from the c_i-th to the d_i-th characters in the text t, inclusive. For each pair of substrings, Okarun wants to know if these substrings are homophones when read as words in the alien language.

† A substring of a string s is a string that can be obtained by removing some number of characters (possibly zero) from the beginning of the string s and some number of characters (possibly zero) from the end of the string s.

Input

The first line contains a non-empty string t $(1 \le |t| \le 500\,000)$, consisting of lowercase Latin letters — the text written by Okarun.

The second line contains two integers n and k $(0 \le n \le 500\,000, 0 \le k < n + 26)$ — the number of sounds in the set that have a length of at least two and the number of pairs of identical sounds noted by Okarun.

The next n lines describe the sounds numbered from the 27th. The i-th of them contains a non-empty string s_{i+26} $(2 \le |s_{i+26}| \le 10^6)$, consisting of lowercase Latin letters — the representation of the $(i+26)$-th sound. It is guaranteed that all sound strings are different. Note that there are no strings from "a"to "z"in the input; however, in each test, they are considered sounds numbered from 1 to 26.

The next k lines describe pairs of identical sounds originally noted by Okarun. Each of them contains a pair of integers x_i and y_i $(1 \le x_i, y_i \le n + 26;\ x_i \ne y_i)$ — a pair of sound numbers that Okarun considers identical in sound. It is guaranteed that each pair of numbers appears no more than once. Note that the equality of some other pairs of sounds may follow from this set.

The next line contains a single integer q ($1 \le q \le 300\,000$) — the number of queries for pairs of substrings of the text, for which it is necessary to determine whether they are alien homophones.

In the next q lines, the i-th of them contains four integers a_i, b_i, c_i, d_i ($1 \le a_i \le b_i \le |t|$, $1 \le c_i \le d_i \le |t|$), which specify the substrings of the text — the first substring from position a_i to position b_i (inclusive) in the text t, the second from position c_i to position d_i (inclusive) in the text t.

Let S denote the sum of the lengths of the sounds $\sum |s_i|$ (excluding the sounds "a" to "z"). It is guaranteed that $S \le 10^6$.

Output

Output q lines. If the pair of strings from the i-th query are identical in sound, output "`Yes`" (without quotes) in the i-th line; otherwise, output "`No`" (without quotes) in the i-th line.

Example

standard input	standard output
abracadabra 2 3 cada ca 1 27 1 28 1 4 4 5 11 1 4 4 6 5 7 5 7 5 8 2 5 2 5	Yes Yes No Yes

Note

In the first query:

- The first substring **cadabra** is read as sounds: **cada**, **b**, **r**, **a**;
- The second substring **abra** is read as sounds: **a**, **b**, **r**, **a**.

The sounds **cada** and **a** are noted as identical (pair (1, 27)), therefore these two substrings are alien homophones.

In the second query:

- The first substring **aca** is read as sounds: **a**, **ca**;
- The second substring **cad** is read as sounds: **ca**, **d**.

The sounds **a** and **ca** are noted as identical (pair (1, 28)), the sounds **ca** and **d** are also identical (since the sounds with numbers (1, 4) and (1, 28) are identical, then the sounds with numbers 4 and 28 are also identical), therefore these two substrings are also alien homophones.

In the third query:

- The first substring **cad** is read as sounds: **ca**, **d**;
- The second substring **cada** is read as sounds: **cada**.

The number of sounds does not match, so these two substrings are definitely not alien homophones.

In the fourth query, two identical substrings are given, so they are read the same and are alien homophones.

Problem B. Tickets for the Train

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

The final round of the Olympiad is about to start. That is why teachers from other cities should start worrying about sending students to the venue of the Olympiad. There is a train that goes from your city to the city where the Olympiad is held. You are responsible for sending all students to the Olympiad in one day.

Due to historic reasons, train cars have different capacities. Also, the cars differ in terms of comfort as well, so the ticket prices for different cars may differ as well.

According to the result of the qualifying round, exactly n students from this city will go to the final stage and, of course, you want to buy the train tickets of the least possible total cost. At first glance, this problem looked quite simple, but then you remembered about the new law which requires students to be accompanied by teachers on such long trips. Moreover, the law formally regulates the number of teachers required: for every k students, there must be at least one teacher. Of course, one teacher can only look after the students traveling in the same car with him. Formally, if there are x students in some car, then this car must have at least $\lceil \frac{x}{k} \rceil$ teachers traveling with them. You can convince any number of teachers to take part in the trip, so there are no restrictions here. However, tickets for the teachers must be bought as well.

