Problem A
Rearranging a Sequence

Input: Standard Input
Time Limit: 2 seconds

You are given an ordered sequence of integers, $(1, 2, 3, \ldots, n)$. Then, a number of requests will be given. Each request specifies an integer in the sequence. You need to move the specified integer to the head of the sequence, leaving the order of the rest untouched. Your task is to find the order of the elements in the sequence after following all the requests successively.

Input

The input consists of a single test case of the following form.

$$\begin{array}{l} n\ m \\ e_1 \\ \vdots \\ e_m \end{array}$$

The integer n is the length of the sequence ($1 \leq n \leq 200000$). The integer m is the number of requests ($1 \leq m \leq 100000$). The following m lines are the requests, namely $e_1, \ldots, e_m$, one per line. Each request e_i ($1 \leq i \leq m$) is an integer between 1 and n, inclusive, designating the element to move. Note that, the integers designate the integers themselves to move, not their positions in the sequence.

Output

Output the sequence after processing all the requests. Its elements are to be output, one per line, in the order in the sequence.

Sample Input 1	Sample Output 1

5 3
4
2
5
	5
	2
	4
	1
	3

Sample Input 2

```
10 8
1
4
7
3
4
10
1
3
```

Sample Output 2

```
3
1
10
4
7
2
5
6
8
9
```

In Sample Input 1, the initial sequence is $(1, 2, 3, 4, 5)$. The first request is to move the integer 4 to the head, that is, to change the sequence to $(4, 1, 2, 3, 5)$. The next request to move the integer 2 to the head makes the sequence $(2, 4, 1, 3, 5)$. Finally, 5 is moved to the head, resulting in $(5, 2, 4, 1, 3)$.

Problem B
Quality of Check Digits

Input: Standard Input
Time Limit: 1 second

The small city where you live plans to introduce a new social security number (SSN) system. Each citizen will be identified by a five-digit SSN. Its first four digits indicate the basic ID number (0000–9999) and the last one digit is a *check digit* for detecting errors.

For computing check digits, the city has decided to use an *operation table*. An operation table is a 10 × 10 table of decimal digits whose diagonal elements are all 0. Below are two example operation tables.

Operation Table 1

	0	1	2	3	4	5	6	7	8	9
0	0	3	1	7	5	9	8	6	4	2
1	7	0	9	2	1	5	4	8	6	3
2	4	2	0	6	8	7	1	3	5	9
3	1	7	5	0	9	8	3	4	2	6
4	6	1	2	3	0	4	5	9	7	8
5	3	6	7	4	2	0	9	5	8	1
6	5	8	6	9	7	2	0	1	3	4
7	8	9	4	5	3	6	2	0	1	7
8	9	4	3	8	6	1	7	2	0	5
9	2	5	8	1	4	3	6	7	9	0

Operation Table 2

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	9	0	1	2	3	4	5	6	7	8
2	8	9	0	1	2	3	4	5	6	7
3	7	8	9	0	1	2	3	4	5	6
4	6	7	8	9	0	1	2	3	4	5
5	5	6	7	8	9	0	1	2	3	4
6	4	5	6	7	8	9	0	1	2	3
7	3	4	5	6	7	8	9	0	1	2
8	2	3	4	5	6	7	8	9	0	1
9	1	2	3	4	5	6	7	8	9	0

Using an operation table, the check digit e for a four-digit basic ID number $abcd$ is computed by using the following formula. Here, $i \otimes j$ denotes the table element at row i and column j.

$$e = (((0 \otimes a) \otimes b) \otimes c) \otimes d$$

For example, by using Operation Table 1 the check digit e for a basic ID number $abcd = 2016$ is computed in the following way.

$$\begin{aligned} e &= (((0 \otimes 2) \otimes 0) \otimes 1) \otimes 6 \\ &= ((1 \otimes 0) \otimes 1) \otimes 6 \\ &= (7 \otimes 1) \otimes 6 \\ &= 9 \otimes 6 \\ &= 6 \end{aligned}$$

Thus, the SSN is 20166.

Note that the check digit depends on the operation table used. With Operation Table 2, we have $e = 3$ for the same basic ID number 2016, and the whole SSN will be 20163.

