Disco CodeJam Challenge 1

A. Log Chopping

2.5 s, 256 megabytes

There are n logs, the i-th log has a length of a_i meters. Since chopping logs is tiring work, errorgorn and maomao90 have decided to play a game.

errorgorn and maomao90 will take turns chopping the logs with **errorgorn chopping first**. On his turn, the player will pick a log and chop it into 2 pieces. If the length of the chosen log is x, and the lengths of the resulting pieces are y and z, then y and z have to be **positive integers**, and $x = y + z$ must hold. For example, you can chop a log of length 3 into logs of lengths 2 and 1, but not into logs of lengths 3 and 0, 2 and 2, or 1.5 and 1.5.

The player who is unable to make a chop will be the loser. Assuming that both errorgorn and maomao90 play optimally, who will be the winner?

Input

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

The first line of each test case contains a single integer n $(1 \le n \le 50)$ — the number of logs.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 50$) — the lengths of the logs.

Note that there is no bound on the sum of n over all test cases.

Output

For each test case, print "errorgorn" if errorgorn wins or "maomao90" if maomao90 wins. (Output without quotes).

input

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

output

```
errorgorn
maomao90
```

In the first test case, errorgorn will be the winner. An optimal move is to chop the log of length 4 into 2 logs of length 2. After this there will only be 4 logs of length 2 and 1 log of length 1.

After this, the only move any player can do is to chop any log of length 2 into 2 logs of length 1. After 4 moves, it will be maomao90's turn and he will not be able to make a move. Therefore errorgorn will be the winner.

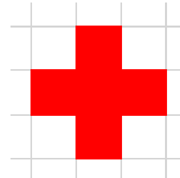
In the second test case, errorgorn will not be able to make a move on his first turn and will immediately lose, making maomao90 the winner.

B. Fox and Cross

2.5 s, 256 megabytes

Fox Ciel has a board with n rows and n columns. So, the board consists of $n \times n$ cells. Each cell contains either a symbol '.', or a symbol '#'.

A cross on the board is a connected set of exactly five cells of the board that looks like a cross. The picture below shows how it looks.



Ciel wants to draw several (may be zero) crosses on the board. Each cross must cover exactly five cells with symbols '#', and any cell with symbol '#' must belong to some cross. No two crosses can share a cell.

Please, tell Ciel if she can draw the crosses in the described way.

Input

The first line contains an integer n $(3 \le n \le 100)$ — the size of the board.

Each of the next n lines describes one row of the board. The i-th line describes the i-th row of the board and consists of n characters. Each character is either a symbol '.', or a symbol '#'.

Output

Output a single line with "YES" if Ciel can draw the crosses in the described way. Otherwise output a single line with "NO".

input

```
5
.#...
####.
.####
...#.
.....
```

output

```
YES
```

input

```
4
####
####
####
####
```

output

```
NO
```

input

```
6
.#....
####..
.####.
.#.##.
######
.#..#.
```

output

```
YES
```

input

```
6
.#..#.
######
.####.
.####.
######
.#..#.
```

output

```
NO
```

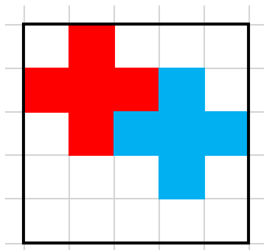
input

```
3
...
...
...
```

output

```
YES
```

In example 1, you can draw two crosses. The picture below shows what they look like.



In example 2, the board contains 16 cells with '#', but each cross contains 5. Since 16 is not a multiple of 5, so it's impossible to cover all.

C. Coloring a Tree

2.5 s, 256 megabytes

You are given a rooted tree with n vertices. The vertices are numbered from 1 to n, the root is the vertex number 1.

Each vertex has a color, let's denote the color of vertex v by c_v. Initially $c_v = 0$.

You have to color the tree into the given colors using the smallest possible number of steps. On each step you can choose a vertex v and a color x, and then color all vectices in the subtree of v (including v itself) in color x. In other words, for every vertex u, such that the path from root to u passes through v, set $c_u = x$.

