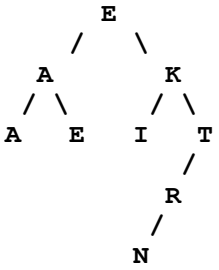
6. Felipe

Program Name: Felipe.java Input File: felipe.dat

Felipe is fascinated with binary search trees, especially with the four different traversals he just learned in class: inorder, preorder, postorder, and reverse order.

According to the rules he learned in class about building a character tree using a word, he practices doing this with his friend's names, and especially enjoys using his beautiful girlfriend's equally beautiful name, EKATERINA, with whom he is head-over-heels in love!

He starts with the first letter, E, as the root, and then sees that the next letter, K, is alphabetically after E, so he inserts it as a right node to the E, and then the A as a left node to the E. When he gets to the T, he goes right at the E, and then right again at the K, and proceeds on through the rest of the letters of her name, resulting in the binary search tree shown below. He decides to allow duplicate letters and slides those to the left as they reach a matching node.



He then proceeds to traverse in order, which means start at the top of the tree, going as deep to the left along each branch, and only outputting a node when reaching the end of a branch (a leaf node) or outputting a node after its left branch has been completely processed.

The inorder traversal (alpha order) of this tree would be: **AAEEIKNRT**
The preorder traversal (output a node BEFORE traversing either the left or right subtrees) would be: **EAAEKITRN**
Postorder traversal (output after both branches are completely traversed): **AEAINRTKE**
The reverse order traversal is just the inorder traversal backwards, which Felipe thinks about for a bit, and then realizes what to do, resulting in: **TRNKIEEAA**

Input: Several uppercased names containing no other symbols , each on one line, and each followed by a single character indicating the traversal to be performed: **E**(preorder), **O** (postorder), **I** (inorder), **R** (reverse order).

Output: The indicated traversal for each name, as shown in the examples.

Sample Input:
```
EKATERINA I
EKATERINA E
EKATERINA O
EKATERINA R
```

Sample Output:
```
AAEEIKNRT
EAAEKITRN
AEAINRTKE
TRNKIEEAA
```