Welcome to Santa's Algorithmic Challenge!

Good luck and have a great time!

Santa and the organizers

Problem A. Dolls

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second(s)
Memory limit:	256 MiB

Santa's toy factory in Lapland is getting ready for the beginning of the Christmas season, and production is in full swing. Now the assembly line is churning out lots of parts for dolls: heads, legs, bodies and arms.

To create one doll, the elves need two arms, two legs, one head and one body. Unfortunately, they don't know how many pieces of each part have been produced, and after production the parts are all dumped into one container where they mix, which creates a huge mess. The elves are having trouble computing the maximum quantity of dolls that can be assembled from the produced parts. Help them!

Input

The first and only line of the input contains a single nonempty string of length at most 50. Each character of this string is one of the following:

- `h`, which represents a head,
- `l`, which represents a leg,
- `a`, which represents an arm,
- `b`, which represents a body.

Output

In the only line of the output, write the maximum number of dolls that can be assembled from the given parts.

Examples

stdin	stdout
`llaahb`	`1`
`lllllllllll`	`0`

Note

There is no difference between a left and a right arm, nor between a left and a right leg.

Problem B. Fairness

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 second(s)
Memory limit:	256 MiB

As usual, Santa is very concerned about being fair to children. He has a big bag of n presents and he is going to choose k of them to distribute. For each present, Santa knows its value. He wants to select the k presents so as to minimize *unfairness*, which is the difference between the highest and the lowest value of a present. Help him!

Input

The first line of the input contains two space-separated integers n and k ($2 \leq k \leq n \leq 100000$) – the number of all presents and the number of presents to select, respectively. The second line of the input contains n space-separated integers a_i ($0 \leq a_i \leq 10^9$) – the values of the n presents.

Output

Your program should print a single line to standard output, consisting of k space-separated integers – the values of presents that Santa should select so as to minimize unfairness. If there are multiple possible choices of k presents with the same minimum unfairness, print any of them.

Examples

stdin	stdout
6 5 1 1 2 2 3 3	1 1 2 3 3

Note

In the example testcase, Santa must select all presents but one. No matter which one he leaves out, the unfairness will be $3 - 1 = 2$.

Problem C. The Claussifier

Input file:	`stdin`
Output file:	`stdout`
Time limit:	5 second(s)
Memory limit:	256 MiB

Algorithms are a central tool for making Santa Claus's operations run on a worldwide scale. Consider just the problem of classifying EPFL students as naughty or nice – because of the large numbers, this would be impossible to do manually. This is why Santa has developed The Claussifier – an automatic machine which, given some information about a student, outputs a single integer number. This integer has the following interpretation: if it is even, then the student is nice (and might be getting some chocolate), and if it is odd, then the student is considered to be naughty (and should rather expect old socks).

Unfortunately, the mischievous villain Sonta Clausson is not asleep. Again, he managed to sneak into Santa's headquarters in Lapland while dressed up as Santa, and snatch The Claussifier (he wants to understand the hidden algorithm and analyze its running time). As a prank, Clausson replaced The Claussifier with a device of his own design, which reads two integers – a student's shoe size x and height in centimeters y – and outputs one integer number. This integer is computed using a ridiculous arbitrary-looking arithmetic expression involving some viciously chosen constants and the numbers x and y. (Again, an even output means a nice student, odd is naughty.)

The prank mostly worked – while Santa sensed that something is wrong with The Claussifier and eventually found out what had happened, some presents have already been sent out. Now the presents of those students who have been misclassified will need to be recalled. To figure out which presents to recall, Santa must know what the output of Sonta's mis-Claussifier is for a given student. Santa was able to read the arithmetic expression which Sonta's mis-Claussifier uses, but since it is rather huge, he is asking for your help.

Santa was smart enough to notice that the parity of Sonta's mis-Claussifier's output depends only on the parities of x and y, not on their exact values. This is why he is giving you these parities and the arithmetic expression, and asking you to compute the parity of the result.

Input

The first line of input holds a string s ($1 \leq |s| \leq 10^6$) – an arithmetic expression built using digits, operators `+`, `-`, `*`, and two variables x and y. This expression does not contain any spaces or parentheses.

The second and third lines of input hold the parities of x and y, respectively, in the following format: `Odd` if the number is odd and `Even` if it is even.

It is guaranteed that in the initial expression there are no adjacent symbols `+`, `-` and `*`, the line begins with a digit or a variable, numbers don't have leading zeroes, and to the left and to the right of each variable there is an operator, a digit, line beginning or line end.

Output

Output a single word: `Odd` if the value of the expression is odd, or `Even` if it is even.

Examples

stdin	stdout
1+23+x+456*y-7 Odd Even	Even
x*x*x*y-x Odd Even	Odd

Problem D. Game

Input file:	stdin
Output file:	stdout
Time limit:	2 second(s)
Memory limit:	256 MiB

This Christmas, Santa brought little Justin and his brother Austin the best present ever. It is a brand-name, factory-new-smelling sequence of positive integers $a_1, a_2, ..., a_n$, in ascending order! The package came complete with an instruction on how to play a two-person game using the sequence.

