Kotlin Heroes: Episode 8

A. Sequence of Comparisons

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Once upon a time, Petya had an array of integers a of length n. But over time, the array itself was lost, and only $n-1$ results of comparisons of neighboring array elements remained. In other words, for every i from 1 to $n-1$, Petya knows exactly one of these three facts:

- $a_i < a_{i+1}$;
- $a_i = a_{i+1}$;
- $a_i > a_{i+1}$.

Petya wonders if it is possible to uniquely determine the result of comparing a_1 and a_n.

You have to help Petya determine the result of comparing a_1 and a_n or report that the result cannot be determined unambiguously.

Input

The first line contains a single integer t ($1 \le t \le 500$) — the number of test cases.

The only line of the test case contains the string s ($1 \le |s| \le 100$), where s_i is:

- <, if $a_i < a_{i+1}$;
- >, if $a_i > a_{i+1}$;
- =, if $a_i = a_{i+1}$.

Output

For each test case, print a single string equal to:

- <, if $a_1 < a_n$;
- >, if $a_1 > a_n$;
- =, if $a_1 = a_n$;
- ?, if it is impossible to uniquely determine the result of the comparison.

Example

input

```
4
>>>
<><=<
=
<<==
```

output

```
>
?
=
<
```

Note

Consider the test cases of the example:

- in the first test case, it's easy to see that $a_1 > a_4$ since $a_1 > a_2 > a_3 > a_4$;
- in the second test case, both sequences $[1, 2, 0, 10, 10, 15]$ and $[10, 11, 1, 2, 2, 5]$ meet the constraints; in the first one, $a_1 < a_6$, and in the second one, $a_1 > a_6$, so it's impossible to compare a_1 and a_6;
- in the third test case, we already know that $a_1 = a_2$;
- in the fourth test case, it's easy to see that $a_3 = a_4 = a_5$, and $a_1 < a_2 < a_3$, so $a_1 < a_5$.

B. Epic Novel

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alex has bought a new novel that was published in n volumes. He has read these volumes one by one, and each volume has taken him several (maybe one) full days to read. So, on the first day, he was reading the first volume, and on each of the following days, he was reading either the same volume he had been reading on the previous day, or the next volume.

Let v_i be the number of the volume Alex was reading on the i-th day. Here are some examples:

- one of the possible situations is $v_1 = 1$, $v_2 = 1$, $v_3 = 2$, $v_4 = 3$, $v_5 = 3$ — this situation means that Alex has spent two days (1-st and 2-nd) on the first volume, one day (3-rd) on the second volume, and two days (4-th and 5-th) on the third volume;
- a situation $v_1 = 2$, $v_2 = 2$, $v_3 = 3$ is impossible, since Alex started with the first volume (so v_1 cannot be anything but 1);
- a situation $v_1 = 1$, $v_2 = 2$, $v_3 = 3$, $v_4 = 1$ is impossible, since Alex won't return to the first volume after reading the third one;
- a situation $v_1 = 1$, $v_2 = 3$ is impossible, since Alex doesn't skip volumes.

You know that Alex was reading the volume v_a on the day a, and the volume v_c on the day c. Now you want to guess which volume was he reading on the day b, which is between the days a and c (so $a < b < c$). There may be some ambiguity, so you want to make any valid guess (i. e. choose some volume number v_b so it's possible that Alex was reading volume v_a on day a, volume v_b on day b, and volume v_c on day c).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the number of volumes the novel consists of.

The second line of each test case contains two integers a and v_a ($1 \le a \le 98$; $1 \le v_a \le a$) denoting that Alex was reading volume v_a at day a.

The third line of each test case contains two integers c and v_c ($a + 2 \le c \le 100$; $v_a \le v_c \le c$) denoting that Alex was reading volume v_c at day c.

The fourth line of each test case contains one integer b ($a < b < c$) — the day you are interested in.

It's guaranteed that the input is not controversial, in other words, Alex could read volume v_a at day a and volume v_c at day c.

Output

For each test case, print the possible index of volume Alex could read at day b. If there are multiple answers, print any.

Example

input

```
4
1
1 1
100 1
99
4
10 1
20 4
16
100
1 1
100 100
42
100
1 1
100 2
99
```

output

```
1
2
42
1
```

Note

In the first test case, since Alex was reading volume 1 both at day 1 and at day 100 then he was reading volume 1 at any day between them.