2016

$\Downarrow$ +check digit using Operation Table 2

20163

$\Downarrow$ human errors

$$\overbrace{\mathbf{0}0163 \quad \mathbf{1}0163 \quad \ldots \quad \mathbf{9}0163 \quad 2\mathbf{1}163 \quad 2\mathbf{2}163 \quad \ldots \quad 2016\mathbf{8} \quad 2016\mathbf{9}}^{\text{Altering one single digit}}$$

$$\overbrace{\mathbf{02}163 \quad 2\mathbf{10}63 \quad 20\mathbf{61}3 \quad 201\mathbf{36}}^{\text{Transposing two adjacent digits}}$$

Figure B.1. Two kinds of common human errors

The purpose of adding the check digit is to detect human errors in writing/typing SSNs. The following check function can detect certain human errors. For a five-digit number *abcde*, the check function is defined as follows.

$$\mathsf{check}(abcde) = ((((0 \otimes a) \otimes b) \otimes c) \otimes d) \otimes e$$

This function returns 0 for a correct SSN. This is because every diagonal element in an operation table is 0 and for a correct SSN we have $e = (((0 \otimes a) \otimes b) \otimes c) \otimes d$:

$$\mathsf{check}(abcde) = ((((0 \otimes a) \otimes b) \otimes c) \otimes d) \otimes e = e \otimes e = 0.$$

On the other hand, a non-zero value returned by check indicates that the given number cannot be a correct SSN. Note that, depending on the operation table used, check function may return 0 for an incorrect SSN. Kinds of errors detected depends on the operation table used; the table decides the quality of error detection.

The city authority wants to detect two kinds of common human errors on digit sequences: altering one single digit and transposing two adjacent digits, as shown in Figure B.1.

An operation table is *good* if it can detect all the common errors of the two kinds on all SSNs made from four-digit basic ID numbers 0000–9999. Note that errors with the check digit, as well as with four basic ID digits, should be detected. For example, Operation Table 1 is good. Operation Table 2 is not good because, for 20613, which is a number obtained by transposing the 3rd and the 4th digits of a correct SSN 20163, check(20613) is 0. Actually, among 10000 basic ID numbers, Operation Table 2 cannot detect one or more common errors for as many as 3439 basic ID numbers.

Given an operation table, decide how good it is by counting the number of basic ID numbers for which the given table cannot detect one or more common errors.

Input

The input consists of a single test case of the following format.

$$\begin{array}{l} x_{00}\ x_{01}\ \cdots\ x_{09} \\ \vdots \\ x_{90}\ x_{91}\ \cdots\ x_{99} \end{array}$$

The input describes an operation table with x_{ij} being the decimal digit at row i and column j. Each line corresponds to a row of the table, in which elements are separated by a single space. The diagonal elements x_{ii} ($i = 0, \ldots, 9$) are always 0.

Output

Output the number of basic ID numbers for which the given table cannot detect one or more common human errors.

Sample Input 1

```
0 3 1 7 5 9 8 6 4 2
7 0 9 2 1 5 4 8 6 3
4 2 0 6 8 7 1 3 5 9
1 7 5 0 9 8 3 4 2 6
6 1 2 3 0 4 5 9 7 8
3 6 7 4 2 0 9 5 8 1
5 8 6 9 7 2 0 1 3 4
8 9 4 5 3 6 2 0 1 7
9 4 3 8 6 1 7 2 0 5
2 5 8 1 4 3 6 7 9 0
```

Sample Output 1

```
0
```

Sample Input 2

```
0 1 2 3 4 5 6 7 8 9
9 0 1 2 3 4 5 6 7 8
8 9 0 1 2 3 4 5 6 7
7 8 9 0 1 2 3 4 5 6
6 7 8 9 0 1 2 3 4 5
5 6 7 8 9 0 1 2 3 4
4 5 6 7 8 9 0 1 2 3
3 4 5 6 7 8 9 0 1 2
2 3 4 5 6 7 8 9 0 1
1 2 3 4 5 6 7 8 9 0
```

Sample Output 2

```
3439
```

Sample Input 3

```
0 9 8 7 6 5 4 3 2 1
1 0 9 8 7 6 5 4 3 2
2 1 0 9 8 7 6 5 4 3
3 2 1 0 9 8 7 6 5 4
4 3 2 1 0 9 8 7 6 5
5 4 3 2 1 0 9 8 7 6
6 5 4 3 2 1 0 9 8 7
7 6 5 4 3 2 1 0 9 8
8 7 6 5 4 3 2 1 0 9
9 8 7 6 5 4 3 2 1 0
```

Sample Output 3

```
9995
```

Sample Input 4

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Sample Output 4

```
10000
```

Problem C
Distribution Center

Input: Standard Input
Time Limit: 3 seconds

The factory of the Impractically Complicated Products Corporation has many manufacturing lines and the same number of corresponding storage rooms. The same number of conveyor lanes are laid out in parallel to transfer goods from manufacturing lines directly to the corresponding storage rooms. Now, they plan to install a number of robot arms here and there between pairs of adjacent conveyor lanes so that goods in one of the lanes can be picked up and released down on the other, and also in the opposite way. This should allow mixing up goods from different manufacturing lines to the storage rooms.

