Hello 2022

A. Stable Arrangement of Rooks

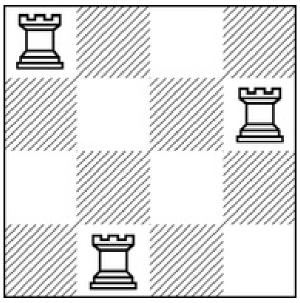
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have an $n \times n$ chessboard and k rooks. Rows of this chessboard are numbered by integers from 1 to n from top to bottom and columns of this chessboard are numbered by integers from 1 to n from left to right. The cell (x, y) is the cell on the intersection of row x and collumn y for $1 \le x \le n$ and $1 \le y \le n$.

The arrangement of rooks on this board is called *good*, if no rook is beaten by another rook.

A rook beats all the rooks that shares the same row or collumn with it.

The **good** arrangement of rooks on this board is called *not stable*, if it is possible to move one rook to the adjacent cell so arrangement becomes not good. Otherwise, the **good** arrangement is *stable*. Here, adjacent cells are the cells **that share a side**.


Such arrangement of 3 rooks on the 4×4 chessboard is good, but it is not stable: the rook from $(1, 1)$ can be moved to the adjacent cell $(2, 1)$ and rooks on cells $(2, 1)$ and $(2, 4)$ will beat each other.

Please, find any stable arrangement of k rooks on the $n \times n$ chessboard or report that there is no such arrangement.

Input

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

The first line of each test case contains two integers n, k $(1 \le k \le n \le 40)$ — the size of the chessboard and the number of rooks.

Output

If there is a stable arrangement of k rooks on the $n \times n$ chessboard, output n lines of symbols . and R. The j-th symbol of the i-th line should be equals R if and only if there is a rook on the cell (i, j) in your arrangement.

If there are multiple solutions, you may output any of them.

If there is no stable arrangement, output -1.

Example

input

```
5
3 2
3 3
1 1
5 2
40 33
```

output

```
..R
...
R..
-1
R
.....
R....
.....
....R
.....
-1
```

Note

In the first test case, you should find stable arrangement of 2 rooks on the 3×3 chessboard. Placing them in cells $(3, 1)$ and $(1, 3)$ gives stable arrangement.

In the second test case it can be shown that it is impossbile to place 3 rooks on the 3×3 chessboard to get stable arrangement.

B. Integers Shop

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

The integers shop sells n segments. The i-th of them contains all integers from l_i to r_i and costs c_i coins.

Tomorrow Vasya will go to this shop and will buy some segments there. He will get all integers that appear in at least one of bought segments. The total cost of the purchase is the sum of costs of all segments in it.

After shopping, Vasya will get some more integers as a gift. He will get integer x as a gift if and only if all of the following conditions are satisfied:

- Vasya hasn't bought x.
- Vasya has bought integer l that is less than x.
- Vasya has bought integer r that is greater than x.

Vasya can get integer x as a gift only once so he won't have the same integers after receiving a gift.

For example, if Vasya buys segment $[2, 4]$ for 20 coins and segment $[7, 8]$ for 22 coins, he spends 42 coins and receives integers $2, 3, 4, 7, 8$ from these segments. He also gets integers 5 and 6 as a gift.

Due to the technical issues only the first s segments (that is, segments $[l_1, r_1], [l_2, r_2], \ldots, [l_s, r_s]$) will be available tomorrow in the shop.

Vasya wants to get (to buy or to get as a gift) as many integers as possible. If he can do this in differents ways, he selects the cheapest of them.

For each s from 1 to n, find how many coins will Vasya spend if only the first s segments will be available.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

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

Each of next n lines contains three integers l_i, r_i, c_i ($1 \le l_i \le r_i \le 10^9, 1 \le c_i \le 10^9$) — the ends of the i-th segments and its cost.

It is guaranteed that the total sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case output n integers: the s-th ($1 \le s \le n$) of them should be the number of coins Vasia will spend in the shop if only the first s segments will be available.

