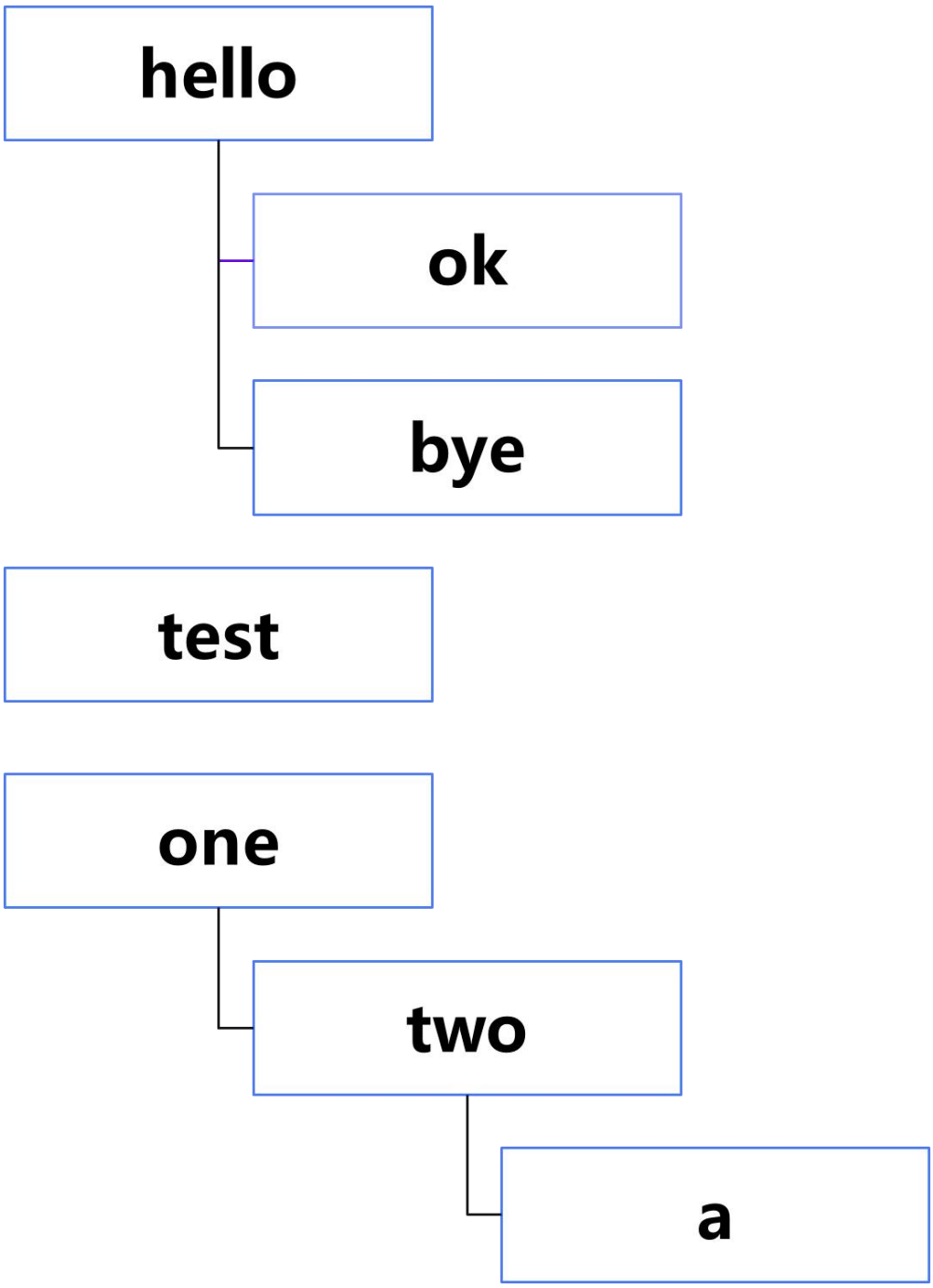
题目描述：
在一个博客网站上，每篇博客都有评论。每一条评论都是一个非空英文字母字符串。
评论具有树状结构，除了根评论外，每个评论都有一个父评论。
当评论保存时，使用以下格式：
首先是评论的内容；
然后是回复当前评论的数量。
最后是当前评论的所有子评论。（子评论使用相同的格式嵌套存储）
所有元素之间都用单个逗号分隔。
例如，如果评论如下：



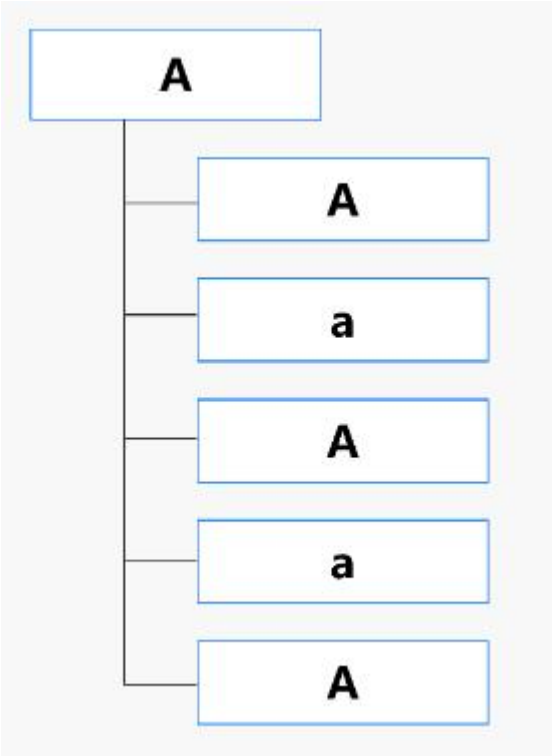
第一条评论是"hello,2,ok,0,bye,0"，第二条评论是"test,0"，第三条评论是"one,1,two,1,a,0"。
所有评论被保存成"hello,2,ok,0,bye,0,test,0,one,1,two,1,a,0"。