In the second test case, Alex could read any volume from 1 to 4 at day 16. For example, he could read volume 1 from day 1 to day 15, volume 2 at days 16 and 17, volume 3 at day 18 and volume 4 at days 19 and 20.

In the third test case, there is only one possible situation: Alex read one volume per day, so at day 42 he read volume 42.

C. Rhyme

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's say that two strings s and t *rhyme* if both strings have length at least k, and their last k characters are equal. For example, if $k = 3$, the strings abcd and cebcd rhyme, the strings ab and ab don't rhyme, the strings aaaa and aaaaa rhyme, the strings abcd and abce don't rhyme.

You have n pairs of strings (s_i, t_i), and for each pair of strings you know, should they rhyme or should not.

Find all possible non-negative integer values for k such that pairs that have to rhyme, rhyme and pairs that must not rhyme, don't rhyme.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of string pairs.

Next n lines contains descriptions of pairs — one per line. The i-th line contains space-separated strings s_i and t_i and marker r_i. Strings are non-empty, consist of lowercase Latin letters and each have length at most $2 \cdot 10^5$. The marker r_i equals to 1 if strings have to rhyme, or 0 if they must not rhyme.

It's guaranteed that for each test case there is at least one pair with r_i equal to 1 and that the total length of all strings over all test cases doesn't exceed $4 \cdot 10^5$.

Output

For each test case, firstly print integer m — the number of possible non-negative integer values of k such that pairs that have to rhyme, rhyme and pairs that must not rhyme, don't rhyme. Next, print all these values of k (without repetitions). You can print them in any order.

Example

input

```
3
1
kotlin heroes 1
2
join kotlin 1
episode eight 0
4
abc abcdef 0
xyz zzz 1
aaa bba 0
c d 0
```

output

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

Note

In the first test case, if k is at least 1 then `kotlin` and `heroes` don't rhyme.

In the second test case, for $k = 2$ `join` and `kotlin` rhyme, and `episode` and `eight` don't rhyme.

D. Sweepstake

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Kotlin Heroes competition is nearing completion. This time n programmers took part in the competition. Now organizers are thinking how to entertain spectators as well. One of the possibilities is holding sweepstakes. So for now they decided to conduct a survey among spectators.

In total, organizers asked m viewers two questions:

1. Who will take the first place?
2. Who will take the last place?

After receiving answers, organizers ranked all spectators based on the number of programmers they guessed right. Suppose, there are c_2 viewers who guessed right both first and last place, c_1 viewers who guessed either first or last place right and c_0 viewers who didn't guess at all. All c_2 viewers will get rank 1, all viewers with one right answer will get rank $c_2 + 1$ and all remaining viewers — rank $c_2 + c_1 + 1$.

You were one of the interviewed spectators. Also, as one of the organizers, you have access to survey results, but not to competition results. Calculate, what is the worst rank you can possibly get according to organizers' ranking system?

Input

The first line contains two integers n and m ($2 \le n \le 1000$; $1 \le m \le 2 \cdot 10^5$) — the number of programmers participating in the competition and the number of surveyed spectators.

Next m lines contain answers of spectators. The i-th line contains two integers f_i and l_i ($1 \le f_i, l_i \le n$; $f_i \ne l_i$) — the indices of programmers who will take the first and last places in opinion of the i-th viewer.

For simplicity, you are the first among spectators, so your answers are f_1 and l_1.

Output

Print the single integer — the worst rank among spectators you can possibly get according to organizers' ranking system (bigger rank — worse, of course).

Examples

input

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

output

```
3
```

input

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

output

```
4
```

Note

In the first example, if the second programmer takes first place, while the first programmer takes last place, you'll have 0 right answers while the other two spectators — 2 right answers. That's why your rank (in the worst case) will be $c_2 + c_1 + 1 = 2 + 0 + 1 = 3$.

In the second example, for example, if the third programmer takes the first place and the second programmer takes the last place, then you'll have 1 right answer. The spectators 2, 4 and 5 will have 2 right answers, spectator 6 — 1 right answer and spectator 3 — 0 right answers. As a result, your rank will be equal to $c_2 + 1 = 3 + 1 = 4$. (Note that spectator 6 will have the same rank 4).

