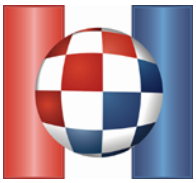

TASK	CSS	HISTOGRAMI	MOSTOVI	NIZOVI
input	standard input			
output	standard output			
time limit	5 seconds	1 second	1 second	1 second
memory limit	256 MB	256 MB	256 MB	256 MB
points	**100**	**100**	**100**	**100**
	total 400			

Popular HTML documents have a hierarchical structure. This means they consist of so-called *elements* that have a name and certain attributes and can contain other elements. In this task we consider documents that consist only of nested *div* elements. Specifically, each document consists of lines of the following form:

`<div id='`**name**`' class='`**class**$_1$ **class**$_2$ ... **class**$_K$`'>`	element beginning
...	element content
`</div>`	element end

Thus, a single element consists of a line that detones its beginning, a line that denotes its end, and in between these two lines is the content of the element which can either be empty of consist of a series of other elements that can again contain other elements and so on. The document itself consists of a series of elements (some of which may contain other elements). Additionally, the following holds for a representation of an element:

- The *name* of the element is an array of lowercase letters of the English alphabet. The name uniquely identifies the element in the document, and no two different elements have the same name.
- Each of the *classes* of elements is also an array of lowercase letters of the English alphabet. If a class is listed in the initial list of elements, we say that an element *belongs to that class*. Each element can belong to one or more different classes. Different elements can belong to same classes.
- The only space charactes that appear in the representation of an element are, respectively, one single space before `id`, one single space before `class` and possible spaces between individual classes, exactly one space between two adjacent classes.

If element E is written in the content of element F (anywhere between the lines that denote the beginning and the end of element E), we say that F *contains* E or, in other words, that E *is contained* in F. We say that element E is *the parent* of element F if F is contained in E and there is not a single element G such that F is contained in G and G is contained in E.

CSS is a style sheet language used for describing the look and formatting of HTML elements when seen in a Web browser. An important part of that language are so-called *selectors* and *classifiers*. A classifier is a series of characters that begins with a dot and is followed by one or more classes separated by dots. Again, only lowercase letters of the English alphabet must appear in classes. Some examples of classifiers are: ".banana.apple", ".banana", ".apple.banana.mango". These classifiers consist of two, one and three classes, respectively. We say that a certain HTML element *corresponds* to a classifier if it **belongs to all classes** that are listed in the classifier.

A selector consists of one or more classifiers and two adjacent classifiers in a selector can be separated with either a single space or a series of space-greater than-space characters (`' > '`). *Correspondence* of an HTML element to a selector is recursively defined in the following way:

- If S is a selector which consists of exactly one classifier k, the element corresponds to S if it corresponds to k.
- If S is a selector in the form of 'T k' where T is a selector and k is a classifier, the element E corresponds to S if it corresponds to k and is contained (not necessarily directly) in an element F which corresponds to T.
- If S is a selector in the form of '$T > k$' where T is a selector and k is a classifier, the element E corresponds to S if it corresponds to k and its parent F corresponds to T.

You are given a document which consists of N lines and you are given M selectors. Write a programme that will, for each given selector, output the names of all elements that correspond to it. For each individual selector, the names must be output in the order they appear in the given document.

INPUT

The first line of input contains an even integer N ($2 \leqslant N \leqslant 10\,000$), the number of lines in the document.

The following N lines contain the lines of the document. Each line will correspond to a beginning or an end of an element as described in the task.

The following line contains the integer M ($1 \leqslant M \leqslant 5$), the number of selectors.

Each of following M lines contains a single selector.

The names of all elements and classes in the input data will consist of only lowercase letters of the English alphabet and their length will not be greater than 20 characters.

Additionally, not a single line in the input data will be longer than 5 000 characters and the sum of the lengths of all the lines in the input will not exceed 5 000 000.

OUTPUT

For each selector in the input data, output a single line. Firstly, output the number of elements that correspond to the given selector and then the names of corresponding elements in aforementioned order.

