Drawing Rooted Binary Trees

Time limit: *1000 ms*
Memory limit: *256 MB*

Binary trees are a common data structure and understanding some basic operations is important. For this problem, you will play with traversals and tree drawing strategies.

An ASCII Representation of a Tree

When you are debugging something that uses a **binary tree** is often valuable to be able to get a quick description of the tree to the console (in characters) so you can check its structure. The basic strategy is to use spaces to position the nodes so that you can infer the tree's connectivity. For example, a tree that is just a root with two children can be "drawn" as:

```
1    a
2  b c
3
```

These types of drawings follow these rules:

- all nodes in a left subtree must be to the left of the root
- all nodes in a right subtree much be to the right of the root
- the root must be between the two subtrees
- no other node will be in the same column as the root
- every node must be as far left as possible without breaking the other rules

Standard input

For this problem, all node labels are single characters and every tree will contain at least one node. Each instance of the problem requires two lines of input:

- the first line will be the infix traversal of the tree
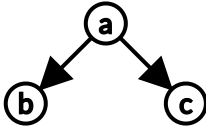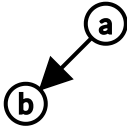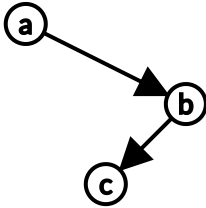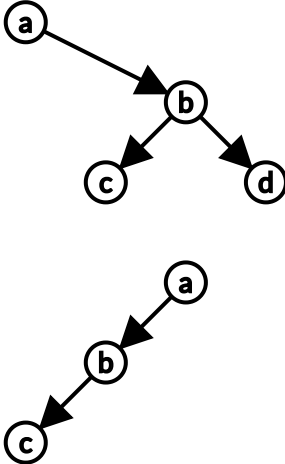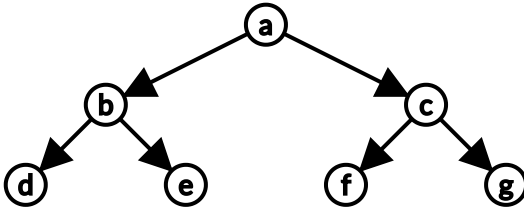- the second line will be the prefix traversal of the tree

One run of the problem may contain many instances of the problem.

Standard output

For each instance of the problem, you must output the ASCII representation of the tree (trailing characters in a line will be ignored). Multiple instance of the problem should follow immediately (no blank lines between them).

Constraints and notes

- All nodes are represented by lowercase English letters

Input	Output	Explanation
bac abc	a b c	
ba ab	a b	
acb abc	a b c	

acbd
abcd
cba
abc
	a
	  b
	  c d
	   a
	  b
	c	

dbeafcg abdecfg	a b c d e f g	