E. Fix the String

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A regular bracket sequence is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters "1" and "+" between the original characters of the sequence. For example:

- bracket sequences "()()" and "(())" are regular (the resulting expressions are: "(1)+(1)" and "((1+1)+1)");
- bracket sequences ")(", "(" and ")" are not.

You are given two strings s and a, the string s has length n, the string a has length $n-3$. The string s is a bracket sequence (i. e. each element of this string is either an opening bracket character or a closing bracket character). The string a is a binary string (i. e. each element of this string is either 1 or 0).

The string a imposes some constraints on the string s: for every i such that a_i is 1, the string $s_i s_{i+1} s_{i+2} s_{i+3}$ should be a regular bracket sequence. Characters of a equal to 0 don't impose any constraints.

Initially, the string s may or may not meet these constraints. You can perform the following operation any number of times: replace some character of s with its inverse (i. e. you can replace an opening bracket with a closing bracket, or vice versa).

Determine if it is possible to change some characters in s so that it meets all of the constraints, and if it is possible, calculate the minimum number of characters to be changed.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of three lines. The first line contains one integer n ($4 \le n \le 2 \cdot 10^5$). The second line contains the string s, consisting of exactly n characters; each character of s is either '(' or ')'. The third line contains the string a, consisting of exactly $n-3$ characters; each character of a is either '1' or '0'.

Additional constraint on the input: the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print one integer: the minimum number of characters that need to be changed in s, or -1 if it is impossible.

Example

input

```
6
4
```

```
))((
1
4
))((
0
4
()()
0
6
))(()(
101
6
))(()(
001
5
(((((
11
```

output

```
2
0
0
4
1
-1
```

F. Kotlinforces

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Kotlinforces is a web platfrom that hosts programming competitions.

The staff of Kotlinforces is asked to schedule n programming competitions on the next m days. Each competition is held in multiple stages; the regulations of the i-th competition state that this competition should consist of exactly k_i stages, and each stage, starting from the second one, should be scheduled **exactly t_i days** after the previous stage. In other words, if the first stage of the i-th competition is scheduled on day x, the second stage should be scheduled on day $x + t_i$, the third stage — on day $x + 2t_i$, ..., the k_i-th stage (which is the last one) — on day $x + (k_i - 1)t_i$.

All n competitions should be scheduled in such a way that they start and finish during the next m days, and on any of these m days, at most one stage of one competition is held (two stages of different competitions should not be scheduled on the same day).

Is it possible to schedule all n competitions to meet these constraints?

Input

The first line contains two integers n and m ($1 \le n, m \le 5000$) — the number of competitions and the number of days, respectively.

Then n lines follow, each describing a competition which should be scheduled. The i-th line contains two integers k_i and t_i ($2 \le k_i \le 5000$; $1 \le t_i \le 2$) — the parameters of the i-th competition.

Output

If it is impossible to schedule all n competitions on the next m days so that there is at most one stage during each day, print -1.

Otherwise, print n integers. The i-th integer should represent the day when the first stage of the i-th competition is scheduled; days are numbered from 1 to m. If there are multiple answers, print any of them.

Examples

input

```
3 7
3 2
2 2
2 2
```

output

```
2 5 1
```

input

```
1 7
4 2
```

output

```
1
```

input

```
1 7
5 2
```

output

```
-1
```

input

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

output

```
-1
```

input

```
2 5
2 1
2 2
```

output

```
4 1
```

G. A Battle Against a Dragon

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A squad of n warriors is defending a castle from a dragon attack. There are m barricades between the castle and the dragon.

The warriors are numbered from 1 to n. The i-th warrior knows k_i attacks: the j-th of them deals $a_{i,j}$ damage to the dragon and can only be applied if there are exactly $b_{i,j}$ barricades between the castle and the dragon.

The warriors make turns one after another, starting from warrior 1. After warrior n makes his turn, the total damage to the dragon is calculated. The i-th warrior performs exactly one of three possible moves in his turn:

1. destroys one barricade (if there are any left);
2. deploys one of his k_i attacks;
3. skips a turn.

The total damage is the sum of damages dealt by the warriors who chose to deploy their attacks in their turn. What is the maximum total damage the warriors can deal to the dragon?

Input

The first line contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) — the number of warriors and the initial number of barricades.