BODOVANJE

In test cases worth 50% of total points, it will hold that selectors will not contain the character '>'.

SAMPLE TESTS

input

```
22
<div id='a' class='banana mango'>
<div id='b' class='orange'>
<div id='c' class='banana'>
<div id='d' class='apple mango orange'>
<div id='e' class='orange'>
</div>
</div>
</div>
</div>
<div id='f' class='orange apple'>
<div id='g' class='apple'>
<div id='h' class='orange apple'>
<div id='i' class='banana'>
</div>
</div>
</div>
<div id='j' class='banana'>
</div>
</div>
</div>
<div id='k' class='glupo voce daj mi sarme'>
</div>
5
.banana
.banana > .sarme
.banana > .orange > .banana
.banana .apple.orange > .orange
.mango > .orange .banana
```

output

```
4 a c i j
0
2 c j
1 e
3 c i j
```

A *histogram* is a graphical representation of a statistical distribution of data. In other words, it is a function that assigns a positive integer value to each number in the interval $0, 1, 2, \ldots, W-1$. For this task, we describe a histogram with a series of points in the standard coordinate system which, sequentially, determine the top edge of the area that the histogram encloses.

More specifically, we define a histogram of width W with an even number of points in the form:

$$(x_0, y_1), (x_1, y_1), (x_1, y_2), (x_2, y_2), (x_2, y_3), \ldots, (x_{N/2-1}, y_{N/2}), (x_{N/2}, y_{N/2}).$$

Therefore, starting from the first point, for every pair of adjacent points it must hold that their y coordinates are equal and that their x coordinates are equal. Thus, the histogram begins and ends with a horizontal segment and in between horizontal and vertical segments alternate.

Additionally, the following holds for the shape of a histogram:

1. $x_0 = 0$ - the histogram begins on the line $x = 0$.
2. $x_{N/2} = W$ - the histogram ends on the line $x = W$.
3. $x_i < x_{i+1}$, for each $i = 1, 2, \ldots, N/2 - 1$ - the histogram is defined from left to right and the length of each horizontal segment is at least one.
4. $y_i > 0$ - the height of the histogram is always greater than zero.
5. $y_i \neq y_{i+1}$ - the length of each vertical segment is at least one.

For a histogram H, let $y_H(x)$ be the height of the histogram in the interval $\langle x, x+1 \rangle$. For example, the surface area which the histogram encloses can be calculated by formula $\sum_{x=0}^{W-1} y_H(x)$. If two histograms, H and H', are given, with **equal width** W (which can possibly be defined with a different number of points), then we can measure the *difference* between these two histograms in various ways. This measure of difference between two histograms is also called *inaccuracy of histogram H' with regard to histogram H*. In this task, we examine two different ways of measuring differences between two histograms and we define them in the following way:

$$\text{diffcount}(H', H) = \sum_{x=0}^{W-1} \text{diff}(y_H(x), y_{H'}(x)), \text{where diff}(y_1, y_2) = 0 \text{ if } y_1 = y_2, \text{ and 0 else.}$$

$$\text{abserror}(H', H) = \sum_{x=0}^{W-1} |y_H(x) - y_{H'}(x)|$$

In other words, the first way measures the number of unit segments $\langle x, x+1 \rangle$ in which the two histograms differ, whereas the second way is the sum of absolute values of differences on those individual unit segments.

Write a program that will, for a given histogram H, set of points S and way of measuring inaccuracy, find and output histogram H' which only uses points from the set S in its definition and has the least possible inaccuracy with regard to the given histogram H.

