Problem Eclair. Open Olympiad in Design

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

Nowadays, many of the people who prepare programming competitions are the former participants. This is good because former competitors not only know many competition details, but are also able to prepare the statements, admin the judgement system and do a lot of other interesting (or not so interesting) stuff. How would the things look like if the competition will be prepared by designers?

In one imaginary world the Open Olympiad in Design takes place. There are n problem, which are prepared by n statements designers (one for each problem), one designer of problem names and one font designer. Each of the n designers who prepare statements has already finished his job and left some fixed space for the problem name. In particular, the lengths of the name of the i-th problem should be equal to exactly l_i characters.

According to competition rules problem names should be made of Unicode characters, be **distinct** and be located in lexicographical order (check the notes section). Font designer asked the problem names designer to pick such names that all conditions are satisfied and least possible number of distinct letters is used, so the amount of job he has to do is minimized.

Find out the minimum possible amount of distinct letters, required to obtain n words of given lengths such that they are distinct and go in lexicographical order. Please, note that you are **not allowed** to change the order of the problems.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 100\,000$) — the number of problems.

The second line contains n integers $l_1, l_2, \ldots, l_n$ ($1 \leq l_i \leq 10^9$) — lengths of the problem names.

Output

Print one integer — the minimum possible amount of distinct letters font designer has to paint in order to make it possible for names designer to achieve his goal.

Examples

input	output
5 2 2 2 2 2	3
4 3 1 2 2	2

Note

String $x_1x_2\ldots x_a$ of length a is called *lexicographically smaller* than string $y_1y_2\ldots y_b$ of length b, if one of the two following statements holds:

- In the first position i such that $x_i \neq y_i$ the first string has the smaller symbol than the second string, i.e. $x_1 = y_1$, $x_2 = y_2$, …, $x_{i-1} = y_{i-1}$, $x_i < y_i$;
- the first string is a strict prefix of a second string, i.e. $a < b$ and $x_1 = y_1$, $x_2 = y_2$, …, $x_a = y_a$.

The sequence of distinct words is said to be sorted in lexicographical order if each word (except the last one) is lexicographically smaller than the next word.

In the first sample, it's enough to use characters ‘`a`’ < ‘`o`’ < ‘`x`’ and names “`aa`”, “`ao`”, “`ax`”, “`ox`”, “`xx`”.

In the second sample, only two distinct letters are required, for example ‘`l`’ $<$ ‘`o`’ and names “`lol`”, “`o`”, “`ol`” and “`oo`”.

Scoring

Tests for this problem are divided into four groups. For each of the groups you earn points only if your solution passes all tests in this group and all tests in all **previous** groups (except sample tests). **Offline evaluation** means that your submission will be evaluated on the tests of the group only after the end of the contest. It’s not required for the solution to pass samples to be evaluated on other tests.

Group	Points	Additional constraints		Comment
		n	l_i	
0	0	–	–	Sample tests
1	11	$n \leq 10$	$l_i \leq 5$	all l_i are equal
2	7	$n \leq 10$	$l_i \leq 5$	
3	20	$n \leq 300$	$l_i \leq 300$	
4	20	$n \leq 5000$	$l_i \leq 5000$	
5	21	–	$l_i \leq 200\,000$	**Offline evaluation**
6	21	–	–	**Offline evaluation**

Problem Muffin. Robot on the Field

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

Andrysha is fond of computer games. He recently he found another exciting game:

Game world is an infinite plane, divided into cells. Each cell has its own game coordinates. Player controls a robot, initially located in the cell with coordinates $(0, 0)$. Player is able to move robot left, up, right and down by pressing keyboard buttons. In fact:

- every time button '`L`' is pressed, the first coordinate of the robot position decreases by 1.
- every time button '`U`' is pressed, the second coordinate of the robot position increases by 1.
- every time button '`R`' is pressed, the first coordinate of the robot position increases by 1.
- every time button '`D`' is pressed, the second coordinate of the robot position decreases by 1.

The goal is to move robot to some given cell (x, y). Though the game sounds easy, Andrysha was unable to finish it and asked his friend Zhenya for help. Zhenya told him a sequence of pressings of the buttons that he used, but this hint didn't help (maybe Zhenya had a different target cell)! Andrysha went mad and slammed keyboard so hard that broke some buttons. Broken buttons no longer react on pressings and this might be really helpful!

