998 Dendogram

This problem is about building a tree from a simple concept.

A dendogram is a tree for visual classification of similarity, commonly used in Biology for grouping species. Let us see an example. The dendogram for the following data $(a, 1)$, $(b, 3)$ and $(c, 12)$, where the first parameter is the object name, and the second parameter is the object value, is presented in the following picture.

```
                      +----- a,1
                      |
 +------------------- 2
 |                    |
 |                    +----- b,3
-7
 |
 +-------------------------- c,12
```

The Definition and Example clarifies how to build it.

Definition and terminology. An **object** is a pair `name,value`. Elements of a dendogram are **clusters** (middle tree elements) and objects (the leaves). An association of two objects or clusters forms a new cluster. In a dendogram, **two** elements are grouped in one cluster when they have the closest values of all elements available. The arithmetic mean of the two elements is associated with the cluster.

Example. In the dendogram of the picture, objects 'a' e 'b' are close in value. Together, they make a *cluster* with mean $(1 + 3)/2 = 2$. Then object 'c' and this cluster with value 2 produce a new cluster with mean $(2 + 12)/2 = 7$.

Write a program that accept a sequence of numbers and produces a dendogram tree as a infix list of elements (clusters and objects).

The input is a sequence of positive numbers. Each number is associated with a label automatically given, starting by letter 'a', then 'b', and so on (see section **Input** below).

Solution uniqueness. Usually there are several ways to draw the same dendogram tree, for example, mirroring the dendogram tree. However, as stated in general rules, we are forced to choose a unique solution representation each time an equal input is given.

To guarantee a unique solution, the comparisons must include the names when equal values are being compared. For example, $(1, a) < (1, b)$ where 1 is the value of both 'a' and 'b' objects. Clusters also have names, and clusters with equal value must use the same lexicographical comparison. The rules are:

1. Each node represents an object or a cluster, and each node has an ASCII label with a maximum width of 30 characters.
2. Each object node has a label which is a single letter, automatically given, as described in the input section.
3. Each cluster node has a label formed by concatenation of the lexicographical 'lower order label' followed by the 'higher order label' of the two branches. The top cluster has a label which contains all used letters in the input.

4. Lower numbers must be grouped first. Example: {1, 2} must be grouped before {4, 5}.

All numeric calculations and store must use 'float' number format.

Input

The input will contain several test cases, each of them as described below. Consecutive test cases are separated by a single blank line.

A sequence of positive numbers, line by line, ended by 0.

Your program must associate each number with a letter from '`a`' to '`z`' using standard ASCII sequence (except the 0, which only ends the sequence).

Please take in consideration the following

Input Assumptions:

1. The input values are from the set {1, . . . , 99}.
2. The input has a maximum of 26 (twenty six) numbers, and more than 2 (two) numbers.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The requested output is a unique dendogram tree without shape: only the list of nodes in *infix* form. The output must obey the following rules:

1. The *infix output* of the tree is always obtained writing the nodes from the lower label to the higher label.
2. Each node represents an object or a cluster.
3. Each node in each line.
4. Each node value is outputted as a positive integer, possibly rounded to the nearest integer of the stored value.
5. Each node has the following output syntax: '*positive integer number*,*label*'. A sample is '`13,kerolinux`': this is a cluster node valuing 13 (possible 12.5 to 13.4999...) and the cluster label is '`kerolinux`'.

Sample Input

```
1
12
45
3
0
```

Sample Output

```
1,a
2,ad
3,d
7,adb
12,b
```

```
26,adbc
45,c
```