Send all the students and the necessary number of teachers by this train while not violating the law. In case it's impossible to send all the students and the necessary number of teachers, print "`-1`". Otherwise, print the minimum total cost of all the tickets.

Input

The first line contains three integers m, n, and k ($1 \le m \le 10^6$, $0 \le n \le 10^9$, $1 \le k \le 20$) — the number of cars in the train, the number of students to go to the final round, and the maximum number of students per one teacher.

Each of the following m lines contains two integers p_i and c_i ($1 \le p_i, c_i \le 10^9$) — the number of seats and the ticket fare in the i-th car.

Output

Print exactly one integer — the minimum total cost for all the tickets, or "`-1`", if it's impossible to send all the students.

Examples

standard input	standard output
5 12 3	70
10 7	
11 6	
6 2	
7 10	
8 5	
8 55 1	1536
70 22	
69 9	
99 44	
31 81	
57 50	
22 79	
99 33	
17 64	

Note

Please note that you don't need a ticket since you prefer to travel by bicycle.

In the first example, the optimal way to send students is as follows: send 4 students with 2 teachers in the third car, 6 students and 2 teachers in the fifth car, and 2 students with 1 teacher in the second car. This way, in every car there will be at least one teacher per three students, and the cost of the trip would be $6 \cdot 2 + 8 \cdot 5 + 3 \cdot 6 = 12 + 40 + 18 = 70$.

Problem C. Spanning Trees

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

You are given an undirected graph without loops and multiple edges. Each edge is either black or white.

There are two types of edges:

- Edges connecting v and $(v \bmod n+1)$, for $v \in \{1, 2, \ldots, n\}$.
- Edges connecting v and $n+1$, for $v \in \{1, 2, \ldots, n\}$.

For each k from 0 to n, find the number of spanning trees in the given graph that have exactly k white edges. As the answers may be very large, find the modulo 998 244 353.

Input

The first line contains one integer n ($3 \le n \le 50\,000$).

The second line contains one string s, consisting of n characters, describing edges of the first type. If s_i is equal to «`-`», there is no edge between i and $(i \bmod n+1)$, if it is equal to «`W`», this edge is white, and if it is equal to «`B`», it is black.

The second line contains one string t, consisting of n characters, describing edges of the second type. If t_i is equal to «`-`», there is no edge between $n+1$ and i, if it is equal to «`W`», this edge is white, and if it is equal to «`B`», it is black.

Output

Print $n+1$: integers $a_0, a_1, \ldots, a_n$: a_k should be the number of spanning trees of the given graph with exactly k white edges, modulo 998 244 353.

Examples

standard input	standard output

`3`
`---`
`WBW`
	`0 0 1 0`

`3`
`WWW`
`BBB`
	`1 6 9 0`

`5`
`BWB-B`
`WB-W-`
	`0 2 6 3 0 0`

`4`
`----`
`----`
	`0 0 0 0 0`

Problem D. Bounded Arithmetic Expression

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

You are given an arithmetic expression consisting of a sum of one or more terms. Each term is a product of exactly two variables, and in each term, the two variables are distinct. No two terms contain the same unordered pair of variables.

There are n variables in total. The i-th variable corresponds to the i-th lowercase letter of the English alphabet ('a', 'b', 'c', ...).

For each variable i:

- Its value x_i must be a real number between l_i and r_i (inclusive).
- The sum of all variables must not exceed a given integer s.

Your task is to maximize the value of the expression under these constraints.

The expression is given as a single line string where each term consists of two letters (representing two variables) and all terms are separated by the '+' sign. For example, `ba+cb` represents the expression:

$$b \cdot a + c \cdot b$$

Even if some variables do not appear in the expression, you must still assign values to all variables satisfying the given bounds and the total sum constraint.

It is guaranteed that a valid assignment of variables always exists.

Input

The first line contains a single non-empty string — the expression, it consists of lowercase letters and '`+`' characters.

The second line contains an integer n ($2 \le n \le 16$) — the number of variables.

The third line contains n integers l_i ($0 \le l_i \le 100$).

The fourth line contains n integers r_i ($l_i \le r_i \le 100$).

The fifth line contains an integer s ($\sum l_i \le s \le 1300$).

The expression contains one or more terms separated by '`+`', and the length of the expression doesn't exceed 2500.

Each term is exactly two distinct letters, chosen from the first n lowercase letters.

No unordered pair of letters appears more than once, and no term contains two identical letters.

Output

Output a single real number: the maximum value of the expression.

Your answer must have an absolute or relative error of at most 10^{-9}.