对于上述格式的评论，请以另外一种格式打印：
首先打印评论嵌套的最大深度。
然后是打印 n 行，第 i（1<=i<=n)行对应于嵌套级别为 i 的评论（根评论的嵌套级别为 1）。
对于第 i 行，嵌套级别为 i 的评论按照它们出现的顺序打印，用空格分隔开。

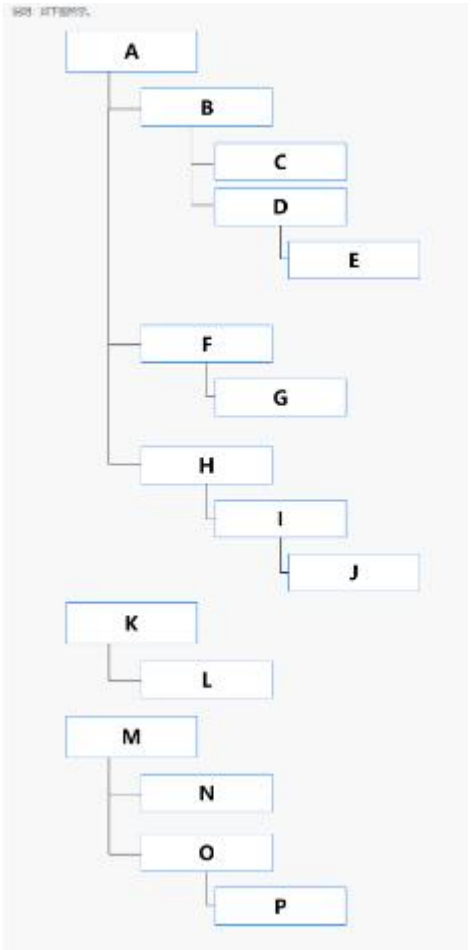
输入描述：
一行评论。由英文字母、数字和英文逗号组成。
保证每个评论都是由英文字符组成的非空字符串。
每个评论的数量都是整数（至少由一个数字组成）。
整个字符串的长度不超过 106。
给定的评论结构保证是合法的。
输出描述：
按照给定的格式打印评论。对于每一级嵌套，评论应该按照输入中的顺序打印。
补充说明：

示例1
输入：hello,2,ok,0,bye,0,test,0,one,1,two,1,a,0
输出：3
hello test one
ok bye two
a
说明：如题目描述中图所示，最大嵌套级别为3。嵌套级别为1的评论是"hello test one"，嵌套级别为2的评论是"ok bye two"，嵌套级别为3的评论为"a"。

示例2
输入：A,5,A,0,a,0,A,0,a,0,A,0
输出：2
A
A a A a A
说明：如下图所示，最大嵌套级别为2，嵌套级别为1的评论是"A"，嵌套级别为2的评论是"A a A a A"



示例3
输入：A,3,B,2,C,0,D,1,E,0,F,1,G,0,H,1,I,1,J,0,K,1,L,0,M,2,N,0,O,1,P,0
输出：4
A K M
B F H L N O
C D G I P
E J
说明：如下图所示。


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String s = in.nextLine();
        String[] arr = s.split(",");
        int len = arr.length;
        int index = 0;
        int level = 0;
        int n_level = 0;
        ArrayList<ArrayList<String>> list = new ArrayList<>();
        for (int i = 0; i < 100000; i++) {
            list.add(new ArrayList<>());
        }
        Deque<Node> stack = new LinkedList<>();
        while (index < len) {
            String ss = arr[index++];
            int count = Integer.valueOf(arr[index++]);
            if (count == 0) {
                if (n_level + 1 > level) level = n_level + 1;
                if (stack.isEmpty()) {
                    list.get(0).add(ss);
                } else {
                    list.get(n_level).add(ss);
                    while (true) {
                        Node h = stack.peek();
                        if(h == null) break;
                        h.count--;
                        if (h.count == 0) {
                            n_level--;
                            list.get(n_level).add(h.val);
                            stack.pop();
                        } else break;
                    }
                }
            } else {
                stack.push(new Node(count, ss));
                n_level++;
                if (n_level > level) level = n_level;
            }

        }
        System.out.println(level);
        for (int i = 0; i < level; i++) {
            ArrayList<String> l = list.get(i);
            for (int j = 0; j < l.size(); j++) {
                System.out.print(l.get(j) + " ");
            }
            System.out.println();
        }
    }
    static class Node {
        int count;
        String val;
        public Node(int count, String val) {
            this.count = count;
            this.val = val;
        }

    }
}
```