Example

input

```
3
2
2 4 20
7 8 22
2
5 11 42
5 11 42
6
1 4 4
5 8 9
7 8 7
2 10 252
1 11 271
1 10 1
```

output

```
20
42
42
42
4
13
11
256
271
271
```

Note

In the first test case if $s = 1$ then Vasya can buy only the segment $[2, 4]$ for 20 coins and get 3 integers.

The way to get 7 integers for 42 coins in case $s = 2$ is described in the statement.

In the second test case note, that there can be the same segments in the shop.

C. Hidden Permutations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

The jury has a permutation p of length n and wants you to guess it. For this, the jury created another permutation q of length n. Initially, q is an identity permutation ($q_i = i$ for all i).

You can ask queries to get q_i for any i you want. After each query, the jury will change q in the following way:

- At first, the jury will create a new permutation q' of length n such that $q'_i = q_{p_i}$ for all i.
- Then the jury will replace permutation q with pemutation q'.

You can make no more than $2n$ queries in order to quess p.

Input

The first line of input contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases.

Interaction

Interaction in each test case starts after reading the single integer n ($1 \leq n \leq 10^4$) — the length of permutations p and q.

To get the value of q_i, output the query in the format $?\ i$ ($1 \leq i \leq n$). After that you will receive the value of q_i.

You can make at most $2n$ queries. After the incorrect query you will receive 0 and you should exit immediately to get `Wrong answer` verdict.

When you will be ready to determine p, output p in format $!\ p_1\ p_2\ \ldots\ p_n$. After this you should go to the next test case or exit if it was the last test case. Printing the permutation is not counted as one of $2n$ queries.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

It is guaranteed that the sum of n over all test cases doesn't exceed 10^4. The interactor is not adaptive in this problem.

Hacks:

To hack, use the following format:

The first line contains the single integer t — the number of test cases.

The first line of each test case contains the single integer n — the length of the permutations p and q. The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ — the hidden permutation for this test case.

Example

input

```
2
4

3

2

1

4

2

4

4
```

output

```
? 3

? 2

? 4

! 4 2 1 3

? 2
? 3

? 2

! 1 3 4 2
```

Note

In the first test case the hidden permutation $p = [4, 2, 1, 3]$.

Before the first query $q = [1, 2, 3, 4]$ so answer for the query will be $q_3 = 3$.

Before the second query $q = [4, 2, 1, 3]$ so answer for the query will be $q_2 = 2$.

Before the third query $q = [3, 2, 4, 1]$ so answer for the query will be $q_4 = 1$.

In the second test case the hidden permutation $p = [1, 3, 4, 2]$.

Empty strings are given only for better readability. There will be no empty lines in the testing system.

D. The Winter Hike

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Circular land is an $2n \times 2n$ grid. Rows of this grid are numbered by integers from 1 to $2n$ from top to bottom and columns of this grid are numbered by integers from 1 to $2n$ from left to right. The cell (x, y) is the cell on the intersection of row x and column y for $1 \leq x \leq 2n$ and $1 \leq y \leq 2n$.

There are n^2 of your friends in the top left corner of the grid. That is, in each cell (x, y) with $1 \leq x, y \leq n$ there is exactly one friend. Some of the other cells are covered with snow.

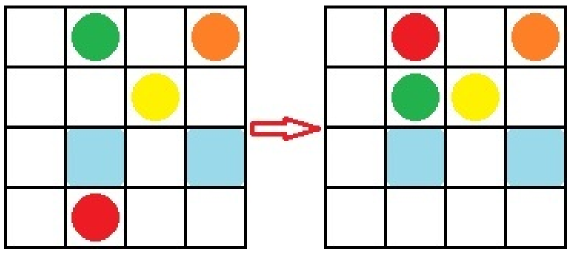
Your friends want to get to the bottom right corner of the grid. For this in each cell (x, y) with $n+1 \leq x, y \leq 2n$ there should be exactly one friend. It doesn't matter in what cell each of friends will be.