It is guaranteed that you have to color each vertex in a color different from 0.

You can learn what a rooted tree is using the link: https://en.wikipedia.org/wiki/Tree_(graph_theory).

Input

The first line contains a single integer n ($2 \le n \le 10^4$) — the number of vertices in the tree.

The second line contains n - 1 integers $p_2, p_3, ..., p_n$ ($1 \le p_i < i$), where p_i means that there is an edge between vertices i and p_i.

The third line contains n integers $c_1, c_2, ..., c_n$ ($1 \le c_i \le n$), where c_i is the color you should color the i-th vertex into.

It is guaranteed that the given graph is a tree.

Output

Print a single integer — the minimum number of steps you have to perform to color the tree into given colors.

input

```
6
1 2 2 1 5
2 1 1 1 1 1
```

output

```
3
```

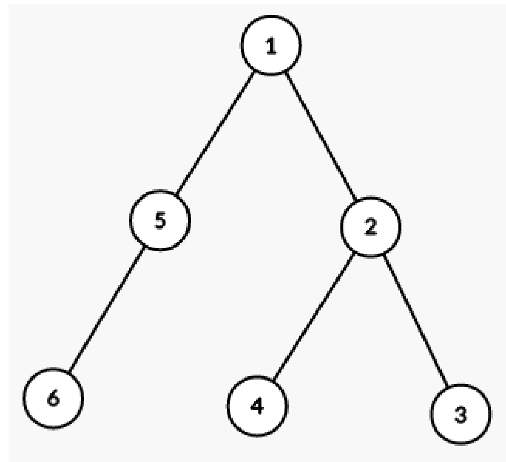
input

```
7
1 1 2 3 1 4
3 3 1 1 1 2 3
```

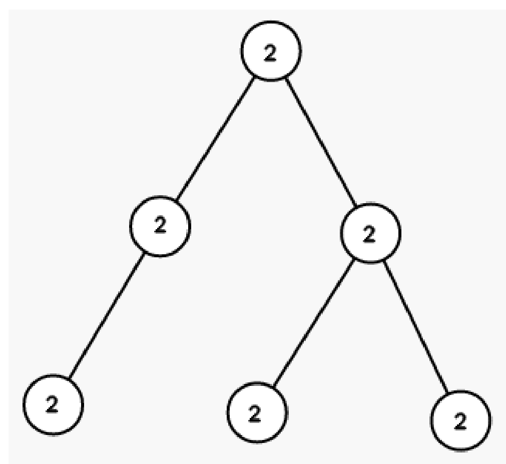
output

```
5
```

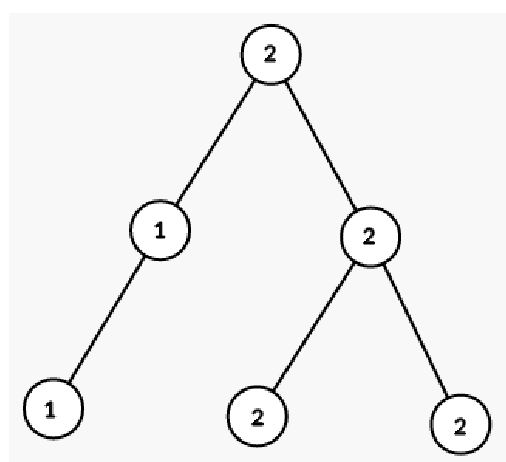
The tree from the first sample is shown on the picture (numbers are vetices' indices):



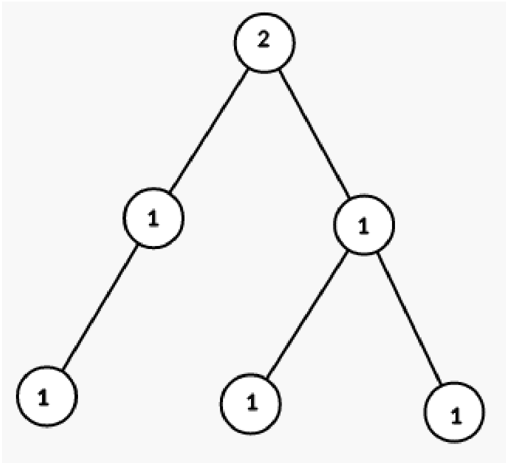
On first step we color all vertices in the subtree of vertex 1 into color 2 (numbers are colors):



