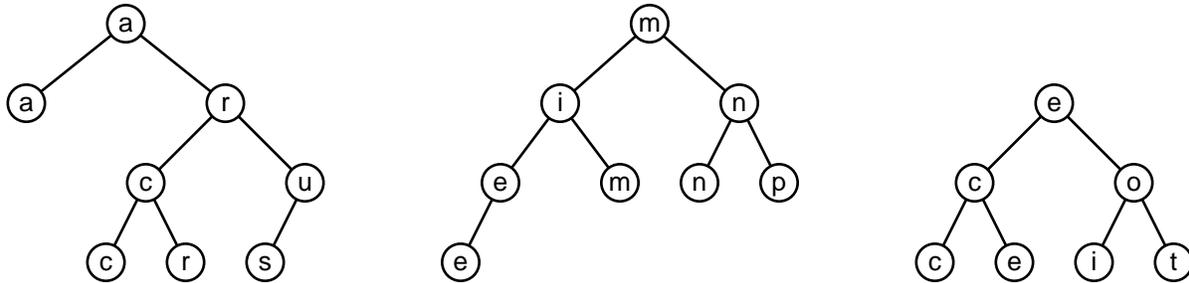


**“A Balancing Act”**

**Problem:** A tempting approach to maintaining a balanced binary search tree is to maintain multiple binary search trees and to insert each new key into the tree that will be more balanced.

More specifically, do the following: The first key is the root of the first tree. The second key is the root of the 2nd tree. And so on for  $k$  trees. To add a new key, insert it into the tree where it would have the smallest depth. If the depths of all trees are the same, then add it to the first tree.

For example, considering the following three trees, built from the string `american computer science`:



The input will be 5 sets of data. Each set will consist of a number  $k$  and a string  $s$ . In the string  $s$ , ignore everything but the letters of the alphabet; uppercase and lowercase are the same. For each input pair, build the  $k$  trees with the letters in  $s$  as described above. Print contents of the first tree in preorder (root, then the left child, then the right child).

**Sample Input:**

Line #1: 3, AMERICAN COMPUTER SCIENCE  
 Line #2: 4, AMERICAN COMPUTER SCIENCE  
 Line #3: 3, SENIOR DIVISION  
 Line #4: 4, SENIOR DIVISION  
 Line #5: 2, EASY PROBLEM

**Sample Output:**

Output #1: A A R C C R U S  
 Output #2: A A I E P  
 Output #3: S I I S V  
 Output #4: S O I V  
 Output #5: E B E S P O M

“A Balancing Act”

Test Input:

Line #1: 2, Time flies like an arrow. Fruit flies like a banana.  
Line #2: 3, I must say that I find television very educational.  
Line #3: 4, The minute somebody turns it on, I go to the library and read a book.  
Line #4: 5, One morning I shot an elephant in my pajamas.  
Line #5: 1, How he got into my pajamas I'll never know!

Test Output:

Output #1: T M F E A A A E F I I I L S R O S T W U  
Output #2: I A A A A I E E I S N L N S R T T Y  
Output #3: T I B B A I D O O N S R U U Y  
Output #4: O N H A M R P T  
Output #5: H H E A A A E E G O O I I N M J M L L K N N O O W T T P S R V W Y