Then the descriptions of attacks for each warrior follow. The i-th description consists of three lines.

The first line contains a single integer k_i ($1 \le k_i \le m + 1$) — the number of attacks the i-th warrior knows.

The second line contains k_i integers $a_{i,1}, a_{i,2}, \dots, a_{i,k_i}$ ($1 \le a_{i,j} \le 10^9$) — the damage of each attack.

The third line contains k_i integers $b_{i,1}, b_{i,2}, \dots, b_{i,k_i}$ ($0 \le b_{i,j} \le m$) — the required number of barricades for each attack. $b_{i,j}$ for the i-th warrior are pairwise distinct. The attacks are listed in the increasing order of the barricades requirement, so $b_{i,1} < b_{i,2} < \dots < b_{i,k_i}$.

The sum of k_i over all warriors doesn't exceed $3 \cdot 10^5$.

Output

Print a single integer — the maximum total damage the warriors can deal to the dragon.

Examples

input

```
2 4
1
2
4
2
10 5
3 4
```

output

```
10
```

input

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

```
20
2
1
100
3
```

output

```
100
```

input

```
2 1
2
10 10
0 1
2
30 20
0 1
```

output

```
30
```

Note

In the first example, the optimal choice is the following:

- warrior 1 destroys a barricade, now there are 3 barricades left;
- warrior 2 deploys his first attack (he can do it because $b_{1,1} = 3$, which is the current number of barricades).

The total damage is 10.

If the first warrior used his attack or skipped his turn, then the second warrior would only be able to deploy his second attack. Thus, the total damage would be $2 + 5 = 7$ or 5.

In the second example, the first warrior skips his move, the second warrior skips his move and the third warrior deploys his only attack.

In the third example, two equivalent options are:

- both warriors deploy their second attacks;
- the first warrior destroys one barricade and the second warrior deploys his first attack.

Both options yield 30 total damage.

H. Laser Beams

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Ira is developing a computer game. This game features randomized generation and difficulty of levels. To achieve randomized difficulty, some enemies in each level are randomly replaced with stronger ones.

To describe how do the levels in the game look, let's introduce a coordinate system in such a way that Ox axis goes from left to right, and Oy axis goes from bottom to top. A level is a rectangle with opposite corners in points $(0, 0)$ and (a, b). Each enemy's position is a point in this rectangle.

As for now, Ira has implemented one type of enemy in the game, in two different versions — basic and upgraded. Both versions of enemies Ira has implemented fire laser rays in several directions:

- basic enemies fire four laser rays in four directions: to the right (in the same direction as the vector $(1, 0)$), to the left (in the same direction as the vector $(-1, 0)$), up (in the same direction as the vector $(0, 1)$), and down (in the same direction as the vector $(0, -1)$);
- upgraded enemies fire eight laser rays in eight directions: four directions listed for basic enemies, and four directions corresponding to vectors $(1, 1)$, $(1, -1)$, $(-1, 1)$, $(-1, -1)$.

Laser rays pass through enemies and are blocked only by the borders of the level (sides of the rectangle that denotes the level). Enemies are unaffected by lasers.

The level Ira is working on has n enemies. The i-th enemy is in the point (x_i, y_i), and it has a probability of p_i to be upgraded (it's either upgraded with probability p_i, or basic with probability $1 - p_i$). All these events are independent.

Ira wants to estimate the expected difficulty. She considers that a good way to evaluate the difficulty of the level is to count the number of parts in which the level is divided by the laser rays. So, she wants to calculate the expected number of these parts.

Help her to do the evaluation of the level!

Input

The first line contains three integers n, a and b ($1 \le n \le 100$; $2 \le a, b \le 100$) — the number of enemies in the level and the dimensions of the level.

Then n lines follow, the i-th of them contains three integers x_i, y_i and p'_i ($1 \le x_i \le a-1$; $1 \le y_i \le b-1$; $1 \le p'_i \le 999999$), meaning that the i-th enemy is located at (x_i, y_i) and has a probability of $\frac{p'_i}{10^6}$ to be upgraded.

No two enemies are located in the same point.

Output

Print one integer — the expected number of parts in which the lasers divide the level, taken modulo 998244353 (i. e. let the expected number of parts be $\frac{x}{y}$ as an irreducible fraction; you have to print $x \cdot y^{-1} \bmod 998244353$, where y^{-1} is a number such that $y \cdot y^{-1} \bmod 998244353 = 1$).