Andrysha wants to find out, if he can finish the game using Zhenya's instructions and breaking some buttons at some moments of time. Pressing broken button doesn't affect the robot. Andrysha can break any button before any pressure of any button or after any pressure of any button. Moreover, he can break several buttons (even all of them) at the same time. He also can break buttons before the very first pressure. He always breaks all the remaining buttons at the end of the game.

Input

The first line of the input contains one integer n ($1 \le n \le 1\,000\,000$) — the length of the Zhenya's list of instructions.

The following line contains n characters '`L`', '`U`', '`R`', '`D`' — list of the instructions.

The third line of the input contains two integers x and y ($-1\,000\,000 \le x, y \le 1\,000\,000$), denoting coordinates of the target cell, where the robot should be located after performing all the instructions.

Output

If there is no way Andrysha can succeed, print one integer -1.

Otherwise output four integers from 0 to n, denoting the moment of time when Andrysha should break buttons "`L`", "`U`", "`R`", "`D`", respectively. If a button should be broken before all pressings, print 0 for it. As Andrysha breaks all working buttons after game, **every printed number** should be between 0 and n.

If there is more than one solution, print any of them.

Examples

input	output
4 LRUD 3 2	-1
4 DURL 0 0	0 0 0 0
8 LLURDRRD 1 -1	1 7 6 8

Note

In the first sample, the destination cell is too far from the starting cell, so the robot can't reach it in four or less pressings.

In the second sample, all buttons can be broken before the beginning of the game, so robot will stay at the starting-destination point.

In the third sample, if all buttons are broken according to the answer, the resulting sequence of robot's actions will be the `LURDRD` and robot will reach the destination.

Scoring

Tests for this problem are divided into four groups. For each of the groups you earn points only if your solution passes all tests in this group and all tests in all **previous** groups.

Group	Points	Additional constraints	Comment
		n	
0	0	–	Sample tests
1	32	$n \leq 20$	
2	29	$n \leq 500$	
3	39	–	

Problem Napoleon. Metropolis Development

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

In the year 20*yy* Moscow city completely ran out of space for the new construction works. The government is actively looking for the new income opportunities, and it finally announced that all railway tracks within the city boundaries are going to be replaced by the underground tunnels. Quite obviously, freed space will be used for development.

Reconstruction process begins with the section of Octyabrskaya Railway Tracks, which is k meters long. Vacant space will attract a lot of tourists, but should balance its pressure onto the tunnel — that means only the most important businesses get a chance to use the newly emerged area, like coffee stops and ice-cream vans.

Renovated track consists of k equal segments, conveniently numbered from 1 to k. There are n companies willing to use the new territory, and i-th of them would like to use all the segments numbered from l_i to r_i. Every company provided a detailed construction plan including integer p_i — the building's pressure. Every company's application can either be rejected, or accepted: there is no way to build only a part of any.

Since the major is quite greedy, he wants to make sure that each segment is used by at least one company. However, for safety reasons it was decided to minimize the maximal pressure on any segment. Please note that it's *possible* for one segment to be used by multiple companies, and in this case the total pressure is determined as a sum of individual pressure values of the buildings.

Please help the major to accept a set of applications so for every segment there is at least one accepted application which intend to use this segment, and the maximal pressure among the segments is as small as possible.

Input

The first line of the input contains two integers n and k ($1 \le n \le 100\,000$, $1 \le k \le 10^9$) — the number of applications and the number of individual segments.

The following n lines describe the applications. Every application consists of three integers l_i, r_i, p_i ($1 \le l_i \le r_i \le 10^9$, $1 \le p_i \le 10^9$), indicating the leftmost point, the rightmost point and the pressure, respectively.

Output

Output one number — the lowest possible maximum pressure among all segments for a set of applications satisfying the requirements, or -1 if such set of applications doesn't exist.

Examples

input	output
`3 4`	`3`
`1 3 1`	
`3 4 2`	
`1 4 5`	
`1 3`	`-1`
`1 2 1`	
`4 5`	`8`
`1 4 3`	
`4 5 5`	
`1 1 3`	
`1 2 1`	
`4 5`	`1`
`1 4 1`	
`4 5 1`	
`3 4 1`	
`5 5 1`	

Note

In the first example the optimal strategy is to accept the first two applications. In this case the maximum pressure is attained on the third segment and equals 3.

In the second example there are no companies willing to use the third segment, and thus it's not possible to satisfy the requirements.

In the third example one of the optimal solutions is to accept all the applications. The maximum pressure is attained on the fourth segment and equals 8. Note that you don't need to minimize or maximize the total number of accepted applications.

