AVL Tree Heist

Even though Arup doesn't like data structures, he's decided that it's still in his students' best interest to get some practice with AVL Trees. As a compromise, he's decided that it's good enough if the students just implement the insert function instead of both the insert and delete. To make his assignment easy to grade, he'll simply give students a sequence of inserts to perform and then ask the students to output a pre-order traversal after each insertion to make sure that the structure of the tree is correct.

You have decided that while you like regular binary search trees, you feel that AVL Trees that have inserts with rebalancing are too much work. Thus, you've come up with a nefarious plan. You will infiltrate Arup's office by bribing the janitorial staff, steal his test cases and remove all of the cases where a rebalancing would have been required. This way, you can just write a regular binary search tree insert and STILL earn a 100!

For this program, read in a sequence of inserts into an AVL tree and simply determine if performing those inserts ever requires a rebalancing. If so, print out "REMOVE", otherwise print out "KEEP". (Note: In the spirit of the assignment, notice that you can solve the given problem without ever performing any rebalances!!!)

Input

The first line of input will contain a single positive integer, *T (T ≤ 100),* the number of input cases. The following *T* lines will contain each of the test cases, one test case per line. For each test case, the first integer on the line, *n (n < 1024),* will represent the total number of inserts for the test case. The following **distinct** *n* positive integers on the line represent the values to insert into the AVL tree, in the order that they are given. All of these values will be separated by a space. (There may or may not be a space at the end of each of these lines.)

Output

For each test case, start the output with a header of the following form:

```
Tree #k:
```

where k is the number of the test case, starting with 1.

Follow this with a space and either the string "REMOVE" or "KEEP", based on the cases described above.

Sample Input

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

Sample Output

```
Tree #1: REMOVE
Tree #2: KEEP
```