Examples

standard input	standard output
`ba+cb`	`2.250000000000`
`3`	
`0 0 1`	
`1 2 1`	
`3`	
`ab`	`1.000000000000`
`3`	
`0 0 10`	
`20 20 20`	
`12`	
`ca+fc+fa+db+da+dc+cb+ed+eb+ea`	`2029.250000000000`
`6`	
`10 11 12 13 14 15`	
`15 16 17 18 19 20`	
`85`	

Note

In the first example, the maximum value is obtained by setting $a = 0.5$, $b = 1.5$, $c = 1$.

In the second example, we have to set a proper value for c even though it is not present in the expression.

Problem E. Cactus

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

You are given a graph with n vertices connected by m bidirectional edges, forming a connected graph where each edge belongs to at most one simple cycle – a cactus. A simple cycle is a cycle that does not visit any vertex more than once.

Considering colorings of this graph with colors $\in \{1, 2, 3\}$ for each vertex, such that adjacent vertices have different colors.

Determine the minimum number of vertices with color 3 among all such colorings, or report that it is impossible to find a proper coloring of this graph in 3 colors.

Input

The first line contains two integers n and m — the number of vertices and edges ($1 \le n \le 100\,000$, $0 \le m \le 150\,000$).

Each of the next m lines contains two integers a_i, b_i, denoting two vertices connected by the i-th edge ($1 \le a_i, b_i \le n$; $a_i \ne b_i$). It is guaranteed that there are no loops and multiple edges, and that the given graph is a cactus (in other words, each edge belongs to at most one simple cycle).

Output

Print -1, if it is impossible to color the given graph in 3 colors, or the minimum number of vertices colored in the third color, otherwise.

Examples

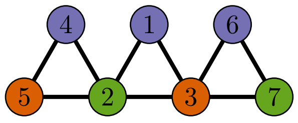
standard input	standard output

7 9
1 2
2 3
3 1
2 4
4 5
5 2
3 6
6 7
7 3
	2

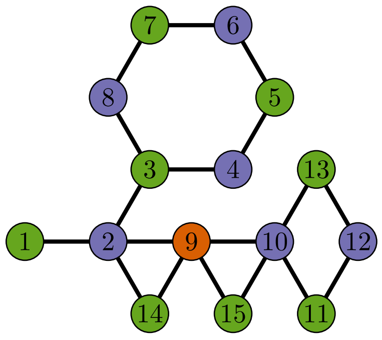
15 18
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 3
2 9
9 10
10 11
11 12
12 13
13 10
2 14
14 9
9 15
15 10
	1

Note

For the first example, one of the ways to color vertices is the following (vertices with color 3 are colored in red):



For the second example:



Problem F. Lazy Cuts

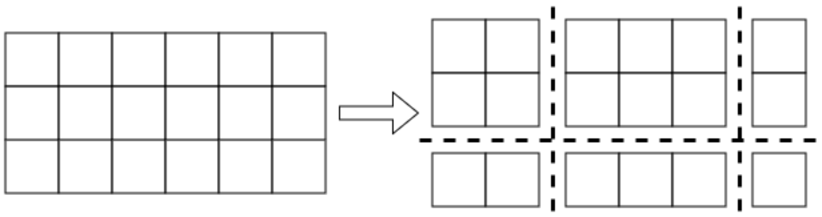
Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Anton needs to bring s squares to school for his craft lesson. Anton is very lazy, but he loves craft lessons, so he is willing to make the minimum necessary number of cuts.

At home, Anton found a sheet of grid paper sized $n \times m$, with $s \leq n \cdot m$. Anton also has a laser cutter that works as follows:

1. The original sheet of paper is placed in the cutter.
2. The cutter makes x straight vertical cuts along the edges of the squares.
3. The cutter makes y straight horizontal cuts along the edges of the squares.
4. The cutter produces $(x + 1) \cdot (y + 1)$ new sheets of paper.

For example, from a 3×6 sheet, three cuts can produce six sheets with areas 4, 2, 6, 2, 1, and 3.



Since Anton is lazy, he wants to make the minimum possible number of cuts so that from the sheets produced by the cutter, he can collect a set with a total area of exactly s squares to bring to school for his craft lesson.

Anton is afraid of overworking while using the cutter, so he asks you to help determine the minimum possible number of cuts he will have to make.

Input

The input consists of a single line containing three integers n, m, and s — the dimensions of the original sheet of grid paper and the required number of squares for the class ($1 \leq n, m \leq 10^5$, $1 \leq s \leq n \cdot m$).