Examples

input

```
1 2 2
1 1 500000
```

output

```
6
```

input

```
2 3 2
1 1 500000
2 1 500000
```

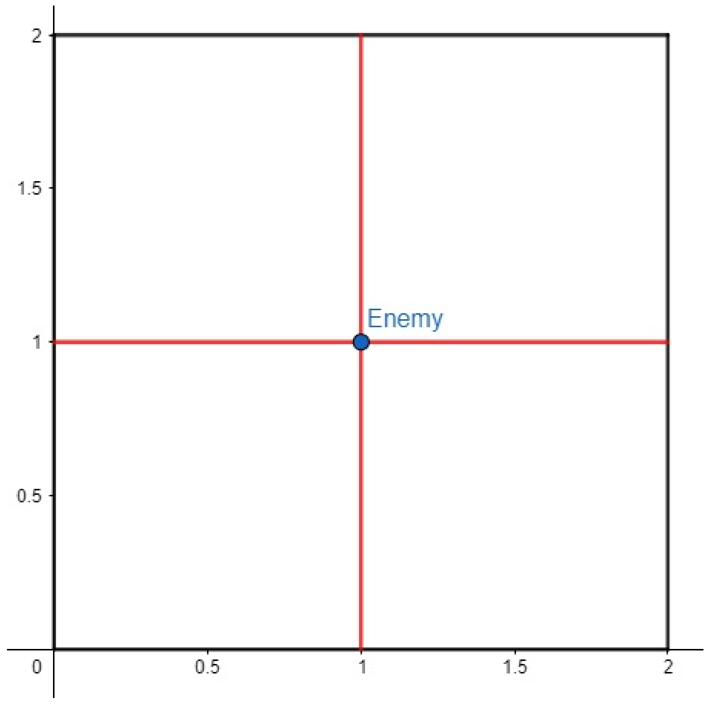
output

```
499122187
```

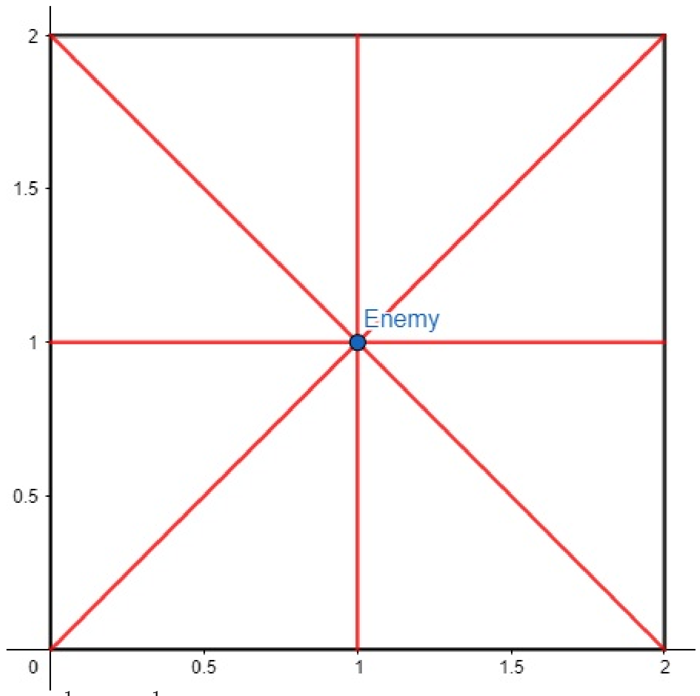
Note

Explanation to the first example:

With probability $\frac{1}{2}$, the only enemy is not upgraded and the level looks like this (4 parts):



With probability $\frac{1}{2}$, the only enemy is upgraded and the level looks like this (8 parts):



So, the expected number of parts is $4 \cdot \frac{1}{2} + 8 \cdot \frac{1}{2} = 6$.

I. Physical Examination

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp plans to undergo a full physical examination at his local clinic. There are n doctors, numbered from 1 to n. The i-th doctor takes patients from minute L_i to minute R_i, so Polycarp can visit him at any minute in this range. It takes each doctor exactly one minute to examine Polycarp's health.

Polycarp wants to arrive at the clinic at some minute x and visit all n doctors in some order **without waiting or visiting any doctor several times**.