In the fourth example the optimal solution is to accept the first and the fourth applications, leading to a valid configuration where the pressure of every segment is equal to 1.

Scoring

Tests for this problem are divided into six groups. For each of the groups you earn points only if your solution passes all tests in this group and all tests in some of the previous groups (see the scoring table). **Offline evaluation** means that your submission will be evaluated on the tests of the group only after the end of the contest. It's not required for the solution to pass samples to be evaluated on remaining tests.

Group	Points	Additional constraints		Required groups	Comment
		n	p_i		
0	0	—	—	–	Sample tests
1	8	$n \leq 10$	$p_i \leq 10^9$	–	
2	15	$n \leq 3000$	$p_i = 1$	–	
3	21	$n \leq 3000$	$p_i \leq 10^9$	1	
4	16	$n \leq 100\,000$	$p_i = 1$	2	
5	40	–	–	1 – 4	**Offline evaluation**

Problem Panna Cotta. Shall We Play a Game?

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

This is an **interactive** problem.

1804. Vice-president of the United States of America Aaron Burr challenged gubernatorial race candidate Alexander Hamilton for writing several offending pamphlets.

But Burr is sane enough to understand that, even if he kills Hamilton, he will lose his reputation and ruin his career. So, the enemies decided to simply play a game. To make everything more fair they decided to play it g times.

Each game starts with Hamilton thinking up a positive integer n, and after that Burr tries to guess it. For any positive integer x Burr may ask Hamilton about fraction of numbers between 1 and n inclusive that are divisible by x. In the other words, when asks a question about x, he receives the value of

$$\frac{\lfloor \frac{n}{x} \rfloor}{n}.$$

Important detail is that Hamilton reports the answer to Burr as an **irreducible** fraction (here $\lfloor r \rfloor$ denotes the integral part of r).

Help Burr find the answer using some certain number of queries.

Interaction format

At the beginning your solution receives a single integer g ($1 \le g \le 1000$), the number of games Hamilton and Burr are going to play.

For each test the number q ($6 \le q \le 60$) is fixed, the maximum number of queries in a single game. It is guaranteed that q queries are enough to solve the problem. This number is not available for your program, but it is mentioned in a scoring table below. If your program performs more than q queries, it receives the "Wrong answer" verdict.

In order to perform a query you should output a line "`X` t" ($1 \le t \le 10^{18}$) where t is a positive integer for which you want to know the value of

$$\frac{\lfloor \frac{n}{t} \rfloor}{n}$$

As a result for a query your program receives two integers a and b, the numerator and denominator of an irreducible fraction or number -1 in case you exceed the query limit.

If your program determines the number n, it should output a line "`N` t" where t is the supposed answer ($1 \le t \le 10^{18}$). If the answer is correct, solution receives a line "`Correct`", otherwise it receives a line "`Wrong`".

After that the next game starts (if any of them left), otherwise solution should exit.

Note that in case you read -1 or line "`Wrong`" you **must** exit your programm immediately. Otherwise the verdict for your solution may be wrong, in particular you may receive Run-time error or Time limit exceeded!

It is guaranteed that in each case the n numbers are fixed at the beginning of the game and they are not changed depending on your queries.

Examples

input	output
`2`	
	`X 2`
`1 2`	
	`X 3`
`3 10`	
	`X 5`
`1 5`	
	`X 4`
`1 5`	
	`X 6`
`1 10`	
	`X 10`
`1 10`	
	`X 11`
`0 1`	
	`N 10`
`Correct`	
	`X 1`
`1 1`	
	`X 2`
`0 1`	
	`N 1`
`Correct`	

Notes

In the first sample case $g = 2$. The example queries are listed above, using results of these queries player may understand that in the first game the answer is 10, and in the second game the answer is 1.

Strictly follow the output format. After printing anything make sure to print a new line and to flush the output buffer, in order to do that you may use `flush(output)` for Pascal/Delphi, `fflush(stdout)` or `cout.flush()` for C/C++, `sys.stdout.flush()` for Python, `System.out.flush()` for Java.

Scoring

Tests for this problem are divided into nine groups. For each of the groups you earn points only if your solution passes all tests in this group and all tests in all **previous** groups.

Group	Points	Additional constraints	Comment
		q	
0	0	$q = 60$	Sample tests
1	30	$q = 60$	
2	10	$q = 30$	
3	4	$q = 20$	
4	4	$q = 15$	
5	5	$q = 10$	
6	5	$q = 9$	
7	11	$q = 8$	
8	10	$q = 7$	
9	21	$q = 6$	