You have decided to help your friends to get to the bottom right corner of the grid.

For this, you can give instructions of the following types:

- You select a row x. All friends in this row should move to the next cell in this row. That is, friend from the cell (x, y) with $1 \leq y < 2n$ will move to the cell $(x, y+1)$ and friend from the cell $(x, 2n)$ will move to the cell $(x, 1)$.
- You select a row x. All friends in this row should move to the previous cell in this row. That is, friend from the cell (x, y) with $1 < y \leq 2n$ will move to the cell $(x, y-1)$ and friend from the cell $(x, 1)$ will move to the cell $(x, 2n)$.
- You select a column y. All friends in this column should move to the next cell in this column. That is, friend from the cell (x, y) with $1 \leq x < 2n$ will move to the cell $(x+1, y)$ and friend from the cell $(2n, y)$ will move to the cell $(1, y)$.
- You select a column y. All friends in this column should move to the previous cell in this column. That is, friend from the cell (x, y) with $1 < x \leq 2n$ will move to the cell $(x-1, y)$ and friend from the cell $(1, y)$ will move to the cell $(2n, y)$.

Note how friends on the grid border behave in these instructions.


Example of applying the third operation to the second column. Here, colorful circles denote your friends and blue cells are covered with snow.

You can give such instructions any number of times. You can give instructions of different types. If after any instruction one of your friends is in the cell covered with snow he becomes ill.

In order to save your friends you can remove snow from some cells before giving the first instruction:

- You can select the cell (x, y) that is covered with snow now and remove snow from this cell for $c_{x,y}$ coins.

You can do this operation any number of times.

You want to spend the minimal number of coins and give some instructions to your friends. After this, all your friends should be in the bottom right corner of the grid and none of them should be ill.

Please, find how many coins you will spend.

Input

The first line contains a single integer t ($1 \leq t \leq 100$) — the number of test cases.

The first line of each test case contains the single integer n ($1 \le n \le 250$).

Each of the next $2n$ lines contains $2n$ integers $c_{i,1}, c_{i,2}, \ldots, c_{i,2n}$ ($0 \le c_{i,j} \le 10^9$) — costs of removing snow from cells. If $c_{i,j} = 0$ for some i, j than there is no snow in cell (i, j). Otherwise, cell (i, j) is covered with snow.

It is guaranteed that $c_{i,j} = 0$ for $1 \le i, j \le n$.

It is guaranteed that the sum of n over all test cases doesn't exceed 250.

Output

For each test case output one integer — the minimal number of coins you should spend.

Example

input

```
4
1
0 8
1 99
2
0 0 0 0
0 0 0 0
9 9 2 2
9 9 9 9
2
0 0 4 2
0 0 2 4
4 2 4 2
2 4 2 4
4
0 0 0 0 0 0 0 2
0 0 0 0 0 0 2 0
0 0 0 0 0 2 0 0
0 0 0 0 2 0 0 0
0 0 0 2 2 0 2 2
0 0 2 0 1 6 2 1
0 2 0 0 2 4 7 4
2 0 0 0 2 0 1 6
```

output

```
100
22
14
42
```

Note

In the first test case you can remove snow from the cells $(2, 1)$ and $(2, 2)$ for 100 coins. Then you can give instructions

- All friends in the first collum should move to the previous cell. After this, your friend will be in the cell $(2, 1)$.
- All friends in the second row should move to the next cell. After this, your friend will be in the cell $(2, 2)$.

In the second test case you can remove all snow from the columns 3 and 4 for 22 coins. Then you can give instructions

- All friends in the first row should move to the next cell.
- All friends in the first row should move to the next cell.
- All friends in the second row should move to the next cell.
- All friends in the second row should move to the next cell.
- All friends in the third column should move to the next cell.
- All friends in the third column should move to the next cell.
- All friends in the fourth column should move to the next cell.
- All friends in the fourth column should move to the next cell.