Depending on the positions of robot arms, the goods from each of the manufacturing lines can only be delivered to some of the storage rooms. Your task is to find the number of manufacturing lines from which goods can be transferred to each of the storage rooms, given the number of conveyor lanes and positions of robot arms.

Input

The input consists of a single test case, formatted as follows.

$$
\begin{array}{l}
n\ m \\
x_1\ y_1 \\
\vdots \\
x_m\ y_m
\end{array}
$$

An integer n ($2 \le n \le 200000$) in the first line is the number of conveyor lanes. The lanes are numbered from 1 to n, and two lanes with their numbers differing with 1 are adjacent. All of them start from the position $x = 0$ and end at $x = 100000$. The other integer m ($1 \le m < 100000$) is the number of robot arms.

The following m lines indicate the positions of the robot arms by two integers x_i ($0 < x_i < 100000$) and y_i ($1 \le y_i < n$). Here, x_i is the x-coordinate of the i-th robot arm, which can pick goods on either the lane y_i or the lane $y_i + 1$ at position $x = x_i$, and then release them on the other at the same x-coordinate.

You can assume that positions of no two robot arms have the same x-coordinate, that is, $x_i \ne x_j$ for any $i \ne j$.

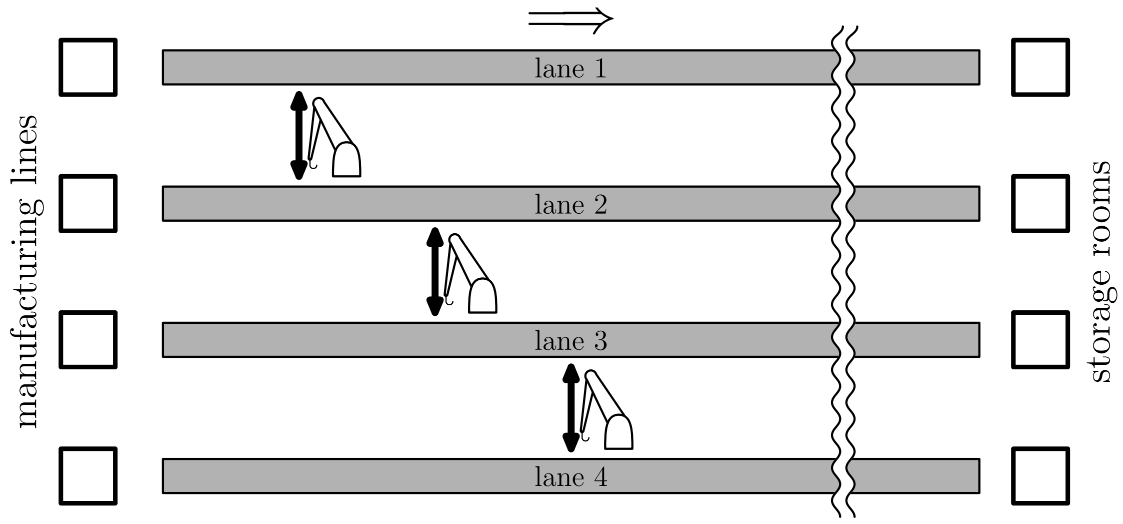

Figure C.1. Illustration of Sample Input 1

Output

Output n integers separated by a space in one line. The i-th integer is the number of the manufacturing lines from which the storage room connected to the conveyor lane i can accept goods.

Sample Input 1

```
4 3
1000 1
2000 2
3000 3
```

Sample Output 1

```
2 3 4 4
```

Sample Input 2

```
4 3
1 1
3 2
2 3
```

Sample Output 2

```
2 4 4 2
```

Problem D
Hidden Anagrams

Input: Standard Input
Time Limit: 10 seconds

An *anagram* is a word or a phrase that is formed by rearranging the letters of another. For instance, by rearranging the letters of "William Shakespeare," we can have its anagrams "I am a weakish speller," "I'll make a wise phrase," and so on. Note that when A is an anagram of B, B is an anagram of A.

In the above examples, differences in letter cases are ignored, and word spaces and punctuation symbols are freely inserted and/or removed. These rules are common but not applied here; only exact matching of the letters is considered.

For two strings s_1 and s_2 of letters, if a substring s_1' of s_1 is an anagram of a substring s_2' of s_2, we call s_1' a *hidden anagram* of the two strings, s_1 and s_2. Of course, s_2' is also a *hidden anagram* of them.

Your task is to write a program that, for given two strings, computes the length of the longest hidden anagrams of them.