More formally, he chooses an integer x and a permutation $p_1, p_2, \dots, p_n$ (a sequence of n integers from 1 to n such that each integer appears exactly once), then proceeds to visit:

- doctor p_1 at minute x;
- doctor p_2 at minute $x + 1$;
- ...
- doctor p_n at minute $x + n - 1$.

The p_i-th doctor should be able to take patients at minute $x + i - 1$, so the following should hold: $L[p_i] \le x + i - 1 \le R[p_i]$.

Determine if it's possible for Polycarp to choose such a minute x and a permutation p that he'll be able to visit all n doctors in without waiting or visiting any doctor several times. If there are multiple answers, print any of them.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of testcases.

Then the descriptions of t testcases follow.

The first line of the testcase contains a single integer n ($1 \le n \le 10^5$) — the number of doctors.

The second line of the testcase contains n integers $L_1, L_2, \dots L_n$ ($1 \le L_i \le 10^9$).

The third line of the testcase contains n integers $R_1, R_2, \dots R_n$ ($L_i \le R_i \le 10^9$).

The sum of n over all testcases doesn't exceed 10^5.

Output

For each testcase print an answer.

If there exists such a minute x and a permutation p that Polycarp is able to visit all n doctors without waiting or visiting any doctor several times, then print x in the first line and a permutation p in the second line. If there are multiple answers, print any of them.

Otherwise, print -1 in the only line.

Example

input

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

output

```
1
3 1 2
3
7 4 6 2 1 8 5 3
-1
-1
7
1
```

Note

In the third testcase it's impossible to visit all doctors, because Polycarp has to visit doctor 2 at minute 2 and doctor 1 at minute 4. However, that would require him to wait a minute between the visits, which is not allowed.

In the fourth testcase all doctors take patients in the span of 2 minutes. However, since there are three of them, Polycarp can't visit them all.

J. Two Railroads

time limit per test: 5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

There are n cities in Berland, and they are connected by two railroads — the Main railroad and the Auxiliary railroad. Each city has two railway stations, one connected to the Main railroad (called the Main station), and the other connected to the Auxiliary railroad.

The railroads are identical in their structure. The Main railroad consists of $n-1$ railroad segments; the i-th railroad segment connects the Main station of the city i with the Main station of the city $i+1$. Similarly, the Auxiliary railroad consists of $n-1$ railroad segments; the i-th railroad segment connects the Auxiliary station of the city i with the Auxiliary station of the city $i+1$.

These railroads are used to transfer different goods and resources from one city to another. In particular, the Ministry of Energetics is interested in using these railroads to transfer coal.

The Ministry has estimated the following capabilities of the railroads:

- for every $i \in [1, n-1]$, at most a_i tons of coal per day can be transferred from the Main station i to the Main station $i+1$ **(only in this direction)**;
- for every $i \in [1, n-1]$, at most b_i tons of coal per day can be transferred from the Auxiliary station i to the Auxiliary station $i+1$ **(only in this direction)**;
- for every $i \in [1, n]$, at most c_i tons of coal per day can be transferred from the Main station i to the Auxiliary station i, **or in the opposite direction**.

To analyze the capacity of the whole railroad network, the Ministry requires a software that would process and answer queries of the following format:

- calculate the maximum number of tons of coal that can be transferred per day from the Main station l_i to the Main station r_i.

Your task is to implement this software.

Input

The first line contains one integer n ($2 \le n \le 3 \cdot 10^5$) — the number of cities.

The second line contains $n-1$ integers $a_1, a_2, \dots, a_{n-1}$ ($1 \le a_i \le 10^9$).

The third line contains $n-1$ integers $b_1, b_2, \dots, b_{n-1}$ ($1 \le b_i \le 10^9$).

The fourth line contains n integers $c_1, c_2, \dots, c_n$ ($1 \le c_i \le 10^9$).

The fifth line contains one integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

Then q lines follow, the i-th line contains two integers l_i and r_i ($1 \le l_i < r_i \le n$) — the parameters of the i-th query.

Output

Print q integers, where the i-th integer should be the answer to the i-th query, i. e. the maximum number of tons of coal that can be transferred per day from the Main station l_i to the Main station r_i.

Example

input

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

output

```
9 8 10 9
```