Tree Grafting

Trees have many applications in computer science. Perhaps the most commonly used trees are rooted binary trees, but there are other types of rooted trees that may be useful as well. One example is ordered trees, in which the subtrees for any given node are ordered. The number of children of each node is variable, and there is no limit on the number. Formally, an ordered tree consists of a finite set of nodes T such that

- there is one node designated as the root, denoted root(T);
- the remaining nodes are partitioned into subsets T1, T2, ..., Tm, each of which is also a tree (subtrees).

Also, define root(T1), ..., root(Tm) to be the children of root(T), with root(Ti) being the i-th child. The nodes root(T1), ..., root(Tm) are siblings.

It is often more convenient to represent an ordered tree as a rooted binary tree, so that each node can be stored in the same amount of memory. The conversion is performed by the following steps:

1. remove all edges from each node to its children;
2. for each node, add an edge to its first child in T (if any) as the left child;
3. for each node, add an edge to its next sibling in T (if any) as the right child.

This is illustrated by the following:

```
   0                          0
  / | \                      /
 1  2  3      ===>              1
   / \                       \
  4   5                         2
                             / \
                            4   3
                             \
                              5
```

In most cases, the height of the tree (the number of edges in the longest root-to-leaf path) increases after the conversion. This is undesirable because the complexity of many algorithms on trees depends on its height.

You are asked to write a program that computes the height of the tree before and after the conversion.

Input

The input is given by a number of lines giving the directions taken in a depth-first traversal of the trees. There is one line for each tree. For example, the tree above would give dudduduudu, meaning 0 down to 1, 1 up to 0, 0 down to 2, etc. The input is terminated by a line whose first character is #. You may assume that each tree has at least 2 and no more than 10000 nodes.

Output

For each tree, print the heights of the tree before and after the conversion specified above. Use the format:

```
Tree t: h1 => h2
```

where t is the case number (starting from 1), h1 is the height of the tree before the conversion, and h2 is the height of the tree after the conversion.

Example

Input:

```
dudduduudu
ddddduuuuu
ddddduduuuu
dddduuduuu
#
```

Output:

```
Tree 1: 2 => 4
Tree 2: 5 => 5
Tree 3: 4 => 5
Tree 4: 4 => 4
```