Alien Piano

An alien has just landed on Earth, and really likes our music. Lucky for us.

The alien would like to bring home its favourite human songs, but it only has a very strange instrument to do it with: a piano with just 4 keys of different pitches.

The alien converts a song by writing it down as a series of keys on the alien piano. Obviously, this piano will not be able to convert our songs completely, as our songs tend to have many more than 4 pitches.

The alien will settle for converting our songs with the following rules instead:

- The first note in our song can be converted to any key on the alien piano.
- For every note after,
    - if its pitch is higher than the previous note, it should be converted into a higher-pitched key than the previous note's conversion;
    - if lower, it should be converted into a lower-pitched key than the previous note's conversion;
    - if exactly identical, it should be converted into the same key as the previous note's conversion.

Note: two notes with the same pitch do not need to be converted into the same key if they are not adjacent.

What the alien wants to know is: how often will it have to break its rules when converting a particular song?

To elaborate, let us describe one of our songs as having **K** notes. The first note we describe as "note 1", the second note "note 2", and the last note "note **K**." So note 2 comes immediately after note 1. Now if note 2 is lower than note 1 in our version of the song, yet converted to an equally-pitched or lower-pitched key (relative to note 2's conversion) in the alien's version of the song, then we consider that a single rule break.
For each test case, return the minimum amount of times the alien must necessarily break one of its rules in converting that song.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.
Each test case consists of two lines.
The first line consists of a single integer, **K**.
The second line consists of **K** space-separated integers, $\mathbf{A_1}$, $\mathbf{A_2}$ ... $\mathbf{A_K}$,
where $\mathbf{A_i}$ refers to the pitch of the i-th note for this test case.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the minimum number of times that particular test case will require the alien to break its own rules during the conversion process.

Limits

Memory limit: 1GB.
$1 \leq \mathbf{T} \leq 100$.
$1 \leq \mathbf{A_i} \leq 10^6$.

Test set 1

Time limit: 20 seconds.
$1 \leq \mathbf{K} \leq 10$.

Test set 2

Time limit: 40 seconds.
$1 \leq \mathbf{K} \leq 10^4$.

Sample

Input	Output
2 5 1 5 100 500 1 8 2 3 4 5 6 7 8 9	Case #1: 0 Case #2: 1

We will use the notation A, B, C, D for the alien piano keys where A is the lowest note, and D is the highest. In sample case #1, the alien can simply map our song into the following sequence: `A B C D C` and this correctly reflects all the following:

- our first note with pitch 1 maps to A,
- our second note with pitch 5 maps to its key B. $5 > 1$, and B is a higher key than A,
- our third note with pitch 100 maps to its key C. $100 > 5$, and C is a higher key than B,
- our fourth note with pitch 500 maps to its key D. $500 > 100$, and D is a higher key than C,
- our fifth note with pitch 1 maps to its key C. $1 < 500$, and C is a lower key than D.

So none of the rules are broken. Note: `A B C D C` is not the only way of conversion. `A B C D A` or `A B C D B` are also eligible conversions.

In sample case #2, the only conversion sequence that provides the minimal result of 1 rule broken is: `A B C D A B C D`. Notably, the rule break comes from the fact that our 4th note with pitch 5 is lower than our 5th note with pitch 6, but A is a lower key than D.