Output

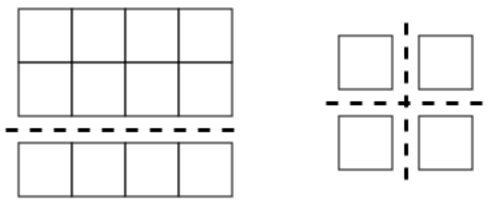
Output a single integer — the minimum possible number of cuts needed so that it is possible to collect a set with a total area of exactly s squares from the resulting sheets of paper.

Examples

standard input	standard output
`3 4 8`	`1`
`2 2 3`	`2`
`10 9 90`	`0`

Note

The image below shows examples of cuts for the first two tests from the example.



In the third test from the example, $s = n \cdot m$, so no cuts are needed.

Problem G. Checkered Pattern

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

You are given a grid of size $n \times m$, where each cell is colored either black or white.

A pattern is considered **beautiful** if it satisfies the following conditions:

- There is **at least one** black cell in the grid.
- If we treat each black cell as a vertex in a graph, and connect any two black cells that share a side (i.e., are adjacent vertically or horizontally) with an edge, then:
  - The resulting graph contains **exactly one connected component**;
  - The graph is **acyclic** (i.e., contains no cycles).

You are allowed to repaint any cell — changing it from black to white or white to black. Your goal is to create a beautiful pattern while minimizing the **number of repainted cells**.

Find **any** pattern that satisfies the conditions and requires the **fewest possible repaints**.

Input

The first line contains two integers n and m — the height and width of the grid ($1 \le n \le 100, 1 \le m \le 10$).

Each of the next n lines contains exactly m characters: dots (‘`.`‘) and hashes (‘`#`‘), representing the initial coloring of the grid.

A dot (‘`.`‘) indicates a white cell, while a hash (‘`#`‘) indicates a black cell.

Output

Print any $n \times m$ grid that satisfies the given constraints and requires the minimum possible number of cell repaints.

Examples

standard input	standard output

`3 3`
`###`
`#.#`
`###`
	`###`
	`#.#`
	`##.`

`4 3`
`##.`
`.##`
`###`
`##.`
	`##.`
	`.##`
	`#.#`
	`###`

`2 3`
`...`
`...`
	`...`
	`#..`

Problem H. Road Lighting

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

Berland is a country with a not very developed road system. There are a total of n cities in Berland and $n-1$ bidirectional roads, the i-th of which connects cities v_i and u_i. It is known that between every pair of cities, it is possible to reach one another using only these roads.

It is currently night in Berland, and some roads need to have their lighting turned on. There is a law, created for the purpose of saving electricity, that prohibits turning on the lighting simultaneously on two roads that share a common endpoint. The residents of Berland are interested in the maximum number of roads on which lighting can be turned on without violating the law, so they have turned to you for help.

Unfortunately, in Berland, some roads may be affected by a heavy snowstorm, which can disrupt the connectivity between some pairs of cities. Initially, no road is affected by a snowstorm. There are q queries of two types:

1. Change the weather on road e_i ($1 \le e_i \le n-1$): if there is currently no snowstorm on road e_i, a snowstorm begins, and vice versa.
2. It is required to turn on the lighting on the maximum number of roads such that both endpoints can be reached from city x_i, using only the roads that are not affected by a snowstorm. Of course, lighting can be turned on without violating the law, meaning there should not be a pair of roads with lighting turned on that exit from the same city. In other words, the original problem needs to be solved while considering only the cities reachable from x_i via roads that are not affected by a snowstorm.

Input

The first line contains a single integer n ($2 \le n \le 300\,000$) — the number of cities.

The next $n-1$ lines describe the roads. The i-th line contains two integers v_i and u_i ($1 \le v_i, u_i \le n$) — the numbers of the cities connected by the i-th road. It is guaranteed that between every pair of cities, it is possible to reach one another using only these roads.

The following line contains a single integer q ($1 \le q \le 300\,000$) — the number of queries.

In the next q lines, the queries are specified. The i-th line begins with an integer t_i ($1 \le t_i \le 2$).

- If $t_i = 1$, then this is a query of the first type, followed by a single integer e_i ($1 \le e_i \le n-1$) — the number of the road on which the weather changes.
- If $t_i = 2$, then this is a query of the second type, followed by a single integer x_i ($1 \le x_i \le n$). In this case, the original problem needs to be solved while considering only the cities reachable from x_i via roads that are not affected by a snowstorm.

Output

For each query of the second type, output the maximum number of roads connecting cities reachable from x_i on which lighting can be turned on without violating the law.