Suppose, for instance, that "anagram" and "grandmother" are given. Their substrings "nagr" and "gran" are hidden anagrams since by moving letters you can have one from the other. They are the longest since any substrings of "grandmother" of lengths five or more must contain "d" or "o" that "anagram" does not. In this case, therefore, the length of the longest hidden anagrams is four. Note that a substring must be a sequence of letters occurring *consecutively* in the original string and so "nagrm" and "granm" are not hidden anagrams.

Input

The input consists of a single test case in two lines.

s_1
s_2

s_1 and s_2 are strings consisting of lowercase letters (`a` through `z`) and their lengths are between 1 and 4000, inclusive.

Output

Output the length of the longest hidden anagrams of s_1 and s_2. If there are no hidden anagrams, print a zero.

Sample Input 1

```
anagram
grandmother
```

Sample Output 1

```
4
```

Sample Input 2

```
williamshakespeare
iamaweakishspeller
```

Sample Output 2

```
18
```

Sample Input 3

```
aaaaaaaabbbbbbbb
xxxxxabababxxxxxabab
```

Sample Output 3

```
6
```

Sample Input 4

```
abababacdcdcd
efefefghghghghgh
```

Sample Output 4

```
0
```

Problem E
Infallibly Crack Perplexing Cryptarithm

Input: Standard Input
Time Limit: 2 seconds

You are asked to crack an encrypted equation over binary numbers.

The original equation consists only of binary digits ("0" and "1"), operator symbols ("+", "-", and "*"), parentheses ("(" and ")"), and an equal sign ("="). The encryption replaces some occurrences of characters in an original arithmetic equation by letters. Occurrences of one character are replaced, if ever, by the same letter. Occurrences of two different characters are never replaced by the same letter. Note that the encryption does not always replace all the occurrences of the same character; some of its occurrences may be replaced while others may be left as they are. Some characters may be left unreplaced. Any character in the Roman alphabet, in either lowercase or uppercase, may be used as a replacement letter. Note that cases are significant, that is, "a" and "A" are different letters. Note that not only digits but also operator symbols, parentheses and even the equal sign are possibly replaced.

The arithmetic equation is derived from the start symbol of Q of the following context-free grammar.

$$\begin{array}{rcl} Q & ::= & E\texttt{=}E \\ E & ::= & T \mid E\texttt{+}T \mid E\texttt{-}T \\ T & ::= & F \mid T\texttt{*}F \\ F & ::= & N \mid \texttt{-}F \mid \texttt{(}E\texttt{)} \\ N & ::= & \texttt{0} \mid \texttt{1}B \\ B & ::= & \varepsilon \mid \texttt{0}B \mid \texttt{1}B \end{array}$$

Here, ε means empty.

As shown in this grammar, the arithmetic equation should have one equal sign. Each side of the equation is an expression consisting of numbers, additions, subtractions, multiplications, and negations. Multiplication has precedence over both addition and subtraction, while negation precedes multiplication. Multiplication, addition and subtraction are left associative. For example, x-y+z means (x-y)+z, not x-(y+z). Numbers are in binary notation, represented by a sequence of binary digits, 0 and 1. Multi-digit numbers always start with 1. Parentheses and negations may appear redundantly in the equation, as in ((--x+y))+z.

Write a program that, for a given encrypted equation, counts the number of different possible original correct equations. Here, an equation should conform to the grammar and it is correct when the computed values of the both sides are equal.

For Sample Input 1, `C` must be `=` because any equation has one `=` between two expressions. Then, because `A` and `M` should be different, although there are equations conforming to the grammar, none of them can be correct.

For Sample Input 2, the only possible correct equation is `-0=0`.

For Sample Input 3 (B-A-Y-L-zero-R), there are three different correct equations, `0=-(0)`, `0=(-0)`, and `-0=(0)`. Note that one of the two occurrences of zero is not replaced with a letter in the encrypted equation.

Input

The input consists of a single test case which is a string of Roman alphabet characters, binary digits, operator symbols, parentheses and equal signs. The input string has at most 31 characters.

Output

Print in a line the number of correct equations that can be encrypted into the input string.

Sample Input 1	Sample Output 1
`ACM`	`0`

Sample Input 2	Sample Output 2
`icpc`	`1`

Sample Input 3	Sample Output 3
`BAYL0R`	`3`

Sample Input 4	Sample Output 4
`-AB+AC-A`	`1`

Sample Input 5	Sample Output 5
`abcdefghi`	`0`

Sample Input 6	Sample Output 6
`111-10=1+10*10`	`1`

Sample Input 7	Sample Output 7
`0=10-1`	`0`

Problem F
Three Kingdoms of Bourdelot

Input: Standard Input
Time Limit: 4 seconds

You are an archaeologist at the Institute for Cryptic Pedigree Charts, specialized in the study of the ancient Three Kingdoms of Bourdelot. One day, you found a number of documents on the dynasties of the Three Kingdoms of Bourdelot during excavation of an ancient ruin. Since the early days of the Kingdoms of Bourdelot are veiled in mystery and even the names of many kings and queens are lost in history, the documents are expected to reveal the blood relationship of the ancient dynasties.

