Evaluate Escape Character

Strings represented in computers often include special characters which are not printed, instead they have some other function, such as the newline character '\n'.

For ease of manipulation, these characters are usually represented by a regular character, preceded by a so called 'escape' - in the above example, the character '\'.

Buj recently bought a string **S** of length **n** on the local market, for (according to him completely legal and within the law) home manipulation.

He forgot to read the online store ratings and customer reviews.

Upong arriving at home and taking the string out of the unbelievably eco-unfriendly packaging, he realized with horror that the characters of the string do not follow any known encoding. The only thing he managed to do was number the characters from **0** to **n-1** in perceived lexicographical order.

Oh well, thought Buj, I'll still manipulate this string to my heart's content... but what if I got scammed, and sold a low-quality string? One which, if it was printed out, would be lexicographically large?

But what would be printed out depends on which character in the string is the escape.

Buj sent us the string by post for analysis, and we uploaded it to this website hoping someone would do the work for us.

Input

The first line contains an integer **1 ≤ T ≤ 10** - the number of test cases. **T** cases follow.

For each case, the first line contains the number **n** - the length of the string - and the second line contains a space-separated permutation of the numbers **0, ... , n-1** - the characters of the string.

The sum of **n** within an input file will not exceed 10^6.

Output

In this problem, escape characters work as follows: if it is the last character in the string, it has no effect and is simply printed out as usual. Otherwise, that character and the character immediately following it is not printed (and instead together they have some other, printing-unrelated function). Take a look at the sample for clarification.

Let s_i be the string which would be printed if we choose character number **i** as the escape character.

Let p_i be the 0-based position of s_i if we sort all such strings lexicographically.

Output in a single line the numbers p_0, p_1, ..., p_{n-1}.

In other words, output in order the answer to the questions

"How many strings out of $\mathbf{s_0}$, ..., $\mathbf{s_{n-1}}$ are lexicographically smaller than $\mathbf{s_0}$?"

"How many are smaller than $\mathbf{s_1}$?"

...

"How many are smaller than $\mathbf{s_{n-1}}$?"

Example

Input:
2
3
0 1 2
3
0 2 1
Output:
2 0 1
2 1 0

In the first case:

If character 0 was the escape, the printed string would be 2.

If 1 was the escape, the printed string would be 0.

If 2 was the escape, the printed string would be 0 1 2.

Ordering these strings, we get [0, 0 1 2, 2].

So $\mathbf{s_0}$ is at index 2, $\mathbf{s_1}$ at index 0, and $\mathbf{s_2}$ at index 1, so the output is 2 0 1.

Note that the **i**-th number in the output is for the string where character **number i** is the escape, not the **i**-th character in the input.