Information

Source code limit

The size of each solution source code can't exceed 256 KiB.

Testing

Notice, that each subtask has a list of required subtasks. Subtask will be tested only if all tests of all required subtasks are passed.

Scoring

We have two types of subtask scoring: "test" and "subtask".

"Test" means that points are awarded for each test in a subtask independently of other tests in this subtask.

"Subtask" means that points are awarded only if all tests in this subtask are passed.

For more information on subtask scoring read "Scoring" section of each problem.

Feedback

To get feedback for your solution, go to "Runs" tab in PCMS2 Web Client and use "View Feedback" link.

For more information, on what feedback is given, read "Feedback" section of each problem.

Problem A. Rock-Paper-Scissors for three

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

Jury of Innopolis Open really like problems about "Rock-Paper-Scissors" game. They even built two special robots to play against each other in this game. As the time passed, robots became so good at this game, so they invited Dasha to play with them.

Rules of "Rock-Paper-Scissors" for three players are as follows:

- In each round all players simultaneously show one of three followings signs: Rock, Paper or Scissors.
- Rock beats Scissors, Scissors beats Paper and Paper beats Rock.
- If one player showed the sign which beats both other players' signs, then he is the winner of the round.
- If two players showed the sign which beats the third player's sign, then they both are winners of the round.
- If all players showed either all different signs or all equal signs, then it's impossible to determine a winner and that round ends with a draw.

What robots didn't know is that Dasha programmed them and knows every move they are going to make in each round. Help Dasha show them who's boss and win as many rounds as possible.

Input format

First line of input contains n — number of rounds in Dasha's and robots' game ($1 \le n \le 100$).

Second line contains a string of length n, which consists of letters "`R`" (Rock), "`S`" (Scissors) and "`P`" (Paper), i-th letter is the first robot's move in the i-th round of the game.

Third line contains description of second robot's moves in the same format.

Output format

Output a string of n characters "`R`", "`P`" and "`S`", describing Dasha's moves in each round, that allows her to win as many rounds as possible.

Scoring

Subtask	Points	Constraints	Scoring	Required subtasks
1	100	$1 \le n \le 100$	test	—

Feedback

On request outcome for every test is shown.

Sample test

standard input	standard output
3 RPR SSR	RSP

Explanation

In the sample input, Dasha can win all three rounds. To achieve that, in the first round she chooses Rock (and wins the round together with the first robot), in the second round she chooses Scissors (and wins the round together with the second robot), in the third round she chooses Paper (winning both robots).

Problem B. Life in Innopolis

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

INNOPOLIS TIMES

December, 18, 2016

Is there life in Innopolis?

Life in Innopolis could have been brought by space body. That's a conclusion made by a scientist after examining the consistency of water taken from this area: strange DNA structure interested him. Research would take years...

To speed up the process of research, he turned to you. Let's represent DNA as a string s, consisting of four uppercase letters, one for each nucleotide: "`A`", "`C`", "`G`" and "`T`". Scientist has only one question: for how many positions i suffix starting at position i is lexicographically less than suffix starting at position $i + 1$.

Suffix is a sequence of consecutive characters, ending with the last character of the string. String itself is also its suffix. For example, `ACGC`, `CGC`, `GC`, and `C` are all suffixes of string `ACGC`.

String a is lexicographically less than string b, if there is such k that first k characters of a and b coincide and $a_{k+1} < b_{k+1}$, or if a is shorter than b and $a_i = b_i$ for all $i \le |a|$. For example, `"A"` $<$ `"G"`, `"AAG"` $<$ `"AAT"`, `"AGC"` $<$ `"AGCA"`.

Input format

Input contains a single string s, consisting of uppercase letters of Latin alphabet: "`A`", "`C`", "`G`" and "`T`".

String length doesn't exceed 3 000 000.

Output format

Output a single integer — number of positions i such that suffix starting at i is lexicographically less than suffix starting at position $i + 1$.

Scoring

Subtask	Points	Constraints	Scoring	Required subtasks
1	42	$1 \le \|s\| \le 1000$	subtask	—
2	37	$1 \le \|s\| \le 200\,000$	subtask	1
3	21	$1 \le \|s\| \le 3\,000\,000$	subtask	1 and 2

Feedback

On request outcome for each test of every subtask is shown.