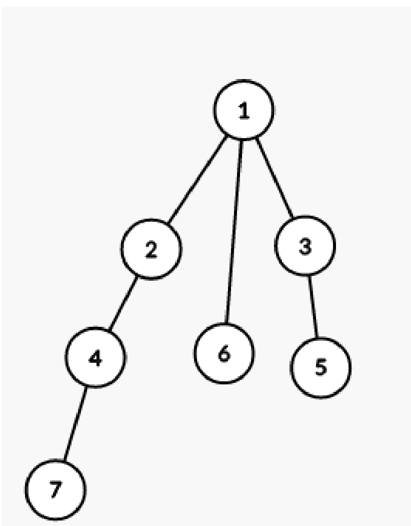
On seond step we color all vertices in the subtree of vertex 5 into color 1:



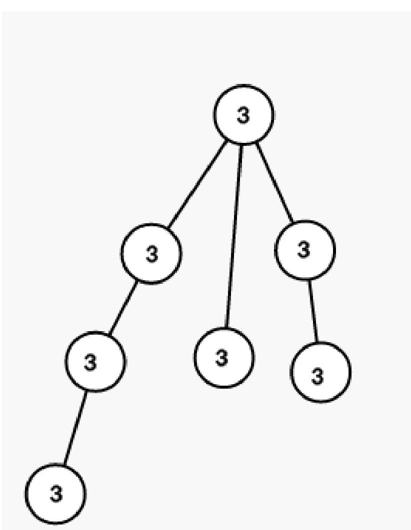
On third step we color all vertices in the subtree of vertex 2 into color 1:



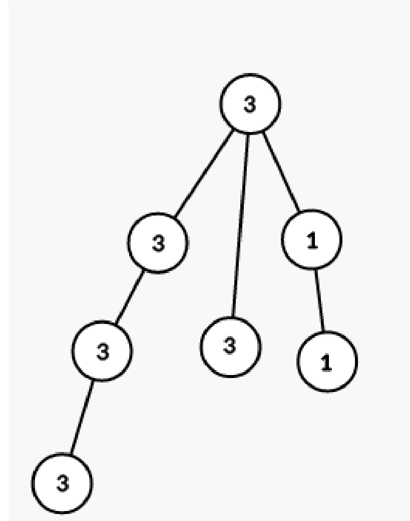
The tree from the second sample is shown on the picture (numbers are vetices' indices):



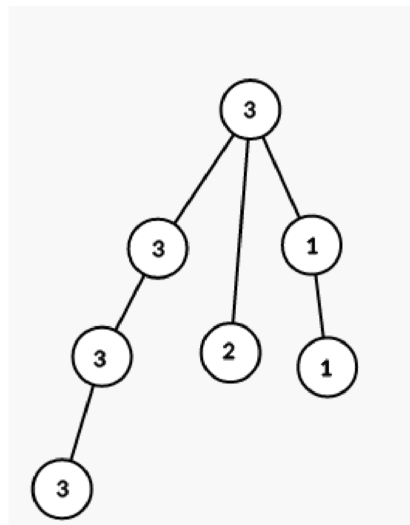
On first step we color all vertices in the subtree of vertex 1 into color 3 (numbers are colors):



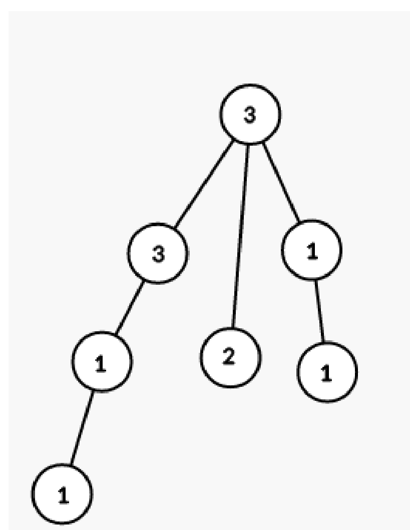
On second step we color all vertices in the subtree of vertex 3 into color 1:



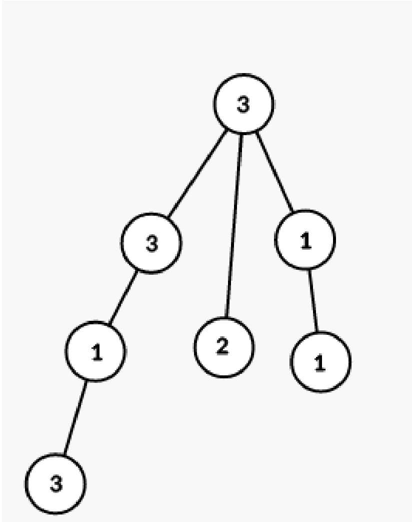
On third step we color all vertices in the subtree of vertex 6 into color 2:



On fourth step we color all vertices in the subtree of vertex 4 into color 1:



On fith step we color all vertices in the subtree of vertex 7 into color 3:



D. NIT Destroys the Universe

4.0 s, 512 megabytes

For a collection of integers S, define $\operatorname{mex}(S)$ as the smallest non-negative integer that does not appear in S.

NIT, the cleaver, decides to destroy the universe. He is not so powerful as Thanos, so he can only destroy the universe by snapping his fingers several times.

The universe can be represented as a 1-indexed array a of length n. When NIT snaps his fingers, he does the following operation on the array:

- He selects positive integers l and r such that $1 \le l \le r \le n$. Let $w = \operatorname{mex}(\{a_l, a_{l+1}, \ldots, a_r\})$. Then, for all $l \le i \le r$, set a_i to w.

We say the universe is destroyed if and only if for all $1 \le i \le n$, $a_i = 0$ holds.

Find the minimum number of times NIT needs to snap his fingers to destroy the universe. That is, find the minimum number of operations NIT needs to perform to make all elements in the array equal to 0.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains one integer n ($1 \le n \le 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

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

Output

For each test case, print one integer — the answer to the problem.

input
```
4
4
0 0 0 0
5
0 1 2 3 4
7
0 2 3 0 1 2 0
1
1000000000
```

output
```
0
1
2
1
```

In the first test case, we do 0 operations and all elements in the array are already equal to 0.

In the second test case, one optimal way is doing the operation with $l=2, r=5$.

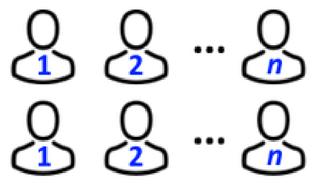
In the third test case, one optimal way is doing the operation twice, respectively with $l=4, r=4$ and $l=2, r=6$.

In the fourth test case, one optimal way is doing the operation with $l=1$, $r=1$.

E. Basketball Exercise

4.0 s, 512 MB

Finally, a basketball court has been opened in SIS, so Demid has decided to hold a basketball exercise session. $2 \cdot n$ students have come to Demid's exercise session, and he lined up them into two rows of the same size (there are exactly n people in each row). Students are numbered from 1 to n in each row in order from left to right.



Now Demid wants to choose a team to play basketball. He will choose players from left to right, and the index of each chosen player (excluding the first one **taken**) will be strictly greater than the index of the previously chosen player. To avoid giving preference to one of the rows, Demid chooses students in such a way that no consecutive chosen students belong to the same row. The first student can be chosen among all $2n$ students (there are no additional constraints), and a team can consist of any number of students.

Demid thinks, that in order to compose a perfect team, he should choose students in such a way, that the total height of all chosen students is maximum possible. Help Demid to find the maximum possible total height of players in a team he can choose.

Input

The first line of the input contains a single integer n ($1 \le n \le 10^5$) — the number of students in each row.

The second line of the input contains n integers $h_{1,1}, h_{1,2}, \ldots, h_{1,n}$ ($1 \le h_{1,i} \le 10^9$), where $h_{1,i}$ is the height of the i-th student in the first row.