The documents have lines, each of which consists of a pair of names, presumably of ancient royal family members. An example of a document with two lines follows.

```
Alice Bob
Bob Clare
```

Lines should have been complete sentences, but the documents are heavily damaged and therefore you can read only two names per line. At first, you thought that all of those lines described the true ancestor relations, but you found that would lead to contradiction. Later, you found that some documents should be interpreted negatively in order to understand the ancestor relations without contradiction. Formally, a document should be interpreted either as a *positive* document or as a *negative* document.

- In a positive document, the person on the left in each line is an ancestor of the person on the right in the line. If the document above is a positive document, it should be interpreted as “Alice is an ancestor of Bob, and Bob is an ancestor of Clare”.
- In a negative document, the person on the left in each line is NOT an ancestor of the person on the right in the line. If the document above is a negative document, it should be interpreted as “Alice is NOT an ancestor of Bob, and Bob is NOT an ancestor of Clare”.

A single document can never be a mixture of positive and negative parts. The document above can never read “Alice is an ancestor of Bob, and Bob is NOT an ancestor of Clare”.

You also found that there can be ancestor-descendant pairs that didn’t directly appear in the documents but can be inferred by the following rule: For any persons x, y and z, if x is an ancestor of y and y is an ancestor of z, then x is an ancestor of z. For example, if the document above is a positive document, then the ancestor-descendant pair of “Alice Clare” is inferred.

You are interested in the ancestry relationship of two royal family members. Unfortunately, the interpretation of the documents, that is, which are positive and which are negative, is unknown. Given a set of documents and two distinct names p and q, your task is to find an interpretation of the documents that does not contradict with the hypothesis that p is an ancestor of q. An interpretation of the documents contradicts with the hypothesis if and only if there exist persons x and y such that we can infer from the interpretation of the documents and the hypothesis that

- x is an ancestor of y and y is an ancestor of x, or
- x is an ancestor of y and x is not an ancestor of y.

We are sure that every person mentioned in the documents had a unique single name, i.e., no two persons have the same name and one person is never mentioned with different names.

When a person A is an ancestor of another person B, the person A can be a parent, a grandparent, a great-grandparent, or so on, of the person B. Also, there can be persons or ancestor-descendant pairs that do not appear in the documents. For example, for a family tree shown in Figure F.1, there can be a positive document:

```
A H
B H
D H
F H
E I
```

where persons C and G do not appear, and the ancestor-descendant pairs such as “A E”, “D F”, and “C I” do not appear.

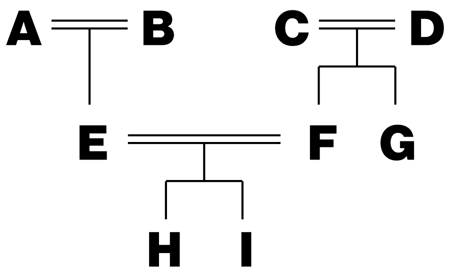

Figure F.1. A Family Tree

Input

The input consists of a single test case of the following format.

$$\begin{array}{l} p\ q \\ n \\ c_1 \\ \vdots \\ c_n \end{array}$$

The first line of a test case consists of two distinct names, p and q, separated by a space. The second line of a test case consists of a single integer n that indicates the number of documents. Then the descriptions of n documents follow.

The description of the i-th document c_i is formatted as follows:

$$\begin{array}{l} m_i \\ x_{i,1}\ y_{i,1} \\ \vdots \\ x_{i,m_i}\ y_{i,m_i} \end{array}$$

The first line consists of a single integer m_i that indicates the number of pairs of names in the document. Each of the following m_i lines consists of a pair of distinct names $x_{i,j}$ and $y_{i,j}$ ($1 \leq j \leq m_i$), separated by a space.

Each name consists of lowercase or uppercase letters and its length is between 1 and 5, inclusive.

A test case satisfies the following constraints.

- $1 \leq n \leq 1000$.
- $1 \leq m_i$.
- $\sum_{i=1}^{n} m_i \leq 100000$, that is, the total number of pairs of names in the documents does not exceed 100000.
- The number of distinct names that appear in a test case does not exceed 300.

Output

Output "`Yes`" (without quotes) in a single line if there exists an interpretation of the documents that does not contradict with the hypothesis that p is an ancestor of q. Output "`No`", otherwise.