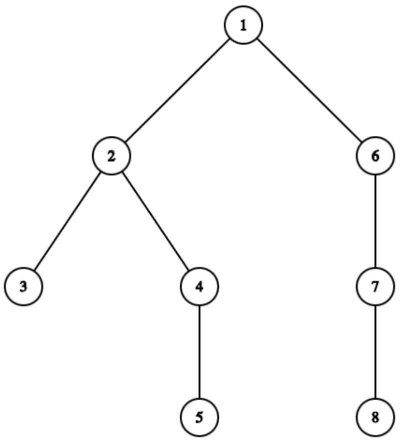
Example

standard input	standard output

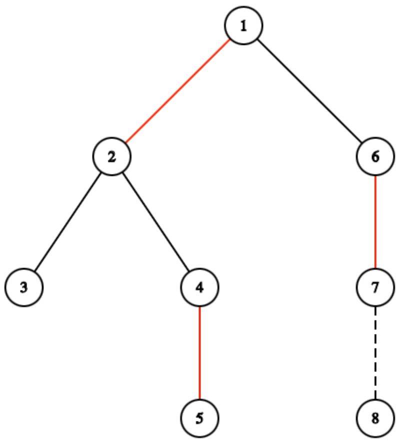
8
1 2
2 3
2 4
4 5
1 6
6 7
7 8
8
1 7
2 1
1 7
2 1
1 3
2 3
1 5
2 6
	3
	4
	3
	1

Note

In the test example, Berland initially has the following structure:

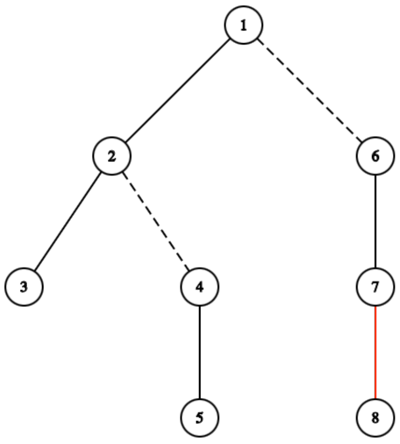


After the first query, a snowstorm begins on the road between cities 7 and 8. Then a query comes regarding city 1. In this case, we consider all cities except for city 8, as it is not reachable from city 1. Below is one of the optimal ways to turn on the lighting on the roads (the roads with lighting turned on are marked in red):



The next query indicates that the snowstorm on the road between cities 7 and 8 ends, meaning everything returns to its original state.

In the very last query, from vertex 6, only vertices 6, 7, and 8 are reachable, so lighting can be turned on at most on one road, for example:



Problem I. Golden Ring

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

In the Golden Ring of Russia, a historic trade route connects several ancient towns. Through this land flows the Volga River, which can be represented as an axis starting from some point (called the river source) in a direction given by an integer vector. It is known that none of the towns is located on the river (including the river source).

Given the coordinates of the Golden Ring towns, you need to connect them all with roads such that:

- each road is a straight-line segment connecting two towns;
- two roads cannot have common points (except when they share the same town);
- a road cannot cross the river, but it may touch the river source;
- it must be possible to start at some town, visit all towns exactly once, and return to the same town.

Input

The first line of the input contains one positive integer n, not exceeding 10000 — the number of towns.

The second line contains four integers X, Y, dX, dY — the coordinates of the river source and the direction vector of the river.

Each of the next n lines describes one town and contains two integers X_i and Y_i. All integers in the input do not exceed 10^4 by absolute value.

You may assume that there exists at least one pair of towns that can be connected directly by a road, and at least one pair of towns that cannot be connected directly because the road segment would intersect the river.

Output

Print n integers — the numbers of towns in the order they should be connected in a cyclic path (each town connected to its two neighbors in the list, with the first town connected to the last one). If it is impossible to build the roads as described, print -1.