Sample tests

standard input	standard output
ACGACA	3
AATTAA	2

Explanations

There are three such positions in the first example:

1. $i = 1$: "ACGACA" $<$ "CGACA"
2. $i = 2$: "CGACA" $<$ "GACA"
3. $i = 4$: "ACA" $<$ "CA"

And only two positions in the second one:

1. $i = 1$: "AATTAA" $<$ "ATTAA"
2. $i = 2$: "ATTAA" $<$ "TTAA"

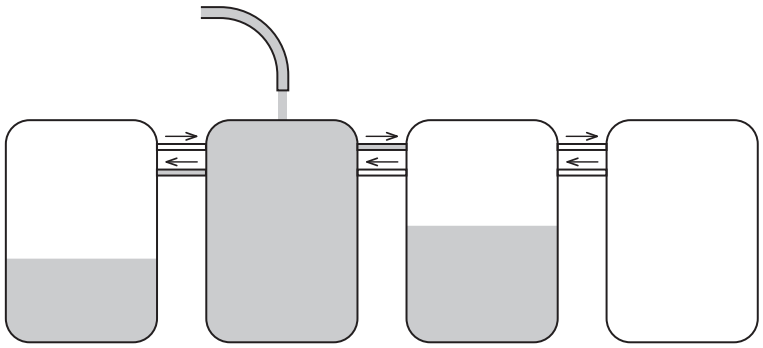
Problem C. Barrels

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

It's a little-known fact that Innopolis has a secret factory that produces secret liquid. There are n barrels locating in a row, indexed from 1 to n. Barrel i can contain a_i nanoliters of secret liquid.

Barrels are connected with directed pipes, secret liquid can flow only in one direction. For all i from 1 to $n-1$ there is a pipe from barrel i to barrel $i+1$ and a pipe from barrel $i+1$ to barrel i. This means there are two pipes between any two consecutive barrels: one for each direction. For each pipe its capacity is known: number of nanoliters of secret liquid that can pass through pipe. When that capacity is reached, liquid can no longer pass through the pipe.

Head of production department wants to choose a single barrel and put a pouring tap above this barrel. When tap is opened, this barrel is filled with secret liquid. After it's fully filled, liquid is going to pass to neighboring barrels through pipes, until either neighboring barrel is filled, or pipe capacity is reached. If pipe capacity wasn't reached, liquid can pass to next barrels, and so on. Process is finished when either all barrels are filled, or pipe capacities are reached, so that no liquid can pass further.



Help head of department to choose a starting barrel so that the total volume of secret liquid in all the barrels is as large as possible in the end.

Input format

First line contains integer n — number of barrels ($1 \le n \le 5 \cdot 10^5$).

Each of the next n lines contains three integers l_i, r_i and a_i — capacities of pipes connecting barrel i to barrel $i-1$ and $i+1$, respectively, and capacity of barrel i ($0 \le l_i, r_i, a_i \le 10^9$; $l_1 = r_n = 0$).

Output format

Output a single integer — total volume of secret liquid in all barrels.

Scoring

Subtask	Points	Constraints	Scoring	Required subtasks
1	17	$n \le 100$	subtask	—
2	29	$n \le 1000$	subtask	1
3	31	$n \le 100\,000$	subtask	1 and 2
4	23	$n \le 500\,000$	subtask	1, 2 and 3

Feedback

In subtask 1 outcome for every test is shown.

In subtasks 2, 3 and 4, points for the subtask and outcome for the first failed test are shown.

Sample tests

standard input	standard output
3 0 2 3 4 0 4 1 0 5	7
3 0 10 5 0 10 2 0 0 1	8

Explanations

In the first sample it is optimal to place the tap above the second barrel. After the second barrel is filled, liquid pours in the first barrel and fills it fully. The third barrel will remain empty, because the capacity of the pipe from barrel 2 to barrel 3 is zero.

In the second sample you should place the tap above the first barrel. After it's filled, liquid to the second barrel and to the third barrel, filling both of them.

Problem D. Wedding cake

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

Julia's wedding is going to have a huge one ton cake. All n guests want to taste the cake, so it's going to be cut in n pieces. But this task is not that easy, because all guests are on a special mathematical diet. Guest i is only willing to eat the cake if the weight of his piece in tons w_i has exactly a_i significant digits after the decimal point. In decimal representation all digits up to the last non-zero digit after the decimal point are significant. For example, number `0.007` contains three significant digits after the decimal point, number `1.45` — two, and number `17.0` has no significant digits after the decimal point.

Your task is to cut the cake for Julia's wedding so that every guest could taste it.