Sample Input 1

```
Alice Bob
3
2
Alice Bob
Bob Clare
2
Bob Clare
Clare David
2
Clare David
David Alice
```

Sample Output 1

```
No
```

Sample Input 2

```
Alice David
3
2
Alice Bob
Bob Clare
2
Bob Clare
Clare David
2
Clare David
David Alice
```

Sample Output 2

```
Yes
```

Sample Input 3

```
Alice Bob
1
2
Clare David
David Clare
```

Sample Output 3

```
Yes
```

Problem G
Placing Medals on a Binary Tree

Input: Standard Input
Time Limit: 4 seconds

You have drawn a chart of a perfect binary tree, like one shown in Figure G.1. The figure shows a finite tree, but, if needed, you can add more nodes beneath the leaves, making the tree arbitrarily deeper.

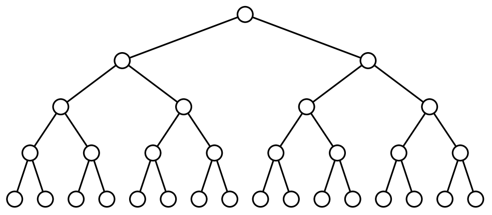
Figure G.1. A Perfect Binary Tree Chart

Tree nodes are associated with their depths, defined recursively. The root has the depth of zero, and the child nodes of a node of depth d have their depths $d + 1$.

You also have a pile of a certain number of medals, each engraved with some number. You want to know whether the medals can be placed on the tree chart satisfying the following conditions.

- A medal engraved with d should be on a node of depth d.
- One tree node can accommodate at most one medal.
- The path to the root from a node with a medal should not pass through another node with a medal.

You have to place medals satisfying the above conditions, one by one, starting from the top of the pile down to its bottom. If there exists no placement of a medal satisfying the conditions, you have to throw it away and simply proceed to the next medal.

You may have choices to place medals on different nodes. You want to find the best placement. When there are two or more placements satisfying the rule, one that places a medal upper in the pile is better. For example, when there are two placements of four medal, one that places only the top and the 2nd medal, and the other that places the top, the 3rd, and the 4th medal, the former is better.

In Sample Input 1, you have a pile of six medals engraved with 2, 3, 1, 1, 4, and 2 again respectively, from top to bottom.

- The first medal engraved with 2 can be placed, as shown in Figure G.2 (A).
- Then the second medal engraved with 3 may be placed , as shown in Figure G.2 (B).
- The third medal engraved with 1 cannot be placed if the second medal were placed as stated above, because both of the two nodes of depth 1 are along the path to the root from nodes already with a medal. Replacing the second medal satisfying the placement conditions, however, enables a placement shown in Figure G.2 (C).
- The fourth medal, again engraved with 1, cannot be placed with any replacements of the three medals already placed satisfying the conditions. This medal is thus thrown away.
- The fifth medal engraved with 4 can be placed as shown in of Figure G.2 (D).
- The last medal engraved with 2 cannot be placed on any of the nodes with whatever replacements.

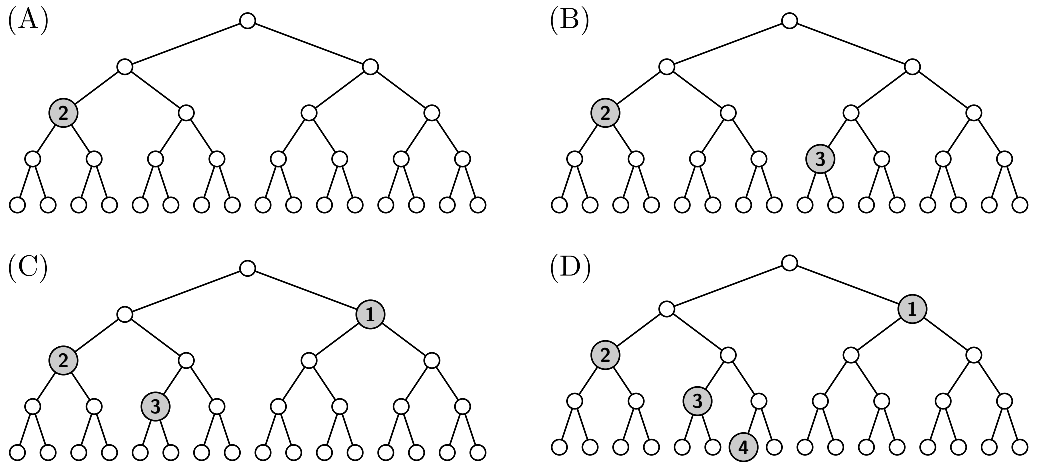

Figure G.2. Medal Placements

Input

The input consists of a single test case in the format below.

$$\begin{array}{l} n \\ x_1 \\ \vdots \\ x_n \end{array}$$

The first line has n, an integer representing the number of medals ($1 \leq n \leq 5 \times 10^5$). The following n lines represent the positive integers engraved on the medals. The i-th line of which has an integer x_i ($1 \leq x_i \leq 10^9$) engraved on the i-th medal of the pile from the top.