Example

standard input	standard output

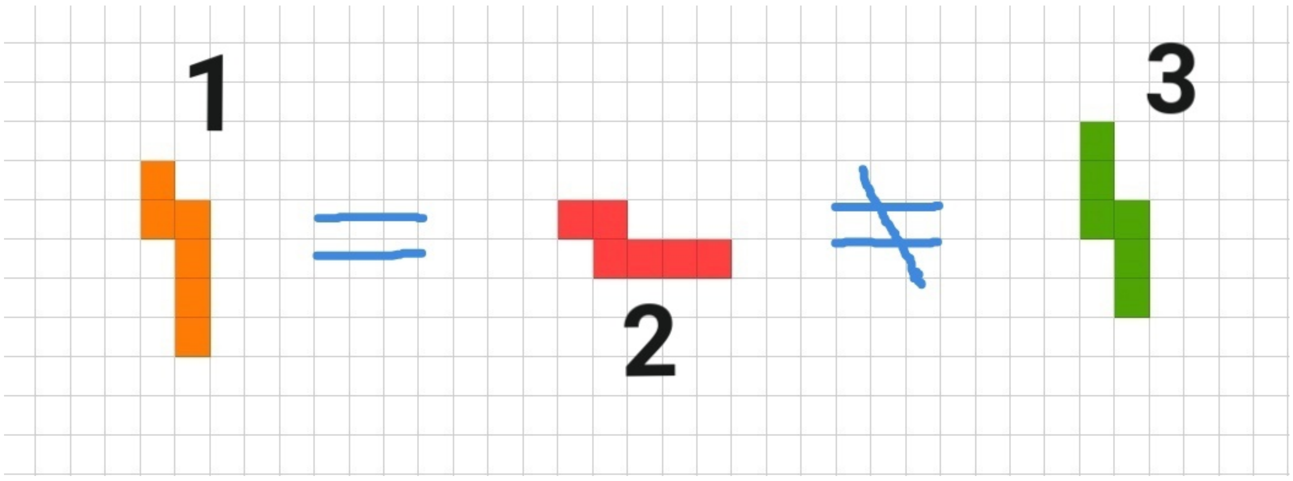
20
0 0 0 -1
-2 3
2 3
4 0
-4 0
-3 -2
3 -2
3 -4
-6 -2
-6 2
6 2
-1 1
-5 -5
-4 -5
2 0
-3 -4
4 -5
1 1
-2 0
6 -2
5 -5
	7 16 20 19 10 2 1 9 8 12 13 15 5 4 18
	11 17 3 14 6 7

Problem J. Multimino

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Let's define an n-mino as a connected figure composed of n squares 1×1. Let's say that two n-minoes are equal if one of them can be translated into another using the operations of parallel transport, rotation and reflection.

For example, in the illustration below, the red hexamino (number 2) is equal to the orange hexamino (number 1), and they both aren't equal to the green hexamino (number 3).



You are given a number n. Cut a checkered board of size $n \times n$ squares into n pairwise distinct non-intersecting n-minoes or report that no such splitting exists.

Input

The only line of the input contains a single integer n ($1 \leqslant n \leqslant 100$).

Output

Output a single integer -1 if the required splitting does not exist.

Otherwise, let's enumerate n-minoes with natural numbers from 1 to n. Output n rows with n integers in each row corresponding to the splitting of the board into n-minoes according to the principle that a cell in the i-th column and j-th row belongs to n-mino number a_{ij} ($1 \leqslant a_{ij} \leqslant n$).

All n-minoes in the splitting must be connected and pairwise distinct from each other. If there are several correct splittings for a board, output any one of them.