The players move in turns. In each turn the current player selects a suffix of the array and subtracts 1 from the suffix. That is, he chooses an index i (with $1 \le i \le n$) and subtracts 1 from each number $a_i, a_{i+1}, ..., a_n$. If, as a result of the move, any two numbers in the sequence become equal or any number becomes 0, the player loses the game. Otherwise, the other player makes a move, and so on.

Justin is allowed to choose which of them starts the game. Of course, he wants to win. Should he go first or second?

Input

The first line of the input contains a single integer n ($1 \le n \le 10^5$) – the length of the sequence from Santa.

The second line of the input contains n space-separated integers $a_1, ..., a_n$ ($1 \le a_i \le 10^9$). The sequence is increasing: $a_{i-1} < a_i$ for all $1 < i \le n$.

Output

Output `First` or `Second` depending on whether Justin should play first or second in order to win. You should assume that both Justin and Austin play perfectly.

Examples

stdin	stdout
5 1 2 3 4 5	Second
1 100	First

Problem E. Traveling Santa Problem

Input file:	stdin
Output file:	stdout
Time limit:	5 second(s)
Memory limit:	256 MiB

Santa will be arriving in Lausanne any moment now. Before this happens, though, he needs your help to plan his route inside the city. As you can imagine, his objective is to visit as many streets as possible, so that he can leave presents in houses on these streets.

The reindeers are going to drop Santa off at a crossroads of his choice. From there, he will make his way through the city on foot, walking on the streets.

Lausanne is very hilly, Santa's bag is massive, and he hasn't got any slimmer lately... All these factors combined made his walking experience last year rather miserable (notable quotes include "I'm too old for this" and "*something* Switzerland"). This year, Santa simply refuses to walk uphill (that is, from a crossroads at a lower altitude to a crossroads at a higher altitude). Because in Lausanne every crossroads is at a different altitude, this effectively means that every street can be traveled by Santa only in one direction – except for those weird ones which begin and end at the same crossroads. At the end, the reindeers will pick Santa up from where he ended his route.

Given a map of Lausanne, plan the longest possible route for Santa.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 10^5$) – the number of streets in Lausanne.

Each of the next n lines contains two space-separated integers a_i and b_i ($0 \leq a_i \leq b_i \leq 2 \cdot 10^5$) describing a single street – where a_i and b_i are the altitudes of the crossroads which the i-th street connects. There is at most one crossroads at any possible altitude.

Output

Output a single integer – the maximum number of streets that Santa can traverse during his walk.

Examples

stdin	stdout
7 1 3 2 3 1 2 1 1 3 4 1 1 0 1	6

Note

In the sample testcase, Santa can walk as follows: $4 \to 3 \to 2 \to 1 \to 1 \to 1 \to 0$. He will have visited 6 streets (note that there are two different streets which both begin and end at crossroads 1) – all except the street `1 3`.

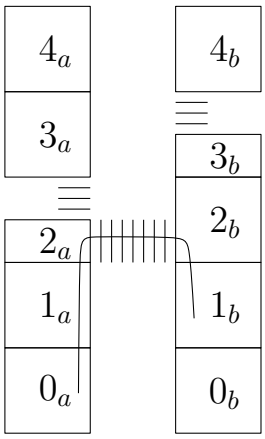
Problem F. Street

Input file:	`stdin`
Output file:	`stdout`
Time limit:	4 second(s)
Memory limit:	256 MiB

Santa's job is not easy. Just last night he was leaving presents in all houses along a certain street, and this one really annoying kid stayed awake all night sitting by the chimney just to catch him. Since he does not want to be seen, he had to skip that house and return to the street tonight, just to visit that single house again.

The street has two sidewalks, one on each side. The left side has buildings numbered 0_a, 1_a, 2_a and so on, and the right side has buildings numbered 0_b, 1_b, 2_b and so on. For each i, the building i_a is across the street from building i_b.

Santa thinks that people can only see him when he walks on a crossing, so his main objective is to minimize the number of crossings he walks through. The street has three types of crossings: I-crossings, which allow one to cross the street (to go from the left to the right sidewalk or vice versa), and left or right T-crossings, which are crossings of this street with another perpendicular street on the left or on the right side (that is, if one walks on the left sidewalk and there is a T-crossing with a street on the left, then one needs to cross it; thus, left T-crossings separate neighbouring houses on the left side of the street, e.g., houses 2_a and 3_a on the picture below).



Santa gave you a map of the street and is asking you to compute the best path for him from the place where he is standing now, which is 0_a, to f_b, which is the house of the annoying kid. The map is a list of crossings, given as follows:

- a left T-crossing is identified by a single house number x, which means that the crossing is on the left of the street between houses x_a and $(x+1)_a$,
- a right T-crossing is identified by a single house number x, which means that the crossing is on the right of the street between houses x_b and $(x+1)_b$,
- an I-crossing is identified by a single house number x, and allows Santa to cross the street between houses x_a and x_b (in either direction). (If he does not want to cross the street, he doesn't have to enter the crossing.)