The third line of the input contains n integers $h_{2,1}, h_{2,2}, \ldots, h_{2,n}$ ($1 \le h_{2,i} \le 10^9$), where $h_{2,i}$ is the height of the i-th student in the second row.

Output

Print a single integer — the maximum possible total height of players in a team Demid can choose.

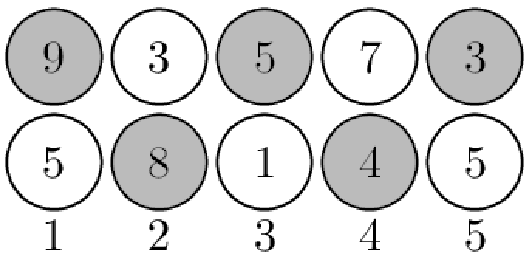
input
```
5
9 3 5 7 3
5 8 1 4 5
```

output
```
29
```

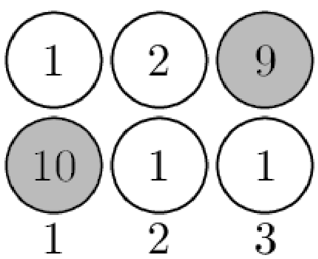
input
```
3
1 2 9
10 1 1
```
output
```
19
```

input
```
1
7
4
```
output
```
7
```

In the first example Demid can choose the following team as follows:



In the second example Demid can choose the following team as follows:



F. Collective Mindsets (hard)

6.0 s, 256 megabytes

Heidi got one brain, thumbs up! But the evening isn't over yet and one more challenge awaits our dauntless agent: after dinner, at precisely midnight, the N attendees love to play a very risky game...

Every zombie gets a number n_i ($1 \leq n_i \leq N$) written on his forehead. Although no zombie can see his own number, he can see the numbers written on the foreheads of all N - 1 fellows. Note that not all numbers have to be unique (they can even all be the same). From this point on, no more communication between zombies is allowed. Observation is the only key to success. When the cuckoo clock strikes midnight, all attendees have to simultaneously guess the number on their own forehead. If at least one of them guesses his number correctly, all zombies survive and go home happily. On the other hand, if not a single attendee manages to guess his number correctly, all of them are doomed to die!

Zombies aren't very bright creatures though, and Heidi has to act fast if she does not want to jeopardize her life. She has one single option: by performing some quick surgery on the brain she managed to get from the chest, she has the ability to remotely reprogram the decision-making strategy of all attendees for their upcoming midnight game! Can you suggest a sound strategy to Heidi which, given the rules of the game, ensures that at least one attendee will guess his own number correctly, for any possible sequence of numbers on the foreheads?

Given a zombie's rank R and the N - 1 numbers n_i on the other attendees' foreheads, your program will have to return the number that the zombie of rank R shall guess. Those answers define your strategy, and we will check if it is flawless or not.

Input

The first line of input contains a single integer T ($1 \leq T \leq 50000$): the number of scenarios for which you have to make a guess.

The T scenarios follow, described on two lines each:

- The first line holds two integers, N ($2 \leq N \leq 6$), the number of attendees, and R ($1 \leq R \leq N$), the rank of the zombie who has to make the guess.
- The second line lists N - 1 integers: the numbers on the foreheads of all other attendees, listed in increasing order of the attendees' rank. (Every zombie knows the rank of every other zombie.)

Output

For every scenario, output a single integer: the number that the zombie of rank R shall guess, based on the numbers n_i on his N - 1 fellows' foreheads.

input
```
4
2 1
1
2 2
1
2 1
2
2 2
2
```
output
```
1
2
2
1
```

input
```
2
5 2
2 2 2 2
6 4
3 2 6 1 2
```
output
```
5
2
```

For instance, if there were $N = 2$ two attendees, a successful strategy could be:

- The zombie of rank 1 always guesses the number he sees on the forehead of the zombie of rank 2.
- The zombie of rank 2 always guesses the opposite of the number he sees on the forehead of the zombie of rank 1.