Examples

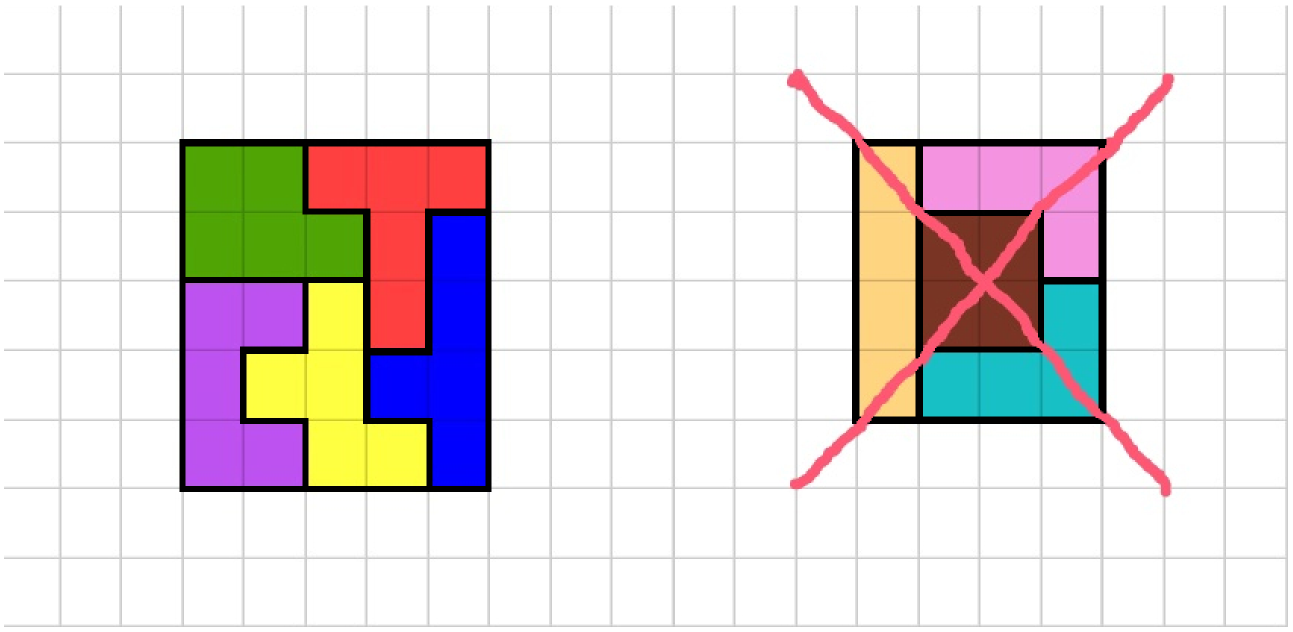
standard input	standard output

5	4 4 2 2 2
	4 4 4 2 3
	5 5 1 2 3
	5 1 1 3 3
	5 5 1 1 3

4	-1

Note

In the first example, the splitting looks like the picture on the left.



In the second example, the picture on the right would not be a valid splitting because the lilac (top) and turquoise (bottom) n-minoes can be translated into each other by parallel transport, rotation, and reflection. Therefore, they are equal.

Problem K. Postfix Notation Blocks

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Postfix notation (or reverse Polish notation) is a way to write arithmetic expressions without brackets, while still preserving the order of operations.

- The postfix notation for a single variable or number consists only of that variable or number.
- The postfix notation for an expression $A\#B$ (where A and B are expressions, and # is the operator evaluated last) is obtained by concatenating the postfix notation for A, the postfix notation for B, and #.

For example, the postfix notation for the expression $((a + f) \cdot b) \cdot (c \cdot d)$ is `af+b*cd**`.

Postfix notation can also be interpreted as instructions for a stack calculator:

- Each variable or number means "push it onto the stack".
- Each operator means "pop the two top items, apply the operation, and push the result back".

When evaluating `af+b*cd**`, the stack evolves as follows: `{a}`, `{a,f}`, `{a+f}`, `{a+f,b}`, `{(a+f)*b}`, `{(a+f)*b,c}`, `{(a+f)*b,c,d}`, `{(a+f)*b,c*d}`, `{((a+f)*b)*(c*d)}`.

You are given an expression in postfix notation, containing only variables and operators. All operators are **binary**, **associative**, and **commutative**. That means for any operator # and any expressions A, B, C:

- Associativity: $A\#(B\#C) = (A\#B)\#C$
- Commutativity: $A\#B = B\#A$

These rules allow you to rearrange operands for the same operator.

For example:

- As an expression: $((a + f) \cdot b) \cdot (c \cdot d)$ can be rearranged to $d \cdot ((a + f) \cdot (b \cdot c))$.
- In postfix notation: `af+b*cd**` can be rearranged to `daf+bc***`.

Your task is to rearrange the initial expression using only associativity and commutativity so that the resulting postfix expression has the **minimum RLE-size**.

The **RLE-size** of a string is the number of blocks of consecutive equal characters. For example, the RLE-size of `xx+yy+zz+**` is 7.

Input

The input consists of a single line containing a non-empty string s — the postfix expression.

- $1 \le |s| \le 2500$
- Each character of s is either:
  - a lowercase English letter ('`a`'–'`z`'), representing a variable, or

  - one of the operator characters (‘+’, ‘*’, ‘#’, ‘!’, ‘@’, ‘$’, ‘%’, ‘^’).

- The string s is guaranteed to be a valid postfix arithmetic expression.
- All operators are **binary**, **associative**, and **commutative**.

Output

Print a single integer — the minimum possible RLE-size of a postfix expression that can be obtained from the initial expression by applying any number (possibly zero) of the allowed rearrangements.

Examples

standard input	standard output
`af+b*cd**`	`7`
`xy*x*y*x*y*`	`3`
`abcdefg!@#$%^`	`13`

Note

In the first example `daf+bc***` is one of the possible best rearrangements.

In the second example the best ones are `xxxyyy*****` or `yyyxxx*****`.

There is no point in changing anything in the third example.

Problem L. Transformations

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

A sequence of n distinct integers represents the initial arrangement of n uniquely numbered individuals. Let this sequence be denoted as $a = [a_1, a_2, \ldots, a_n]$, where a_i is the identifier of the individual in the i-th position.

The objective is to rearrange the sequence into a target configuration denoted as $b = [b_1, b_2, \ldots, b_n]$, where b_i is the required identifier at position i in the final arrangement.

To perform the reordering, a specific operation is permitted, termed a **reorganization**. In each reorganization, the following process is carried out:

- A non-empty subset of elements (identified by their values) is selected from the current sequence.
- These elements are removed from their current positions in the sequence.
- The selected elements are then reversed and inserted at the beginning of the sequence.
- The relative order of the remaining (non-selected) elements remains unchanged.

