

Problem A. Almost Palindrome

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

A string is considered palindrome if it's the same if you read it from the left or from the right (if it stays the same when you reverse it).

In this problem you are given a string, and your task is to swap 2 characters at different positions to make the string palindrome (note that you must do this step exactly once).

Input

The input contains just 1 line which is the string you are going to check, the string consists of at least 2 and at most 500,000 lower case English letters.

Output

Print "YES" if you can make the given string palindrome by applying the above step exactly once, otherwise print "NO".

Examples

standard input	standard output
ab	NO
aba	YES
axax	YES

Note

In the second test case, you can swap the first and last characters (which won't make any change), and the string will stay palindrome.

P.S. The judge is cute enough today and didn't allow strings with length 1.

Problem B. Wet Shark and Coordinate Plane Game

Input file: `stdin`
Output file: `stdout`
Time limit: 1 second
Memory limit: 256 megabytes

Wet Shark is now trying to help Dry Shark cheat in a game so that Dry Shark can win. The game is played on a regular xy coordinate grid. There is a stack of n cards. Wet Shark will hand all n cards to Dry Shark, in some order. When Dry Shark receives a card, he must move in the direction stated by the card.

In the game, there are 2 types of cards, as follows:

Type 1 cards are in the form "Right X , Left Y ." This means that Wet Shark must move right X units and then move left Y units, in that order.

Type 2 cards are in the form "Up X , Down Y ." This means that Wet Shark must move up X units and then move down Y units, in that order.

Dry Shark initially starts on the point $(1, 1)$. If Wet Shark ever crosses or lands on the x or y axis after he is handed any card, he becomes consumed by the Evil Sharks, and therefore loses the game. If Wet Shark can hand Dry Shark the cards in some order so that he never gets consumed by the Evil Sharks, Dry Shark wins the game.

Determine if there is a way Wet Shark can hand Dry Shark the cards so that Dry Shark can win the game.

The Evil Sharks, however, still want to eat Dry Shark in the game. Therefore, help the Evil Sharks determine what point Dry Shark will land on if he wins.

Input

The first line contains an integer n ($1 \leq n \leq 1000$).

Each of the next n lines contains 3 space-separated integers t_i, X_i, Y_i ($1 \leq t_i \leq 2, -1000 \leq X_i, Y_i \leq 1000$), representing the type and values written on the i th card. If $t_i = 1$, a Type 1 card is depicted; X and Y are the units Dry Shark is to move right and left, respectively. If $t_i = 2$, a Type 2 card is depicted; X and Y are the units Dry Shark is to move up and down, respectively.

Output

On the first line of the output, output a single "YES" or "NO" (quotes for clarity). Output "YES" if Dry Shark can win and "NO" if Dry Shark cannot win.

If Dry Shark can win, output the coordinates of the point Wet Shark will be at after Wet Shark gives him all the cards. Check the sample output for the exact format.

Examples

stdin	stdout
5 1 4 1 1 2 7 2 4 9 2 5 6 2 3 4	NO
2 1 50 1 2 2 1	YES (50, 2)

Note

Note that the X_i, Y_i could be negative. Going right $-n$ blocks is equivalent to going left n blocks, and going up $-m$ blocks is equivalent to going down m blocks.

Problem C. Wet Shark and Digit Enemies

Input file: `stdin`
Output file: `stdout`
Time limit: 4 seconds
Memory limit: 256 megabytes

Wet Shark has a function $\text{SquareDigitSum}(\text{number})$, which returns the sum of the squares of the digits of a number. For example, $\text{SquareDigitSum}(45) = 4^2 + 5^2 = 16 + 25 = 41$ and $\text{SquareDigitSum}(105) = 1^2 + 0^2 + 5^2 = 1 + 0 + 25 = 26$.

Dry Shark is trying to fight many enemies. These enemies are strange, however, and each enemy is represented by a different positive integer, let's call it ID. Dry Shark has a function $\text{GetPower}(\text{ID})$, which takes the ID of the enemy and returns the amount of power that enemy has. The body of the $\text{GetPower}(\text{ID})$ function is $\text{return}(\text{ID} - \text{SquareDigitSum}(\text{ID}))$, which utilises the function that Wet Shark gave to him. Note that $\text{GetPower}(45) = 45 - 41 = 4$ and $\text{GetPower}(105) = 105 - 26 = 79$.