Input format

First line contains single integer n — number of guests ($1 \le n \le 10^5$).

Next line contains n integers a_i — constraint for the weight of i-th piece ($1 \le a_i \le 10^5$).

Sum of all a_i doesn't exceed 10^5.

Output format

Output should contain "`NO`", if there is no way to cut the cake.

Otherwise, output "`YES`" on the first line. Each of the next n lines should contain one single real number w_i — weight of the piece for the i-th guest with exactly a_i digits after the decimal point. All a_i digits after the decimal point have to be significant.

Scoring

Subtask	Points	Constraints		Scoring	Required subtasks
		n	a_i		
1	17	$n \le 100$	$a_i \le 10$	subtask	—
2	21	$n \le 10^5$	all a_i are equal	subtask	—
3	25	$n \le 10^3$	sum of all a_i doesn't exceed 10^3	subtask	1
4	37	$n \le 10^5$	sum of all a_i doesn't exceed 10^5	subtask	1, 2 and 3

Feedback

In subtask 1 outcome for every test is shown.

In subtasks 2, 3 and 4, points for the subtask and outcome for the first failed test are shown.

Sample test

standard input	standard output
5 2 4 4 3 2	YES 0.47 0.1234 0.1326 0.024 0.25

Problem E. Magical hourglass store

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

Vasya owns a magical hourglass store and he is crazy about hourglasses. Vasya wants everything in his store to work like a clock. Whenever some hourglass in his store becomes empty, Vasya immediately turns it over.

His store has n hourglasses. Hourglass i has sand for exactly b_i minutes and will magically appear in the store a_i minutes after Vasya's work day starts. Each hourglass appears empty, so Vasya has to turn it over immediately. Vasya is working for t minutes every day.

Vasya is very tired of walking around his store and turning his hourglasses over and over again. So he would like to know how many hourglasses he will have to turn over at every minute from the start of his work day.

Input format

First line contains two integers n and t — number of hourglasses in Vasya's store and number of minutes in Vasya's work day ($1 \le n, t \le 2 \cdot 10^5$).

Next n lines contain the description of Vasya's hourglasses: two integers a_i and b_i — the moment when i-th hourglass appears, and the number of minutes it works without turning it over ($1 \le a_i, b_i \le 2 \cdot 10^5$).

Output format

Output t integers, where i-th number is equal to the number of hourglasses Vasya has to turn over after i minutes pass.

Scoring

Subtask	Points	Constraints			Scoring	Required subtasks
		n	t	Additional		
1	16	$n \le 5000$	$t \le 5000$	—	subtask	—
2	23	$n \le 2 \cdot 10^5$	$t \le 5000$	—	subtask	1
3	27	$n \le 2 \cdot 10^5$	$t \le 2 \cdot 10^5$	all b_i are different	subtask	—
4	34	$n \le 2 \cdot 10^5$	$t \le 2 \cdot 10^5$	—	subtask	1, 2 and 3

Feedback

In subtask 1 outcome for every test is shown.

In subtasks 2, 3 and 4, points for the subtask and outcome for the first failed test are shown.

Sample tests

standard input	standard output
2 5 1 2 2 4	1 1 1 0 1
4 8 2 2 1 3 3 4 5 10	1 1 1 2 1 1 2 1
2 12 2 1 1 2	1 1 2 1 2 1 2 1 2 1 2 1

Explanations

In the first sample Vasya has to:

- First minute: turn the first hourglass over (first hourglass appears after 1 minute)
- Second minute: turn the second hourglass over (second hourglass appears after 2 minutes)
- Third minute: turn the first hourglass over again (2 minutes passed, so you have to turn it over again)
- Fourth minute: nothing to do (first hourglass still has 1 more minute, second one — 2 more minutes)
- Fifth minute: turn the first hourglass over (2 minutes passed, turn it over again)

In the second sample Vasya has to:

- First minute: turn the second hourglass over (second hourglass appears after 1 minute)
- Second minute: turn the first hourglass over (first hourglass appears after 2 minutes)
- Third minute: turn the third hourglass over (third hourglass appears after 3 minutes)
- Fourth minute: turn the first and the second hourglasses over
- Fifth minute: turn the fourth hourglass over (forth hourglass appears after 5 minutes)
- Sixth minute: turn the first hourglass over
- Seventh minute: turn the second and the third hourglasses over
- Eighth minute: turn the first hourglass over