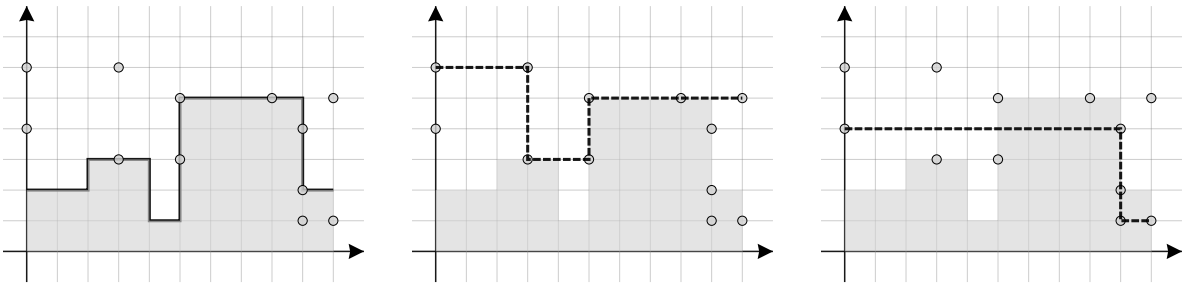

Figure 1: Illustration of the histogram and the set S in both test cases given below and the solutions for the first and the second way of measuring inaccuracy.

The definition of histogram H' must comply with all aforementioned conditions (for example, there shouldn't be two continuous horizontal segments in the definition of histogram H'), and each point in the definition must be one of the points from the set S.

INPUT

The first line of input contains integers N, M, G ($2 \leqslant N \leqslant 100\,000$, $2 \leqslant M \leqslant 100\,000, 1 \leqslant G \leqslant 2, N$ even), respectively: the total number of points in the definition of histogram H, number of points in set S and the number which determines what method of measuring the inaccuracy is used: 1 for diffcount and 2 for abserror.

Each of the following N lines contains integers X and Y ($0 \leqslant X \leqslant 10^6$, $1 \leqslant Y \leqslant 10^6$) - coordinates of one point from the definition of histogram H. The histogram's definition will comply to all the conditions from the task statement.

Each of the following M lines contains integers X and Y ($0 \leqslant X \leqslant 10^6$, $1 \leqslant Y \leqslant 10^6$) - coordinates of one point from set S. All the points in the set S will be different from one another, but they can match with points from the definition of the histogram H.

OUTPUT

The first line of output must contain the least possible inaccuracy D. The second line of output must contain an even integer L - the total number of points in the definition of the required optimal histogram. In each of the following L lines, output two integers X and Y - coordinates of the point in the histogram's definition.

The histogram's definition must comply to all the conditions from the task.

Please note: The solution may not be unique, but the input data will be such that there is always at least one solution.

SCORING

If the histogram's definition is incorrect or wasn't output, but the first line of output is correct (minimal inaccuracy), your solution will get 70% of total points for that test case.
In test cases worth 50 points, it will hold $G = 1$. In part of those test cases worth 25 points, it will additionally hold $M \leqslant 5\ 000$.
In test cases worth 50 points, it will hold $G = 2$. In part of these test cases worth 25 points, it will additionally hold $M \leqslant 5\ 000$.

SAMPLE TESTS

input

```
10 12 1
0 2
2 2
2 3
4 3
4 1
5 1
5 5
9 5
9 2
10 2
0 4
0 6
3 3
3 6
9 4
9 1
5 3
5 5
10 5
9 2
10 1
8 5
```

output

```
5
6
0 6
3 6
3 3
5 3
5 5
10 5
```

input

```
10 12 2
0 2
2 2
2 3
4 3
4 1
5 1
5 5
9 5
9 2
10 2
0 4
0 6
3 3
3 6
9 4
9 1
5 3
5 5
10 5
9 2
10 1
8 5
```

output

```
14
4
0 4
9 4
9 1
10 1
```

There are $2N$ towns in a certain country and a long river that runs from west to east. There are N towns on each side of the river. On the north side, they are labeled with numbers from 1 to N and on the south with numbers from $N+1$ to $2N$. The towns on each river bank are labeled in a way that the town further east on the bank always has the greater label.

On the north bank, there is a **one-way** road from each town (except town N) to its nearest town to the east. On the south bank, there is a **one-way** road from each town (except town $N+1$) to its nearest town to the west.

