Practical Test (19. 6. 2016)

Problem 1: for a positive integer n, let f(n) denote the sum of the digits of n when represented in base 10. It is easy to see that the sequence of numbers n, f(n), f(f(n)), f(f(f(n))), … eventually becomes a single digit number that repeats forever. Let this single digit be denoted g(n). For example, consider n=1234567892, then:

f(n) = 1+2+3+4+5+6+7+8+9+2 = 47
f(f(n)) = 4+7 = 11
f(f(f(n))) = 1+1 = 2, therefore g(1234567892) = 2

Input	Output
2	2
11	2
47	2
1234567892	2
0	

Input: Each line of input contains a single positive integer n at most 2,000,000. Input is terminated by n=0 which should not be processed.

Output: For each such integer, you are to output a single line containing g(n).

==========

Problem 2: A palindrome is a word, number or phase that reads the same forwards as backwards. For example, the name 'anna' is a palindrome. Numbers can also be palindromes (eg. 151 or 753357). Additionally, numbers can of course be ordered in size. The first few palindrome numbers are: 1,2,3,4,5,6,7,8,9,11,22,33, …. The number 10 is not a palindrome (even though you could write it as 010) but a zero as leading digit is not allowed.

Input: The input consists of a series of lines with each line containing one integer value i. The i indicates the index of the palindrome number that is to be written to the output, where index 1 stands for the first palindrome number (1), index 2 is stands for the second palindrome number (2) and so on. The input is terminated by a line containing '0;.

Output: For each line of input (except the last one) exactly one line of output containing a single (decimal) integer value is to be produced. For each input value i the i-th palindrome number is to be written to the output.

Input	Output
1	1
12	33
24	151
0	

==========

Problem 3: Whenever we think of sorting integers, we tend to think of sorting them in ascending or descending order. However, we can play around a bit and define new sorting criteria. One criterion could be sorting numbers in terms of their summation of digits. Therefore in this sorting criterion, 13 would come before 9 as sum of the digits of 13 is 4 and that of 9 is 9.

In this problem, we are concerned with sorting numbers in the range 1 to 2000000 with the following sorting criteria. Numbers in this range must be sorted in terms of the number of factors in their prime factorization. Incase of a tie, the smaller number will come first. For example, 20=2*2*5, so it has 3 numbers in its prime factorization. Similarly 35=5*7 has 2 numbers in its prime factorization. Therefore 35 will come before 20 according to this criterion. If two numbers have the same prime factorization, the smaller of the sum of these prime numbers is the first. For example, 15=3*5 has also two prime factorization, but 3+5 is smaller than 5+7, so 35 followed by 15. (15, 35, 20)

Input	Output
20	15
15	35
35	20
0	

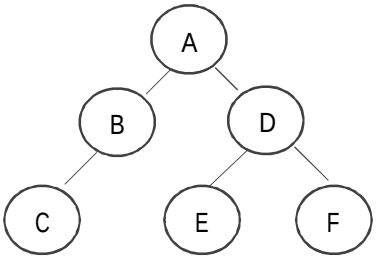
Input: Each case of input will consist of a positive integer n<200000. The last case is followed by a '0'.

Output: The output is the sorted sequence after the rule is applied.

==========

Problem 4: A common problem in data structures is to determine the traversal of a binary tree. There are three classic ways to do it:

- **Pre-order:** You must visit in sequence the root, the left subtree and the right subtree.
- **In-order:** You must visit in sequence the left subtree, the root and the right subtree.
- **Post-order:** You must visit in sequence the left subtree, the right subtree and the root.



In the following picture, the pre, in and post-order traversal are ABCDEF, CBAEDF, and CBEFDA. In this problem, you must compute the post-order traversal of a binary tree given its in-order and pre-order traversals.

***Input*:** The input set consists of a positive number c<=2000, that gives the number of test cases and C lines, one for each test case. Each test case starts with a number, ie, the number of nodes in this binary tree. After there will be two strings S1 and S2 that describe the pre-order and in-order traversal of the tree. The nodes of the tree are labeled with different characters in the range 'a'…'z' and 'A'…'Z'. The values of N, S1 and S2 are separated by a blank space.

***Output*:** For each input set, you should output a line containing the post-order traversal for the current tree.

Input	Output
3 3 xYz Yxz 3 abc cba 6 ABCDEF CBAEDF	Yzx cba CBEFDA

Problem 5: A bit is a binary digit, taking a logical value of either 0 or 1. And every decimal number has a binary representation which is actually a series of bits. If a bit of a number is 1 and its next bit is also 1 then we can say that the number has a 1 adjacent bit. And you have to find out how many times this scenario occurs for all numbers up to N.

***Input*:** For each test case, you are given an integer number. The last test case is followed by a negative integer in a line by itself.

Output: For each test case, print a line of the form 'Case <1 space> X: <1 space> Y' where X is the serial of output starting from 1 and Y is the summation of all adjacent bits from 0 to N.

Input	Output
0 6 22 -1	Case 1: 0 Case 2: 2 Case 3: 14

==========

Problem 6: The students of University of Computer Studies, Yangon like to play the game "HardNumbers". Sure? The rules are the following: you are given a sequence of digits. Your goal is to obtain a given number from them by using some operations. The operations you may perform are given in the input.

Input: The first line a test case description contains the original sequence of at most five digits without spaces. The second line contains the integer number you have to obtain. The third line contains a sequence of characters without spaces describing the rules. Each character allows one to use some operation. The operations and their characters are listed below.

Character	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Real division
x	Throwing away some (not all) of the digits
j	Concatenation of some subsequent digits in a single integer
p	Permutation of the digits

The original sequence of digits and rule description string are nonempty. The test cases are separated by blank lines.

Output: for each test cases in the input, output "Impossible" (without quotes) if it is impossible to obtain the given number using the given set of operations, or any expression yielding the number otherwise. The expression syntax is given below:

expression ::= number | '(' expression '+' expression ')' |
'(' expression '-' expression ')' | '(' expression '' expression ')' |*
'(' expression '/' expression ')' | '(' expression '+' expression ')' ;
number ::= digit | digit number;
digit ::= 0|1|2|3|4|5|6|7|8|9;

Input	Output
1 1346 24 +-*/p	(6/(1-(3/4)))

The expression must contain no spaces. It may contain an operation sign (+-/*) only if this operation is permitted by the rules. If throwing away digits is permitted, each digit must enter in the expression no more times than it entered in the original sequence of digits, otherwise each digit must enter in the expression as many times as it entered in the original sequence of digits. If concatenation of subsequent digits is not permitted, each number in the expression must contain exactly one digit. If permutation of the digits is not permitted, the sequence of digits in the expression must be a subsequence of the original sequence of digits. The value of the expression must be equal to the given number.

==========