Binary Search Heap Construction

Read the statement of problem G for the definitions concerning trees. In the following we define the basic terminology of heaps. A **heap** is a tree whose internal nodes have each assigned a **priority** (a number) such that the priority of each internal node is less than the priority of its parent. As a consequence, the root has the greatest priority in the tree, which is one of the reasons why heaps can be used for the implementation of priority queues and for sorting.

A binary tree in which each internal node has both a label and a priority, and which is both a binary search tree with respect to the labels and a heap with respect to the priorities, is called a **treap**. Your task is, given a set of label-priority-pairs, with unique labels and unique priorities, to construct a treap containing this data.

Input Specification

The input contains several test cases. Every test case starts with an integer `n`. You may assume that `1<=n<=50000`. Then follow `n` pairs of strings and numbers $l_1/p_1,...,l_n/p_n$ denoting the label and priority of each node. The strings are non-empty and composed of lower-case letters, and the numbers are non-negative integers. The last test case is followed by a zero.

Output Specification

For each test case output on a single line a treap that contains the specified nodes. A treap is printed as `(<left sub-treap><label>/<priority><right sub-treap>)`. The sub-treaps are printed recursively, and omitted if leafs.

Sample Input

```
7 a/7 b/6 c/5 d/4 e/3 f/2 g/1
7 a/1 b/2 c/3 d/4 e/5 f/6 g/7
7 a/3 b/6 c/4 d/7 e/2 f/5 g/1
0
```

Sample Output

```
(a/7(b/6(c/5(d/4(e/3(f/2(g/1)))))))
(((((((a/1)b/2)c/3)d/4)e/5)f/6)g/7)
(((a/3)b/6(c/4))d/7((e/2)f/5(g/1)))
```