It can be shown that none of the friends will become ill and that it is impossible to spend less coins.

E. New School

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

You have decided to open a new school. You have already found n teachers and m groups of students. The i-th group of students consists of $k_i \ge 2$ students. You know age of each teacher and each student. The ages of teachers are $a_1, a_2, \ldots, a_n$ and the ages of students of the i-th group are $b_{i,1}, b_{i,2}, \ldots, b_{i,k_i}$.

To *start lessons* you should assign the teacher to each group of students. Such assignment should satisfy the following requirements:

- To each group exactly one teacher assigned.
- To each teacher at most 1 group of students assigned.
- The average of students' ages in each group doesn't exceed the age of the teacher assigned to this group.

The average of set $x_1, x_2, \ldots, x_k$ of k integers is $\frac{x_1+x_2+\ldots+x_k}{k}$.

Recently you have heard that one of the students will refuse to study in your school. After this, the size of one group will decrease by 1 while all other groups will remain unchanged.

You don't know who will refuse to study. For each student determine if you can start lessons in case of his refusal.

Note, that it is **not guaranteed** that it is possible to start lessons before any refusal.

Input

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

The first line of each test case contains two integers n and m $(1 \le m \le n \le 10^5)$ — the number of teachers and the number of groups of students.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^5)$ — the ages of teachers.

The next $2m$ lines contains descriptions of groups.

The first line of description of group contains a single integer k_i $(2 \le k_i \le 10^5)$ — the number of students in this group.

The second line of description of group contains k_i integers $b_{i,1}, b_{i,2}, \ldots, b_{i,k_i}$ $(1 \le b_{i,j} \le 10^5)$ — the ages of students of this group.

It is guaranteed that the total sum of n over all test cases doesn't exceed 10^5 and that the total sum of $k_1 + k_2 + \ldots + k_m$ over all test cases doesn't exceed $2 \cdot 10^5$

Output

For each test case output string of symbols 0 and 1 of length $k_1 + k_2 + \ldots + k_m$. The i-th symbol of this string should be equals 1 if it is possible to start lessons in case of the i-th student refuse and it should be equals 0 otherwise.

Students are numbered by integers from 1 to $k_1 + k_2 + \ldots + k_m$ in the order they appear in the input. Thus, students of the 1-st group are numbered by integers 1, 2, $\ldots$, k_1, students of the 2-nd group are numbered by integers $k_1 + 1$, $k_1 + 2$, $\ldots$, $k_1 + k_2$ and so on.

Example

input

```
2
1 1
30
3
25 16 37
4 2
9 12 12 6
2
4 5
3
111 11 11
```

output

```
101
00100
```

Note

In the first test case there is one group of students with average age $\frac{25+16+37}{3} = 26$ and one teacher with age 30. There exists only one assignment that allows to start lessons.

If the student with age 16 will refuse to study, the average age of students in this group will become $\frac{25+37}{2} = 31$ so there won't be any assignment that allows to start lessons.

In the second test case it is impossible to start lessons initially. However, if the 3-rd student with age 111 will refuse to study, the average ages of groups will become $\frac{4+5}{2} = 4.5$ and $\frac{11+11}{2} = 11$ correspondingly. Then it is possible to assing the first group to the first teacher and the second group to the third teacher.

F. Strange Instructions

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

Dasha has 10^{100} coins. Recently, she found a binary string s of length n and some operations that allows to change this string (she can do each operation any number of times):

1. Replace substring `00` of s by `0` and receive a coins.
2. Replace substring `11` of s by `1` and receive b coins.
3. Remove `0` from any position in s and **pay** c coins.

It turned out that while doing this operations Dasha should follow the rule:

- It is forbidden to do two operations with the same parity in a row. Operations are numbered by integers 1-3 in the order they are given above.

Please, calculate what is the maximum profit Dasha can get by doing these operations and following this rule.