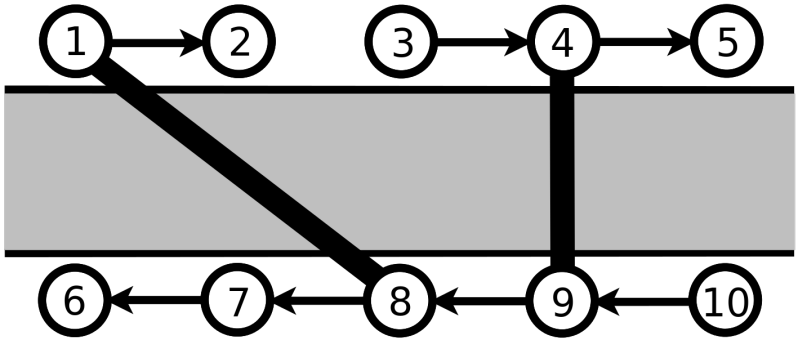

Figure 1: Figure accompanying the first test case after all the blockings and constructions are done.

Given the fact there is a lot of heavy traffic, the roads can sometimes get run down so they are permanently blocked because of safety reasons. In order to enable connectivity of cities that are not necessarily on the same side of the bank, the people in charge build bridges that directly connect some two cities located on opposite sides of the river. Bridge construction is a financially challenging task so they are built with special attention and they can never be run down. Because of the same reason, the bridges are **two-way**, unlike cheap roads on the banks of the river. Additionally, the constructed bridges will **never intersect**, not even in their starting or ending towns – thus, each town will be directly connected by a bridge with at most one town on the other side of the river.

Mirko works at the information office in a leading bus company. Every day hundreds of people come to him and ask whether it is possible to get from one town to another. Mirko then glances on roads that are currently passable and bridges built so far and checks if there is a way to get from one town to another. He likes this job very much, but it overlaps with his daily coffee break from 11 a.m. to 2 p.m. so he asked you to write a program that will do this job for him!

Write a program that will simulate M events given in chronological order. Each event is either an information about a newly blocked road or an information about a newly built bridge or an enquiry from a traveler regarding the existence of a way between two towns. The events are given in the following form:

- '`A` G_1 G_2' - a bridge is built between towns G_1 and G_2.
- '`B` G_1 G_2' - a one-way road between towns G_1 and G_2 is blocked.

- '`Q` G_1 G_2' - a traveler wants to know whether there is a way from town G_1 to town G_2, given the current passable roads and built bridges.

In the beginning, all roads are available and no bridges are built.

INPUT

The first line of input contains integers N ($1 \leqslant N \leqslant 10^9$) and M ($1 \leqslant \mathbf{M} \leqslant 200\,000$), the number of towns on **one bank** of the river and the number of events.

Each of the following M lines contains the description of a single event in the form described in the task. For town labels in the events, it will hold $1 \leqslant G_1, G_2 \leqslant 2N$. The towns G_1 and G_2 will always be different.

You can assume that the road which is being blocked was free up until that moment, and the bridge that is being built didn't exist up until then.

OUTPUT

For each traveler enquiry, you must print (in the order they were given) on the standard output '`DA`' (Croatian word for yes) if there is a way from town G_1 to town G_2 or print '`NE`' (Croatian word for no) otherwise.

SCORING

In test cases worth 30 points, it will hold $N,\ M \leqslant 1\,000$.
Furthermore, in test cases worth an additional 30 points, it will hold that $N \leqslant 10^9$, $M \leqslant 1\,000$.

SAMPLE TESTS

input	**input**
5 6 A 4 9 Q 1 7 B 3 2 Q 1 7 A 1 8 Q 1 7	6 10 A 3 7 A 4 10 Q 1 11 A 12 5 Q 2 11 B 10 11 Q 2 10 Q 9 6 B 1 2 Q 1 2
output	**output**
DA NE DA	NE DA DA DA NE

Mergesort is a quick sorting algorithm invented by John Von Neumann in 1945. In the heart of the algorithm lies a procedure that combines two already sorted sequences into a new sorted sequence. In this task you need to write a programme which does a similar thing.

