A. Sakurako and Kosuke

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

Sakurako and Kosuke decided to play some games with a dot on a coordinate line. The dot is currently located in position $x = 0$. They will be taking turns, and **Sakurako will be the one to start**.

On the i-th move, the current player will move the dot in some direction by $2 \cdot i - 1$ units. Sakurako will always be moving the dot in the negative direction, whereas Kosuke will always move it in the positive direction.

In other words, the following will happen:

1. Sakurako will change the position of the dot by -1, $x = -1$ now
2. Kosuke will change the position of the dot by 3, $x = 2$ now
3. Sakurako will change the position of the dot by -5, $x = -3$ now
4. $\cdots$

They will keep on playing while the absolute value of the coordinate of the dot does not exceed n. More formally, the game continues while $-n \le x \le n$. It can be proven that the game will always end.

Your task is to determine who will be the one who makes the last turn.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of games that Sakurako and Kosuke played.

Each game is described by one number n ($1 \le n \le 100$) — the number that defines the condition when the game ends.

Output

For each of the t games, output a line with the result of that game. If Sakurako makes the last turn, output "`Sakurako`" (without quotes); else output "`Kosuke`".

Standard Input	Standard Output
4 1 6 3 98	Kosuke Sakurako Kosuke Sakurako

,

B. Sakurako and Water

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

During her journey with Kosuke, Sakurako and Kosuke found a valley that can be represented as a matrix of size $n \times n$, where at the intersection of the i-th row and the j-th column is a mountain with a height of $a_{i,j}$. If $a_{i,j} < 0$, then there is a lake there.

Kosuke is very afraid of water, so Sakurako needs to help him:

- With her magic, she can select a square area of mountains and increase the height of each mountain on the main diagonal of that area by exactly one.

More formally, she can choose a submatrix with the upper left corner located at (i, j) and the lower right corner at (p, q), such that $p - i = q - j$. She can then add one to each element at the intersection of the $(i + k)$-th row and the $(j + k)$-th column, for all k such that $0 \le k \le p - i$.

Determine the minimum number of times Sakurako must use her magic so that there are no lakes.

Input

The first line contains a single integer t ($1 \le t \le 200$) — the number of test cases.

Each test case is described as follows:

- The first line of each test case consists of a single number n ($1 \le n \le 500$).
- Each of the following n lines consists of n integers separated by spaces, which correspond to the heights of the mountains in the valley a ($-10^5 \le a_{i,j} \le 10^5$).

It is guaranteed that the sum of n across all test cases does not exceed 1000.

Output

For each test case, output the minimum number of times Sakurako will have to use her magic so that all lakes disappear.

Standard Input	Standard Output
4 1 1 2 -1 2 3 0 3 1 2 3 -2 1 -1 0 0 -1 5 1 1 -1 -1 3 -3 1 4 4 -4 -1 -1 3 0 -5	0 1 4 19

4 5 3 -3 -1 3 1 -3 -1 5	

C. Sakurako's Field Trip

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Even in university, students need to relax. That is why Sakurakos teacher decided to go on a field trip. It is known that all of the students will be walking in one line. The student with index i has some topic of interest which is described as a_i. As a teacher, you want to minimise the *disturbance* of the line of students.

The *disturbance* of the line is defined as the number of neighbouring people with the same topic of interest. In other words, *disturbance* is the number of indices j ($1 \le j < n$) such that $a_j = a_{j+1}$.

In order to do this, you can choose index i ($1 \le i \le n$) and swap students at positions i and $n - i + 1$. You can perform any number of swaps.

Your task is to determine the minimal amount of *disturbance* that you can achieve by doing the operation described above any number of times.

Input

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

Each test case is described by two lines.

- The first line contains one integer n ($2 \le n \le 10^5$) — the length of the line of students.
- The second line contains n integers a_i ($1 \le a_i \le n$) — the topics of interest of students in line.

It is guaranteed that the sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the minimal possible *disturbance* of the line that you can achieve.

Standard Input	Standard Output
9 5 1 1 1 2 3 6 2 1 2 2 1 1 4 1 2 1 1 6 2 1 1 2 2 4 4 2 1 2 3 6 1 2 2 1 2 1 5 4 5 5 1 5 7 1 4 3 5 1 1 3	1 2 1 0 0 1 1 0 2

7 3 1 3 2 2 3 3	

Note

In the first example, it is necessary to apply the operation to $i = 2$, thus the array will become $[1, \mathbf{2}, 1, \mathbf{1}, 3]$, with the bold elements indicating those that have swapped places. The *disturbance* of this array is equal to 1.

In the fourth example, it is sufficient to apply the operation to $i = 3$, thus the array will become $[2, 1, \mathbf{2}, \mathbf{1}, 2, 4]$. The *disturbance* of this array is equal to 0.

In the eighth example, it is sufficient to apply the operation to $i = 3$, thus the array will become $[1, 4, \mathbf{1}, 5, \mathbf{3}, 1, 3]$. The *disturbance* of this array is equal to 0.

D. Kousuke's Assignment

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

After a trip with Sakurako, Kousuke was very scared because he forgot about his programming assignment. In this assignment, the teacher gave him an array a of n integers and asked him to calculate the number of **non-overlapping** segments of the array a, such that each segment is considered *beautiful*.

A segment $[l, r]$ is considered *beautiful* if $a_l + a_{l+1} + \dots + a_{r-1} + a_r = 0$.

For a fixed array a, your task is to compute the maximum number of non-overlapping *beautiful* segments.

Input

The first line of input contains the number t ($1 \le t \le 10^4$) — the number of test cases. Each test case consists of 2 lines.

- The first line contains one integer n ($1 \le n \le 10^5$) — the length of the array.
- The second line contains n integers a_i ($-10^5 \le a_i \le 10^5$) — the elements of the array a.

