G – Suffix reconstruction

Given a text $s[1..n]$ of length n, we create its suffix array by taking all its suffixes:

$$s[1..n], s[2..n], \ldots, s[n..n]$$

and sorting them lexicographically. As a result we get a sorted list of suffixes:

$$s[p(1)..n], s[p(2)..n], \ldots, s[p(n)..n]$$

and call the sequence $p(1), p(2), \ldots, p(n)$ the suffix array of $s[1..n]$. For example, if $s = abbaabab$, the sorted list of all suffixes becomes:

$$aabab, ab, abab, abbaabab, b, baabab, bab, bbaabab$$

and the suffix array is $4, 7, 5, 1, 8, 3, 6, 2$.

It turns out that it is possible to construct this array in a linear time. Your task will be completely different, though: given $p(1), p(2), p(3), \ldots, p(n)$ you should check if there exist at least one text consisting of lowercase letters of the English alphabet for which this sequence is the suffix array. If so, output any such text. Otherwise output -1.

Input

The input contains several descriptions of suffix arrays. The first line contains the number of descriptions t ($t \leq 100$). Each description begins with a line containing the length of both the text and the array n ($1 \leq n \leq 500000$). Next line contains integers $p(1), p(2), \ldots, p(n)$. You may assume that $1 \leq p(i) \leq n$ and no value of $p(i)$ occurs twice. Total size of the input will not exceed 50MB.

Output

For each test case output **any** text resulting in the given suffix array. In case there is no such text consisting of lowercase letters of the English alphabet, output -1.

Example

Input	Output

5
2
1 2
2
2 1
3
2 3 1
6
3 4 5 1 2 6
14
3 10 2 12 14 5 13 4 1 8 6 11 7 9
7
5 1 7 4 3 2 6
	ab
	aa
	bab
	suffix
	reconstruction
	issofun