Example: If the current sequence is: $[3, 4, 7, 6, 2, 5, 1]$ and the selected subset is $\{4, 7, 5\}$, the result after the reorganization is: $[5, 7, 4, 3, 6, 2, 1]$

The goal is to compute a sequence of at most 15 such reorganizations that transforms the initial sequence a into the target sequence b.

It is guaranteed that such a sequence of reorganizations exists.

Input

The first line contains one integer n — the number of individuals ($1 \le n \le 10\,000$).

Second line contains n distinct integer numbers a_i from 1 to n — the initial arrangement ($1 \le a_i \le n$). Third line contains n distinct integer numbers b_i from 1 to n — the target arrangement($1 \le b_i \le n$).

Output

In the first line, print an integer k ($0 \le k \le 15$) — the number of reorganizations in the found solution. In each of the following k lines, print description of the reorganizations, which should be done. For each reorganization first output number c_i — the number of individuals, that should go out of the line ($1 \le c_i \le n$), then c_i different integer numbers from 1 to n — the identifiers of people, who should go out of the line. The numbers of the people can be written in any order.

Examples

standard input	standard output

5
5 4 3 2 1
3 4 5 1 2
	4
	5 1 2 3 4 5
	1 5
	1 4
	1 3

7
3 4 7 6 2 5 1
2 6 3 4 5 7 1
	3
	3 6 5 7
	3 3 4 5
	3 2 6 3

Note

In the first test, the order of people is changing in the following way:

$5, 4, 3, 2, 1 \to 1, 2, 3, 4, 5 \to 5, 1, 2, 3, 4 \to 4, 5, 1, 2, 3 \to 3, 4, 5, 1, 2$

In the second test, the order of people is changing in the following way:

$3, 4, 7, 6, 2, 5, 1 \to 5, 6, 7, 3, 4, 2, 1 \to 4, 3, 5, 6, 7, 2, 1 \to 2, 6, 3, 4, 5, 7, 1$

Problem M. Distinctive Features

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

You are developing a system to assist consultants in smartphone stores.

All smartphones in the store are arranged in a row and numbered with integers from 1 to n in the order they are placed.

Each smartphone has certain **distinctive features**, such as durability or a large battery. There are a total of m different distinctive features, and they are numbered with integers from 1 to m.

The most common question that consultants have to answer is: how does this smartphone differ from those next to it? We formalize this question as follows:

Given a segment of smartphones numbered from l_i to r_i inclusive, and the number of a certain smartphone p_i in this segment ($l_i \le p_i \le r_i$), determine how many distinctive features are present in the given smartphone p_i but not in any other smartphone in the segment $[l_i, r_i]$.

To make the consultants' work easier, you have been tasked with developing a system capable of efficiently answering such queries.

Input

The first line contains two integers n and m ($1 \le n, m \le 500\,000$) — the number of smartphones in the store and the number of different distinctive features.

Each of the following n lines describes the distinctive features of the next smartphone in the following format:

At the beginning of the line is an integer k_i ($0 \le k_i \le m$) — the number of distinctive features of the i-th smartphone. Then in the same line are k_i integers $1 \le a_{i,1} < \ldots < a_{i,k_i} \le m$ — the distinctive features of the i-th smartphone in increasing order.

The next line contains an integer q ($1 \le q \le 500\,000$) — the number of queries.

The following q lines describe the queries. In the i-th line, there are three integers l_i, r_i, and p_i ($1 \le l_i \le p_i \le r_i \le n$) — the parameters of the i-th query.

Let s denote the total number of distinctive features across all smartphones ($s = \sum\limits_{i=1}^{n} k_i$). It is guaranteed that $n, m \le s \le 500\,000$.

Output

Output q integers — the number of distinctive features for each query.

Example

standard input	standard output

6 4
2 1 3
0
2 1 4
3 2 3 4
2 1 2
2 2 3
5
1 3 2
4 4 4
3 5 4
1 3 1
4 6 5
	0
	3
	1
	1
	1

Note

Consider the example.

In the first query, the second smartphone has no distinctive features, so the answer to the query is zero.

In the second query, the segment consists of a single element, so all distinctive features of the fourth smartphone apply.

In the third query, the distinctive features of the third smartphone are $[1, 4]$, the fourth smartphone has $[2, 3, 4]$, and the fifth smartphone has $[1, 2]$. Among all the distinctive features of the fourth smartphone, only feature 3 is unique to it, so the answer to the query is 1.