Let A and B be sequences of **integers sorted in ascending order**, the sequence A is called *small sequence* and B *large sequence*. As it can be derived from these names, the sequence A contains less or equal number of elements than sequence B, and often the sequence A is **a lot shorter**. The sequence C is obtained by combining sequences A and B in such a way that first we sequentially take all the numbers from sequence A and then, after them, sequentially take all the numbers from sequence B. We use the notation $C[X]$ to denote the X^{th} element of sequence C where X is an integer in the interval from 1 to total number of elements in sequence C.

In the beginning, your programme knows only the lengths of sequences A and B, but not the sequence elements themselves. The programme must **sort the sequence C in ascending order** using only the following interactive commands:

- '`cmp` X Y' is a command which compares $C[X]$ and $C[Y]$ returns -1 if $C[X]$ is less than $C[Y]$, 1 if $C[X]$ is greater than $C[Y]$ and 0 if they are equal.
- '`reverse` X Y' is a command which reverses a subsequence of sequence C between the X^{th} and Y^{th} element (inclusive). For example, if sequence C is equal to $(2, 5, 2, 3)$ then the '`reverse` 2 4' command will alter it so it becomes $(2, 3, 2, 5)$.
- '`end`' is a command that denotes the end of interaction and your programme should call it when the sequence C is sorted in ascending order.

The cost of command '`reverse` X Y' is equal to the length of the reversed subsequence (in other words, $Y - X + 1$), whereas the rest of the commands do not have a cost. Write a programme that will sort the sequence C obtained by combining sorted sequences A and B as described above using **at most** 100 000 **commands in total** (including the command '`end`') and additionally, so that the total cost of all '`reverse`' commands is **at most** 3 000 000).

INTERACTION

Before interacting, your programme must read two integers from the standard input, N_A and N_B ($1 \leqslant N_A \leqslant 1\,000$, $N_A \leqslant N_B \leqslant 1\,000\,000$) - lengths of sequences A and B.

Afterwards, your programme can give commands by outputting using the standard output. Each command must be output in its own line and has to be in the exact form of one of the three commands described in the task. Regarding commands '`cmp`' and '`reverse`', X and Y have to be integers less than or equal to $N_A + N_B$, and regarding the command reverse, it additionally must hold that X is less than or equal to Y.

After each given command, your program needs to flush the standard output.

After command 'cmp', your programme must input one integer using the standard input - the command result. After other commands, your program must not input anything. After outputting the command 'end', your programme needs to flush the standard output and finish executing.

Please note: Code samples that interact correctly and flush the standard output are available for download through the evaluation system.

SCORING

In test cases worth 30 points, it will hold that N_A, $N_B \leqslant 50$.

In test cases worth an additional 30 points, it will hold that N_A, $N_B \leqslant 500$.

INTERACTION EXAMPLE

Output	Input	Sequence C	Cost
	2 3	-1 3 2 5 5	
cmp 1 4	-1	-1 3 2 5 5	
reverse 2 5		-1 5 5 2 3	4
cmp 2 3	0	-1 5 5 2 3	
reverse 4 5		-1 5 5 3 2	2
reverse 2 5		-1 2 3 5 5	4
end			

TESTING

You can test your solution in two ways, locally or through the evaluation system. In any case, you first need to create a file that will contain test cases for testing your solution.

The first line of file must contain two integers, N_A and N_B, lengths of sequences A and B. The following line must contain exactly N_A integers separated by a single space - elements of sequence A. The following line must contain exactly N_B integers separated by a single space - elements of sequence B. Both sequences must be sorted in ascending order and their elements must fit in a 32-bit signed integer data type.

For example, an input corresponding to the interaction example above is:

```
2 3
-1 3
2 5 5
```

In order to test using the evaluation system, you must first submit the source code of your solution using the page SUBMIT and then submit the test data using the page TEST.
Local testing (Unix only) is done using `nizovi_test` file which can be downloaded through the evaluation system. You need to write the following command:

```
./nizovi_test ./your_solution input_file
```

Whatever method of testing selected, the output will provide you with information whether your programme solved your test case correctly and information about the commands your programme has given and the answers your programme obtained. In case of local testing, these additional information will be output in the file `nizovi.log` in the current directory.