Input

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

The first line of each test case contains four integers n, a, b, c ($1 \le n \le 10^5, 1 \le a, b, c \le 10^9$).

The second line of each test case contains a binary string s of length n.

It is guaranteed that the total sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case print the answer.

Example

input

```
3
5 2 2 1
01101
6 4 3 5
110001
6 3 2 1
011110
```

output

```
3
11
4
```

Note

In the first test case one of the optimal sequences of operations is 0**11**01 → 0**1**01 → 0**11** → 01. This sequence of operations consists of operations 2, 3 and 2 in this order. It satisfies all rules and gives profit 3. It can be shown that it is impossible to achieve higher profit in this test case, so the answer is 3.

In the second test case one of the optimal sequences of operations is 110**0**01 → **11**001 → 1**00**1 → 101.

In the third test case one of the optimal sequences of operations is 01**11**10 → **0**1110 → **1**110 → 11**0** → **11** → 1.

G. Weighted Increasing Subsequences

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given the sequence of integers $a_1, a_2, \ldots, a_n$ of length n.

The sequence of indices $i_1 < i_2 < \ldots < i_k$ of length k denotes the subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ of length k of sequence a.

The subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ of length k of sequence a is called increasing subsequence if $a_{i_j} < a_{i_{j+1}}$ for each $1 \le j < k$.

The *weight* of the increasing subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ of length k of sequence a is the number of $1 \le j \le k$, such that exists index $i_k < x \le n$ and $a_x > a_{i_j}$.

For example, if $a = [6, 4, 8, 6, 5]$, then the sequence of indices $i = [2, 4]$ denotes increasing subsequence $[4, 6]$ of sequence a. The weight of this increasing subsequence is 1, because for $j = 1$ exists $x = 5$ and $a_5 = 5 > a_{i_1} = 4$, but for $j = 2$ such x doesn't exist.

Find the sum of weights of all increasing subsequences of a modulo $10^9 + 7$.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains the single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the sequence a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the sequence a.

It is guaranteed that the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print the sum of weights of all increasing subsequences a modulo $10^9 + 7$.

Example

input

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

output

```
4
12
0
6
```

Note

In the first test case the following increasing subsequences of a have not zero weight:

- The weight of $[a_1] = [6]$ is 1.
- The weight of $[a_2] = [4]$ is 1.
- The weight of $[a_2, a_3] = [4, 8]$ is 1.
- The weight of $[a_2, a_4] = [4, 6]$ is 1.

The sum of weights of increasing subsequences is 4.
In the second test case there are 7 increasing subsequences of a with not zero weight: 3 with weight 1, 3 with weight 2 and 1 with weight 3. The sum of weights is 12.

H. Trains and Airplanes

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Railway network of one city consists of n stations connected by $n-1$ roads. These stations and roads forms a tree. Station 1 is a city center. For each road you know the time trains spend to pass this road. You can assume that trains don't spend time on stops. Let's define $dist(v)$ as the time that trains spend to get from the station v to the station 1.

This railway network is splitted into zones named by first k capital latin letters. The zone of the i-th station is z_i. City center is in the zone A. For all other stations it is guaranteed that the first station on the road from this station to the city center is either in the same zone or in the zone with lexicographically smaller name. Any road is completely owned by the zone of the most distant end from the city center.

Tourist will arrive at the airport soon and then he will go to the city center. Here's how the trip from station v to station 1 happends:

- At the moment 0, tourist enters the train that follows directly from the station v to the station 1. The trip will last for $dist(v)$ minutes.
- Tourist can buy tickets for any subset of zones at any moment. Ticket for zone i costs $pass_i$ euro.
- Every T minutes since the start of the trip (that is, at the moments $T, 2T, \ldots$) the control system will scan tourist. If at the moment of scan tourist is in the zone i without zone i ticket, he should pay $fine_i$ euro. Formally, the zone of tourist is determined in the following way:
  - If tourist is at the station 1, then he already at the city center so he shouldn't pay fine.
  - If tourist is at the station $u \neq 1$, then he is in the zone z_u.
  - If tourist is moving from the station x to the station y that are directly connected by road, then he is in the zone z_x.

