Problem E. 2-3-4 Tree

Time limit: 10 seconds

In computer science, a 2-3-4 tree, as a search tree, is a self-balancing data structure that is commonly used to implement dictionaries. The numbers mean a tree where every node with children (internal node) has either two, three or four child nodes.

A 2-node is the same as a binary node, which has one data element, and if internal has two child nodes. A 3-node has two data elements, and if internal has three child nodes. A 4-node has three data elements, and if internal has four child nodes. All leaves are at the same depth, and all data is kept in sorted order.

Here we introduce the procedure of inserting a value into a 2-3-4 tree. If the number of values that have been inserted in is no more than 2, the 2-3-4 tree after the insertion has only one node (the root) with several data elements.

Otherwise, we start the following procedure at the root of the 2-3-4 tree.

1. If the current node is a 4-node:
   - Remove and save the middle value to get a 3-node
   - Split the remaining 3-node up into a pair of 2-nodes (the now missing middle value is handled in the next step).
   - If this is the root node (which thus has no parent):
     - the middle value becomes the new root 2-node and the tree height increases by 1. Ascend into the root (which means that the current node becomes the new root node).
   - Otherwise, push the middle value up into the parent node. Ascend into the parent node (which means that the current node becomes its parent node).
2. If the current node is a leaf, insert the value into the node and finish.
3. Otherwise,
   - Find the child whose interval contains the value to be inserted.
   - Descend into the child and repeat from step 1.

Now you are given a 2-3-4 tree which is empty at first. Then we insert n values $a_1, a_2, \cdots, a_n$ into the 2-3-4 tree one by one. You are asked to write a program to print the 2-3-4 tree after inserting all the given values along the pre-order traversal. For a node with one or more data elements, your program will print these data elements in one line in ascending order.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 50.

For each test case, the first line contains an integer n ($1 \leq n \leq 5000$) indicating the number of values that will be inserted into the 2-3-4 tree. The second line contains n integers $a_1, a_2, \cdots, a_n$. We guarantee that $a_1, a_2, \cdots, a_n$ is a permutation of $1, 2, \cdots, n$.

Output

For each test case, output a line containing `Case #x:` at first, where `x` is the test case number starting from 1. The following several lines describe the 2-3-4 tree along the pre-order traversal, each line of which describes a node with several integers indicating the data elements of a node in the 2-3-4 tree.

Sample

standard input	standard output
3 4 1 2 3 4 4 4 3 2 1 17 6 3 5 7 1 10 2 9 4 8 11 12 13 14 15 16 17	Case #1: 2 1 3 4 Case #2: 3 1 2 4 Case #3: 5 9 2 1 3 4 7 6 8 11 13 15 10 12 14 16 17

Hint

The first figure as follow is the process of the 4-th insertion in the first sample test case.

The second one shows the final 2-3-4 tree of the third test case.