Commuting Functions

Two functions f and g (f, g: X → X) are commuting if and only if f(g(x)) = g(f(x)) for each x in X. For example, functions f(x) = x + 1 and g(x) = x – 2 are commuting, whereas functions f(x) = x + 1 and g(x) = 2x are not commuting.

Each function h (h: N_n → N_n, where N_n = {1, 2, ... n} and n is positive integer) can be represented as a value list — a list in which the i-th element is equal to h(i). For example, a function h(x) = ceil(x/2) from N_5 to N_5 has the value list [1, 1, 2, 2, 3].

The value lists are ordered lexicographically: list [a_1 ... a_n] is smaller than list [b_1 ... b_n] if and only if there exists such an index k that $a_k < b_k$, and $a_l = b_l$ for any index l < k.

The function f (f: X → X) is bijective if for every y in X, there is exactly one x in X such that f(x) = y.

Given a bijective function f (f: N_n → N_n, n is positive integer), find the function g that is commuting with f and has the lexicographically smallest possible value list.

Input

The first line of the input file contains the number of test cases. Each test case is described by a line containing a single integer number n — the number of the elements in the value list of a bijective function f (1 ≤ n ≤ 200000), followed by another line which contains the value list of the function f.

Output

For each test case, output a single line containing n integer numbers — the value list of function g that commutes with the function f and has the lexicographically smallest value list.

Example

Input:

```
2
10
1 2 3 4 5 6 7 8 9 10
10
2 3 4 5 6 7 8 1 9 10
```

Output:

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