  Note, that tourist can pay fine multiple times in the same zone.

Tourist always selects such way to buy tickets and pay fines that minimizes the total cost of trip. Let $f(v)$ be such cost for station v.

Unfortunately, tourist doesn't know the current values of $pass_i$ and $fine_i$ for different zones and he has forgot the location of the airport. He will ask you queries of 3 types:

- $1\ i\ c$ — the cost of **ticket** in zone i has changed. Now $pass_i$ is c.
- $2\ i\ c$ — the cost of **fine** in zone i has changed. Now $fine_i$ is c.
- $3\ u$ — solve the following problem for current values of $pass$ and $fine$:
  - You are given the station u. Consider all the stations v that satisfy the following conditions:
    - $z_v = z_u$
    - The station u is on the path from the station v to the station 1.

    Find the value of $\min(f(v))$ over all such stations v with the following assumption: **tourist has the ticket for the zone of station z_u**.

Input

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

Each of the next $n-1$ lines contains three integers v_i, u_i, t_i ($1 \le v_i, u_i \le n, 1 \le t_i \le 10^9$) — the ends of the i-th road and the time it takes a train to pass this road. It is guaranteed that this roads forms a tree.

The next line contains the single integer k ($1 \le k \le 26$) — the number of zones.

The next line contains n symbols $z_1 z_2 \ldots z_n$ — z_i is the name of the zone of the i-th station. It is guaranteed that the conditions from the second paragraph are satisfied.

The next line contains k integers $pass_1$, $pass_2$, ..., $pass_k$ ($1 \le pass_i \le 10^9$) — initial costs of tickets.

The next line contains k integers $fine_1$, $fine_2$, ..., $fine_k$ ($1 \le fine_i \le 10^9$) — initial fines.

The next line contains the single integer T ($1 \le T \le 10^9$) — the time gap between scans of control system.

The next line contains the single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Next q lines contains queries as described in the statement. It is guaranteed that in the queries of the first and the second type i is a correct name of the zone (one of the first k capital latin letters) and $1 \le c \le 10^9$, and in the queries of the third type $1 \le u \le n$.

Output

For each query of the third type print the answer to it.

Example

input

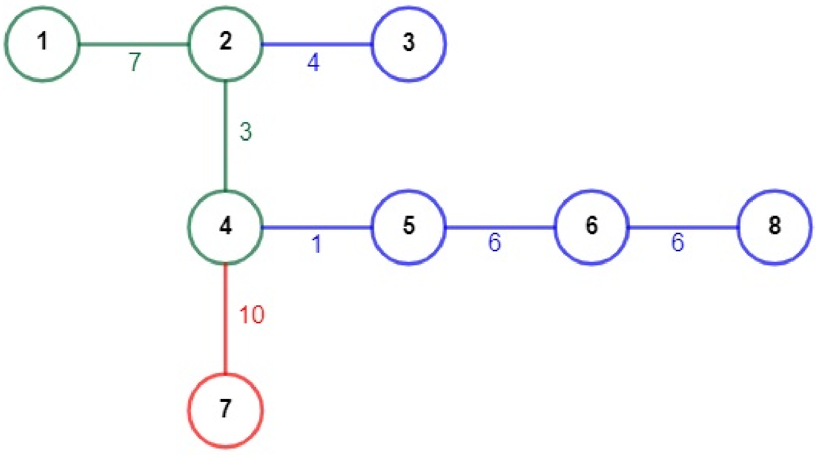
```
8
1 2 7
2 3 4
2 4 3
4 5 1
5 6 6
4 7 10
6 8 6
4
AABABBDB
11 12 10 42
16 15 15 30
4
6
3 2
1 A 10
3 3
2 A 3
3 7
3 6
```

output

```
0
10
6
6
```

Note

Note, that the fine can be cheaper than the pass.