It is guaranteed that the sum of n across all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output a single integer: the maximum number of non-overlapping *beautiful* segments.

Standard Input	Standard Output
3 5 2 1 -3 2 1 7 12 -4 4 43 -3 -5 8 6 0 -4 0 3 0 1	1 2 3

E. Sakurako, Kosuke, and the Permutation

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Sakurako's exams are over, and she did excellently. As a reward, she received a permutation p. Kosuke was not entirely satisfied because he failed one exam and did not receive a gift. He decided to sneak into her room (thanks to the code for her lock) and spoil the permutation so that it becomes *simple*.

A permutation p is considered *simple* if for every i $(1 \le i \le n)$ one of the following conditions holds:

- $p_i = i$
- $p_{p_i} = i$

For example, the permutations $[1, 2, 3, 4]$, $[5, 2, 4, 3, 1]$, and $[2, 1]$ are *simple*, while $[2, 3, 1]$ and $[5, 2, 1, 4, 3]$ are not.

In one operation, Kosuke can choose indices i, j $(1 \le i, j \le n)$ and swap the elements p_i and p_j.

Sakurako is about to return home. Your task is to calculate the minimum number of operations that Kosuke needs to perform to make the permutation *simple*.

Input

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

Each test case is described by two lines.

- The first line contains one integer n ($1 \le n \le 10^6$) — the length of the permutation p.
- The second line contains n integers p_i ($1 \le p_i \le n$) — the elements of the permutation p.

It is guaranteed that the sum of n across all test cases does not exceed 10^6.

It is guaranteed that p is a permutation.

Output

For each test case, output the minimum number of operations that Kosuke needs to perform to make the permutation *simple*.

Standard Input	Standard Output
6 5 1 2 3 4 5 5 5 4 3 2 1 5 2 3 4 5 1 4 2 3 4 1 3 1 3 2	0 0 2 1 0 2

7 2 3 1 5 6 7 4	

Note

In the first and second examples, the permutations are already *simple*.

In the fourth example, it is sufficient to swap p_2 and p_4. Thus, the permutation will become $[2, 1, 4, 3]$ in 1 operation.

,

F. Kosuke's Sloth

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

Kosuke is too lazy. He will not give you any legend, just the task:

Fibonacci numbers are defined as follows:

- $f(1) = f(2) = 1$.
- $f(n) = f(n-1) + f(n-2)$ $(3 \leq n)$

We denote $G(n, k)$ as an index of the n-th Fibonacci number that is divisible by k. For given n and k, compute $G(n, k)$.
As this number can be too big, output it by modulo $10^9 + 7$.

For example: $G(3, 2) = 9$ because the 3-rd Fibonacci number that is divisible by 2 is 34. $[1, 1, \mathbf{2}, 3, 5, \mathbf{8}, 13, 21, \mathbf{34}]$.

Input

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

The first and only line contains two integers n and k $(1 \leq n \leq 10^{18}$, $1 \leq k \leq 10^5)$.

It is guaranteed that the sum of k across all test cases does not exceed 10^6.

Output

For each test case, output the only number: the value $G(n, k)$ taken by modulo $10^9 + 7$.

Standard Input	Standard Output
3 3 2 100 1 1000000000000 1377	9 100 999244007

G. Sakurako and Chefir

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   256 megabytes
```

Given a tree with n vertices rooted at vertex 1. While walking through it with her cat Chefir, Sakurako got distracted, and Chefir ran away.

To help Sakurako, Kosuke recorded his q guesses. In the i-th guess, he assumes that Chefir got lost at vertex v_i and had k_i *stamina*.

Also, for each guess, Kosuke assumes that Chefir could move along the edges an arbitrary number of times:

- from vertex a to vertex b, if a **is an ancestor*** of b, the *stamina* will not change;
- from vertex a to vertex b, if a **is not an ancestor** of b, then Chefir's *stamina* decreases by 1.

If Chefir's *stamina* is 0, he cannot make a move of the second type.

For each assumption, your task is to find the distance to the farthest vertex that Chefir could reach from vertex v_i, having k_i *stamina*.

*Vertex a is an ancestor of vertex b if the shortest path from b to the root passes through a.

Input

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

Each test case is described as follows:

- The first line contains a single integer n $(2 \le n \le 2 \cdot 10^5)$ — the number of vertices in the tree.
- The next $n-1$ lines contain the edges of the tree. It is guaranteed that the given edges form a tree.
- The next line consists of a single integer q $(1 \le q \le 2 \cdot 10^5)$, which denotes the number of guesses made by Kosuke.
- The next q lines describe the guesses made by Kosuke, with two integers v_i, k_i $(1 \le v_i \le n, 0 \le k_i \le n)$.

It is guaranteed that the sum of n and the sum of q across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case and for each guess, output the maximum distance to the farthest vertex that Chefir could reach from the starting point v_i having k_i *stamina*.

Standard Input	Standard Output
3 5 1 2 2 3 3 4 3 5 3 5 1	2 1 2 0 5 2 4 5 5 5 1 3 4

3 1 2 0 9 8 1 1 7 1 4 7 3 4 9 3 2 1 5 3 6 7 6 0 2 3 6 2 8 2 2 4 9 2 6 3 6 2 1 2 5 2 4 5 6 4 3 3 3 1 1 3 6 5	

Note

In the first example:

- In the first query, you can go from vertex 5 to vertex 3 (after which your stamina will decrease by 1 and become 0), and then you can go to vertex 4;
- In the second query, from vertex 3 with 1 stamina, you can only reach vertices 2, 3, 4, and 5;
- In the third query, from vertex 2 with 0 stamina, you can only reach vertices 2, 3, 4, and 5;