Help Santa!

Input

The first line of the input contains three space-separated integers n, m and k ($1 \le n, m, k \le 10^5$) – numbers of left T-crossings, right T-crossings, and I-crossings, respectively.

The second line contains n space-separated integers x ($0 \le x \le 10^5$) – a list of all left T-crossings.

The third line contains m space-separated integers x ($0 \le x \le 10^5$) – a list of all right T-crossings.

The fourth line contains k space-separated integers x ($0 \le x \le 10^5$) – a list of all I-crossings.

The fifth line contains a single integer f ($0 \le f \le 10^5$) – the number of the house f_b on the right where the kid lives.

Output

Output a single integer: the minimum number of crossings necessary.

Examples

stdin	stdout
1 1 1 2 3 2 1	1
4 3 3 1 4 5 8 8 9 11 0 7 15 14	4

Note

The first sample input is shown on the picture above.

Problem G. Linear Programming

Input file:	`stdin`
Output file:	`stdout`
Time limit:	10 second(s)
Memory limit:	256 MiB

Little Chris broke his leg while playing football, just before Christmas! Now he cannot move too much at all, and pretty much the only reasonable gift for him is a book. This made Santa's job easy – knowing about his interest in algorithms and computer science, he brought Chris the book *Linear Programming For Dummies*.

The mesmerized Chris immediately started studying the book. He learned that linear programming is not a computer programming technique, but that it is about solving systems of linear inequalities involving a set of variables $x_1, x_2, ..., x_n$. The book did not immediately jump to general algorithms such as the simplex method – it is a beginner book, after all – but started off with an exercise designed to whet the reader's appetite. The exercise asked how to, given a set of inequalities of the form $x_i + x_j > 0$, $x_i - x_j > 0$, $-x_i + x_j > 0$ or $-x_i - x_j > 0$, find out whether it is *feasible*: that is, whether there exist real numbers $x_1, ..., x_n \in \mathbb{R}$ such that all these inequalities are satisfied.

Chris sees that sometimes there is no solution: for example, if there are four inequalities $x_1 + x_2 > 0$, $x_3 + x_4 > 0$, $-x_1 - x_3 > 0$, $-x_2 - x_4 > 0$, then adding the first two shows that $x_1 + x_2 + x_3 + x_4 > 0$, whereas adding the third and the fourth shows that $x_1 + x_2 + x_3 + x_4 < 0$. However, the poor kid cannot see a general method to understand which systems are feasible. Can you help him?

Input

In the first line of input there are two space-separated integers n and m ($1 \leq n \leq 100\,000$, $1 \leq m \leq 500\,000$). They specify the number of variables and the number of inequalities, respectively.

Each of the m lines that follow describes one inequality, encoded by a `+` or `-` sign, an integer i ($1 \leq i \leq n$), another `+` or `-` sign, and an integer j ($1 \leq j \leq n$), separated by single spaces. These correspond to the inequality $\pm x_i \pm x_j > 0$ (with the signs as preceding i and j in the description). It might happen that $i = j$.

Output

The first and only line of the output should contain a single word: `YES` if the system of inequalities is satisfiable (feasible), and `NO` otherwise.

Examples

stdin	stdout
3 3 + 1 - 2 - 3 + 1 + 2 - 3	YES
3 3 + 1 - 2 + 3 - 1 + 2 - 3	NO

Note

The first system of inequalities is satisfiable, e.g. by $x_1 = 3$, $x_2 = 2$, $x_3 = 1$. The second system of inequalities is not satisfiable.

Problem H. Santastic

Input file:	`stdin`
Output file:	`stdout`
Time limit:	6 second(s)
Memory limit:	512 MiB

Santa is trying to come up with a *santastic* collection of gifts to be the prize for solving this problem. (How meta is this?) He has a bag of n presents and he will select a subset of them.

Each present in his bag has a label – a nonempty string of lowercase letters. For reasons which will probably never become clear to mortals, Santa computes the *santasticity* of a collection of gifts as the product of three numbers:

- the number of gifts in the collection,
- the length of the longest common prefix of the labels of all gifts in the collection,
- the length of the longest common suffix of the labels of all gifts in the collection.

Help Santa select the most santastic collection and win it!

Input

The first line of the input contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) – the number of available gifts. Each of the next n lines contains one string – the label of a gift. The total length of all the labels does not exceed $2 \cdot 10^5$.

Output

Output a single integer – the maximum possible santasticity.

Examples

stdin	stdout
4 aa aa aa bb	12
4 aba ab abbba aba	25
5 abac abaac aac abac abc	32

Note

In the first sample testcase, the best collection consists of the three first gifts. Their longest prefix and suffix are both 2, so santasticity is $3 \cdot 2 \cdot 2 = 12$.

In the second sample testcase, it is optimal to choose only the third gift. The santasticity will be $1 \cdot 5 \cdot 5 = 25$.

In the third sample testcase, take the two `abac` gifts.

We do not guarantee that this problem is solvable in slow languages (like Python). We recommend C++ or Java.