

**Due:** Friday, October 14 2016

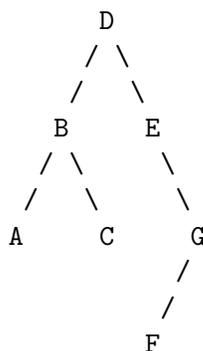
Rather than have a lab devoted to current topics in the course, we're going to have a problem-solving lab instead. You won't be graded on how much you complete successfully (as usual), but we do suggest giving it a try. The problems here come from various programming contests. The idea in these contests is to work for speed of programming. Some teams of three at the ACM International Programming Contest, for example, will solve 12 (or even 13) problems like these in five hours.

In all of these problems, the input will come from the standard input and output will go to the standard output.

You'll find (trivial) skeletons and some test data for your answers in the `lab8` staff directory.



2. [Adapted from the Valladolid archives] Consider binary trees whose nodes are labeled with single digits and letters (upper- and lowercase, case-sensitive). Given the preorder (root, left subtree, right subtree) node order and the inorder (left subtree, root, right subtree) node order for the same tree, it is possible to reconstruct the tree, assuming no two nodes in the tree have the same label. For example, given the preorder traversal order “DBACEGF” and the inorder traversal order “ABCDEFGG,” you can compute that the tree these come from is



You are to write a program to do this reconstruction on any such tree.

The input consists of one or more cases in free format. Each case consists of two non-empty strings *PRE* and *IN*, representing the preorder traversal and inorder traversal of a non-empty binary tree, and consisting of letters and digits. Case is significant.

For each test case, print the reconstructed tree’s postorder traversal (left subtree, right subtree, root), using the format shown in the example.

**Example:**

Input	Output
DBACEGF ABCDEF BCAD CBAD	Case 1: ACBFGED  Case 2: CDAB

**3.** [Due to E. W. Dijkstra] Consider decimal numerals containing only the digits 1–3. A numeral is considered “good” if no two adjacent non-empty substrings of it are equal; otherwise it is “bad.” Hence, the numerals ‘1’, ‘12’, and ‘1213’ are good, while ‘11’, ‘32121’, and ‘121321312’ are bad.

You are to write a program that, given  $n > 0$ , finds the smallest good  $n$ -digit numeral. The input consists of a sequence of positive integers. For each of these integers,  $n$ , the output is to contain a line of the form

The smallest good numeral of length  $n$  is  $s$ .

where  $s$  is the answer. For example,

Input	Output
1 4	The smallest good numeral of length 1 is 1.
7	The smallest good numeral of length 4 is 1213.
9	The smallest good numeral of length 7 is 1213121.
	The smallest good numeral of length 9 is 121312313.