Output

When the best placement is chosen, for each i from 1 through n, output `Yes` in a line if the i-th medal is placed; otherwise, output `No` in a line.

Sample Input 1

```
6
2
3
1
1
4
2
```

Sample Output 1

```
Yes
Yes
Yes
No
Yes
No
```

Sample Input 2

```
10
4
4
4
4
4
4
4
4
1
4
```

Sample Output 2

```
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
Yes
No
```

Problem H
Animal Companion in Maze

Input: Standard Input
Time Limit: 2 seconds

George, your pet monkey, has escaped, slipping the leash!

George is hopping around in a maze-like building with many rooms. The doors of the rooms, if any, lead directly to an adjacent room, not through corridors. Some of the doors, however, are one-way: they can be opened only from one of their two sides.

He repeats randomly picking a door he can open and moving to the room through it. You are chasing him but he is so quick that you cannot catch him easily. He never returns immediately to the room he just has come from through the same door, believing that you are behind him. If any other doors lead to the room he has just left, however, he may pick that door and go back.

If he cannot open any doors except one through which he came from, voilà, you can catch him there eventually.

You know how rooms of the building are connected with doors, but you don't know in which room George currently is.

It takes one unit of time for George to move to an adjacent room through a door.

Write a program that computes how long it may take before George will be confined in a room. You have to find the longest time, considering all the possibilities of the room George is in initially, and all the possibilities of his choices of doors to go through.

Note that, depending on the room organization, George may have possibilities to continue hopping around forever without being caught.

Doors may be on the ceilings or the floors of rooms; the connection of the rooms may not be drawn as a planar graph.

Input

The input consists of a single test case, in the following format.

$$
\begin{array}{l}
n\ m \\
x_1\ y_1\ w_1 \\
\vdots \\
x_m\ y_m\ w_m
\end{array}
$$

The first line contains two integers n ($2 \le n \le 100000$) and m ($1 \le m \le 100000$), the number of rooms and doors, respectively. Next m lines contain the information of doors. The i-th line of these contains three integers x_i, y_i and w_i ($1 \le x_i \le n, 1 \le y_i \le n$, $x_i \ne y_i$, $w_i = 1$ or 2), meaning that the i-th door connects two rooms numbered x_i and y_i, and it is one-way from x_i to y_i if $w_i = 1$, two-way if $w_i = 2$.

Output

Output the maximum number of time units after which George will be confined in a room. If George has possibilities to continue hopping around forever, output "`Infinite`".

Sample Input 1	Sample Output 1
2 1 1 2 2	1

Sample Input 2	Sample Output 2

2 2
1 2 1
2 1 1
	Infinite

Sample Input 3	Sample Output 3

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

Sample Input 4	Sample Output 4

3 2
1 3 1
1 3 1
	1

Problem I
Skinny Polygon

Input: Standard Input
Time Limit: 3 seconds

You are asked to find a polygon that satisfies all the following conditions, given two integers, x_{bb} and y_{bb}.

- The number of vertices is either 3 or 4.
- Edges of the polygon do not intersect nor overlap with other edges, i.e., they do not share any points with other edges except for their endpoints.
- The x- and y-coordinates of each vertex are integers.
- The x-coordinate of each vertex is between 0 and x_{bb}, inclusive. Similarly, the y-coordinate is between 0 and y_{bb}, inclusive.
- At least one vertex has its x-coordinate 0.
- At least one vertex has its x-coordinate x_{bb}.
- At least one vertex has its y-coordinate 0.
- At least one vertex has its y-coordinate y_{bb}.
- **The area of the polygon does not exceed 25000.**

The polygon may be non-convex.

Input

The input consists of multiple test cases. The first line of the input contains an integer n, which is the number of the test cases ($1 \leq n \leq 10^5$). Each of the following n lines contains a test case formatted as follows.

x_{bb} y_{bb}

x_{bb} and y_{bb} ($2 \leq x_{bb} \leq 10^9$, $2 \leq y_{bb} \leq 10^9$) are integers stated above.

Output

For each test case, output description of one polygon satisfying the conditions stated above, in the following format.

$$\begin{array}{l} v \\ x_1\ y_1 \\ \vdots \\ x_v\ y_v \end{array}$$

Here, v is the number of vertices, and each pair of x_i and y_i gives the coordinates of the i-th vertex, (x_i, y_i). The first vertex (x_1, y_1) can be chosen arbitrarily, and the rest should be listed either in clockwise or in counterclockwise order.