Dry Shark is given a number n . For a number n , Dry Shark knows there could potentially be any enemy whose number has n digits in it. Formally, given an integer n , the enemies could be represented as any integer from 10^{n-1} to $10^n - 1$, inclusive. For example, if $n = 2$, enemies could be represented by any integer from 10 to 99 inclusive.

Dry Shark wants to know four things:

- 1) The ID of any of the enemies with maximum power
- 2) The ID of any of the enemies with minimum power
- 3) The power of the enemies with maximum power
- 4) The power of the enemies with minimum power

Input

The only line contains an integer n ($1 \leq n \leq 2,500,000$).

Output

Output 4 lines, one integer on each line, representing the things Dry Shark wants to know.

The first line should state the ID of any of the enemies with maximum power.

The second line should state the ID of any of the enemies with minimum power.

The third line should state the power of the enemies with maximum power.

The fourth line should state the power of the enemies with minimum power.

If there are multiple solutions to any of Wet Shark's queries, you can output any of them.

Examples

stdin	stdout
1	1 9 0 -72

Note

Problem note: 001 is not a valid 3-digit ID because it is not in the range 10^2 to $10^3 - 1$. 0 is not a valid 1-digit ID for a similar reason.

Sample Test 1: We can tabulate $\text{GetPower}(n)$ for all 1-digit IDs:

$$\text{GetPower}(1) = 1 - 1^2 = 0$$

$$\text{GetPower}(2) = 2 - 2^2 = -2$$

$$\text{GetPower}(3) = 3 - 3^2 = -6$$

$$\text{GetPower}(4) = 4 - 4^2 = -12$$

$$\text{GetPower}(5) = 5 - 5^2 = -20$$

$$\text{GetPower}(6) = 6 - 6^2 = -30$$

$$\text{GetPower}(7) = 7 - 7^2 = -42$$

$$\text{GetPower}(8) = 8 - 8^2 = -56$$

$$\text{GetPower}(9) = 9 - 9^2 = -72$$

Therefore, the minimum is -72, the maximum is 0, and values 9 and 1, respectively, produce these.

Problem D. Wet Shark and Smallest Number

Input file: `stdin`
Output file: `stdout`
Time limit: 1 second
Memory limit: 256 megabytes

Today, Wet Shark was given a list of non-necessarily distinct prime integers, $a_1, a_2, \dots, a_n$.

Wet Shark defines k as the smallest integer such that for $1 \leq i \leq n$, $k \bmod a_i = a_i - 1$.

Wet Shark also has an additional integer, o . Tell Wet Shark whether or not k is divisible by o .

Input

The first line contains one integer, n $1 \leq n \leq 100000$.

The second line consists of n space separated integers, the list of primes. Each prime ranges from 2 to 100000.

The third line consists of the additional integer, o , ranging from 1 to 100000.

Output

If k is divisible by o , output "YES". Else, output "NO".

Examples

stdin	stdout
1 3 2	YES
2 3 3 4	NO

Problem E. Mike and Cities

Input file: **standard input**
Output file: **standard output**
Time limit: 3 seconds
Memory limit: 256 megabytes

Mike has a vacation of m days, and he's going to explore n cities numbered from 1 to n . He can start to visit on the first day any city. If on day d he visited city c_1 then on the day $d+1$ he can visit city c_1 (stay where he's for one more day) or city c_2 if there is a road between cities c_1 and c_2 .

He can visit a city any number of times. There are k types of roads, each type is described using 3 integers a_i , b_i and c_i , signifying that there are bidirectional roads between cities $(a_i \text{ and } c_i)$, $(a_i + 1 \text{ and } c_i)$... $(b_i \text{ and } c_i)$ note that there can be a road between a city and itself, it is also given the fact that $c_i \neq c_j$ ($1 \leq i < j \leq k$).

In day i ($1 \leq i \leq m$), if he visits city j ($1 \leq j \leq n$) he gets pleasure $P_{i,j}$.

Help him find the maximal sum of pleasure in all the days he can achieve.

Input

The first line contains 3 integers n , m and k ($1 \leq n \times m$, $k \leq 300,000$).

The next m lines contains n integers separated by a space, representing matrix P ($0 \leq P_{i,j} \leq 10^9$).

The next k lines, each one contains 3 integers separated by a space, representing the different types of roads – a_i , b_i , c_i ($1 \leq a_i \leq b_i \leq n$, $1 \leq c_i \leq n$).

Output

Output the maximal sum of pleasures he can achieve.

Example

standard input	standard output
3 3 2 1 2 3 5 6 7 11 5 3 1 1 2 2 3 3	20