The railway network from the example. Green color is used for stations and roads of zone A, blue color is used for zone B and red color is used for zone D. The integer near each road is time that trains spend to pass it.

In the first query, the airport can be located near the station 2 or near the station 4. During the trip, tourist will always stay in the zone A. He already has the pass for this zone so the answer is 0.

After the second query, the cost of the pass in the zone A has become 10.

In the third query, the airport can be located only near the station 3. Optimal solution will be to buy the pass for zone A. During the first 3 seconds of trip tourist will be in the zone B. Then he will move to the zone A and will be scanned there on the 4-th and the 8-th second of his ride. Since he have a pass for this zone, he won't pay fines.

After the forth query, the fine in the zone A has become 3.

In the fifth query, the airport can be located only near the station 7 and $f(7) = 6$.

In the sixth query, the airport can be located near the station 6 or near the station 8. Since $f(6) = 9$ and $f(8) = 6$ the answer is 6.

I. Two Sequences

time limit per test: 8 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Consider an array of integers $C = [c_1, c_2, \ldots, c_n]$ of length n. Let's build the sequence of arrays $D_0, D_1, D_2, \ldots, D_n$ of length $n+1$ in the following way:

- The first element of this sequence will be equals C: $D_0 = C$.
- For each $1 \leq i \leq n$ array D_i will be constructed from D_{i-1} in the following way:
    - Let's find the lexicographically smallest subarray of D_{i-1} of length i. Then, the first $n-i$ elements of D_i will be equals to the corresponding $n-i$ elements of array D_{i-1} and the last i elements of D_i will be equals to the corresponding elements of the found subarray of length i.

Array x is subarray of array y, if x can be obtained by deletion of several (possibly, zero or all) elements from the beginning of y and several (possibly, zero or all) elements from the end of y.

For array C let's denote array D_n as $op(C)$.

Alice has an array of integers $A = [a_1, a_2, \ldots, a_n]$ of length n. She will build the sequence of arrays $B_0, B_1, \ldots, B_n$ of length $n+1$ in the following way:

- The first element of this sequence will be equals A: $B_0 = A$.
- For each $1 \leq i \leq n$ array B_i will be equals $op(B_{i-1})$, where op is the transformation described above.

She will ask you q queries about elements of sequence of arrays $B_0, B_1, \ldots, B_n$. Each query consists of two integers i and j, and the answer to this query is the value of the j-th element of array B_i.

Input

The first line contains the single integer n ($1 \leq n \leq 10^5$) — the length of array A.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the array A.

The third line contains the single integer q ($1 \leq q \leq 10^6$) — the number of queries.

Each of the next q lines contains two integers i, j ($1 \leq i, j \leq n$) — parameters of queries.

Output

Output q integers: values of $B_{i,j}$ for required i, j.

Example

input

```
4
2 1 3 1
4
1 1
1 2
1 3
1 4
```

output

```
2
1
1
3
```

Note

In the first test case $B_0 = A = [2, 1, 3, 1]$.

B_1 is constructed in the following way:

- Initially, $D_0 = [2, 1, 3, 1]$.
- For $i = 1$ the lexicographically smallest subarray of D_0 of length 1 is $[1]$, so D_1 will be $[2, 1, 3, 1]$.

- For $i = 2$ the lexicographically smallest subarray of D_1 of length 2 is $[1, 3]$, so D_2 will be $[2, 1, 1, 3]$.
- For $i = 3$ the lexicographically smallest subarray of D_2 of length 3 is $[1, 1, 3]$, so D_3 will be $[2, 1, 1, 3]$.
- For $i = 4$ the lexicographically smallest subarray of D_3 of length 4 is $[2, 1, 1, 3]$, so D_4 will be $[2, 1, 1, 3]$.
- So, $B_1 = op(B_0) = op([2, 1, 3, 1]) = [2, 1, 1, 3]$.