When more than one polygon satisfies the conditions, any one of them is acceptable. You can prove that, with the input values ranging as stated above, there is at least one polygon satisfying the conditions.

Sample Input 1

```
2
5 6
1000000000 2
```

Sample Output 1

```
4
5 6
0 6
0 0
5 0
3
1000000000 0
0 2
999999999 0
```

Problem J
Cover the Polygon with Your Disk

Input: Standard Input
Time Limit: 5 seconds

A convex polygon is drawn on a flat paper sheet. You are trying to place a disk in your hands to cover as large area of the polygon as possible. In other words, the intersection area of the polygon and the disk should be maximized.

Input

The input consists of a single test case, formatted as follows. All input items are integers.

$$\begin{array}{l} n\ r \\ x_1\ y_1 \\ \vdots \\ x_n\ y_n \end{array}$$

n is the number of vertices of the polygon ($3 \le n \le 10$). r is the radius of the disk ($1 \le r \le 100$). x_i and y_i give the coordinate values of the i-th vertex of the polygon ($1 \le i \le n$). Coordinate values satisfy $0 \le x_i \le 100$ and $0 \le y_i \le 100$.

The vertices are given in counterclockwise order. As stated above, the given polygon is convex. In other words, interior angles at all of its vertices are less than 180°. Note that the border of a convex polygon never crosses or touches itself.

Output

Output the largest possible intersection area of the polygon and the disk. The answer should not have an error greater than 0.0001 (10^{-4}).

Sample Input 1	Sample Output 1

4 4
0 0
6 0
6 6
0 6
	35.759506

Sample Input 2

```
3 1
0 0
2 1
1 3
```

Sample Output 2

```
2.113100
```

Sample Input 3

```
3 1
0 0
100 1
99 1
```

Sample Output 3

```
0.019798
```

Sample Input 4

```
4 1
0 0
100 10
100 12
0 1
```

Sample Output 4

```
3.137569
```

Sample Input 5

```
10 10
0 0
10 0
20 1
30 3
40 6
50 10
60 15
70 21
80 28
90 36
```

Sample Output 5

```
177.728187
```

Sample Input 6

```
10 49
50 0
79 10
96 32
96 68
79 90
50 100
21 90
4 68
4 32
21 10
```

Sample Output 6

```
7181.603297
```

Problem K
Black and White Boxes

Input: Standard Input
Time Limit: 2 seconds

Alice and Bob play the following game.

1. There are a number of straight piles of boxes. The boxes have the same size and are painted either black or white.
2. Two players, namely Alice and Bob, take their turns alternately. Who to play first is decided by a fair random draw.
3. In Alice's turn, she selects a black box in one of the piles, and removes the box together with all the boxes above it, if any. If no black box to remove is left, she loses the game.
4. In Bob's turn, he selects a white box in one of the piles, and removes the box together with all the boxes above it, if any. If no white box to remove is left, he loses the game.

Given an initial configuration of piles and who plays first, the game is a *definite perfect information game.* In such a game, one of the players has sure win provided he or she plays best. The draw for the first player, thus, essentially decides the winner.

In fact, this seemingly boring property is common with many popular games, such as chess. The chess game, however, is complicated enough to prevent thorough analyses, even by supercomputers, which leaves us rooms to enjoy playing.

This game of box piles, however, is not as complicated. The best plays may be more easily found. Thus, initial configurations should be fair, that is, giving both players chances to win. A configuration in which one player can always win, regardless of who plays first, is undesirable.

You are asked to arrange an initial configuration for this game by picking a number of piles from the given candidate set. As more complicated configuration makes the game more enjoyable, you are expected to find the configuration with the maximum number of boxes among fair ones.

Input

The input consists of a single test case, formatted as follows.

$$\begin{array}{l} n \\ p_1 \\ \vdots \\ p_n \end{array}$$

A positive integer n (≤ 40) is the number of candidate piles. Each p_i is a string of characters `B` and `W`, representing the i-th candidate pile. `B` and `W` mean black and white boxes, respectively. They appear in the order in the pile, from bottom to top. The number of boxes in a candidate pile does not exceed 40.

Output

Output in a line the maximum possible number of boxes in a fair initial configuration consisting of some of the candidate piles. If only the empty configuration is fair, output a zero.

Sample Input 1

```
4
B
W
WB
WB
```

Sample Output 1

```
5
```

Sample Input 2

```
6
B
W
WB
WB
BWW
BWW
```

Sample Output 2

```
10
```