UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS

I MARATON INTERNA DE PROGRAMACION

UNIVERSIDAD DISTRITAL

17 de Mayo de 2007

PROBLEMAS

Elaborado por: Hector Florez
Basado de www.acis.org.co

Problem 1 Continuous Fractions

Source file name: cfrac.c, cfrac.cpp or cfrac.java
Input: cfrac.in
Output: standar output

A simple continuous fraction has the form: where the a_i are integer numbers.

$$a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}}$$

The previous continuous fraction could be noted as [a1, a2, . . . , an]. It is not difficult to show that any rational number p/q , with integers p>q>0, can be represented in a unique way by a simple continuous fraction with n terms, such that p q = [a1, a2, . . . , an−1, 1], where n and the ai are positive natural numbers. Your task is to find and print the simple continuous fraction that corresponds to a given rational number.

Input

Input will consist of a series of cases, each one in a line. A line describing a case contains p and q, two integer numbers separated by a space, with $10^{20} > p > q > 0$. The end of the input is indicated by a line containing 0 0.

Output

Cases must be analyzed in the order that are read from the input. Output for each case will consist of several lines. The first line indicates the case number, starting at 1, using the format:
Case i:
replacing i by the corresponding case number. The second line displays the input data in the form p/q.
The remaining lines must contain the continuous fraction corresponding to the rational number, p/q, specified in the given input line. The continuous fraction must be printed accordingly to the following rules:
• Horizontal bars are formed by sequences of dashes '-'.
• The width of each horizontal bar is exactly equal to the width of the denominator under it.
• Blank characters should be printed using periods '.'
• The number on a fraction numerator must be printed center justified. That is, the number of spaces at either side must be same, if possible; in other case, one more space must be added at the right side.

Sample input	Sample Output
75 34 65 60 0 0	Case 1: 75 / 34 .....1..... 2.+.---------- .....1.... ....4.+.--------- .....1..

	.....1.+.----- .....1 .....5.+.- .....1 Case 2: 65 / 60 .....1... 1.+.------ .....1 ....11.+.- 1

Problem 2 Factorials !!!

Source file name: factorials.c, factorials.cpp or factorials.java
Input: factorials.in
Output: standar output

Definition

n!!..! = n(n-k)(n-2k)..(n mod k), if k doesn't divide n
n!!..! = n(n-k)(n-2k)..k, if k divides n (There are k marks ! in the both cases).

For example, 10 mod 3 = 1; 3! = 3 x 2 x 1; 10!!! = 10 x 7 x 4 x 1;

Given numbers n and k we have calculated a value of the expression in the first definition. Can you do it as well?

Input

There will be multiple cases. Each test case will be contained on one line. Each line will start with the followed by exactly one space, then k exclamation marks.

Output

For each test case you should print one line of output which contains one number – n !!..! (there k marks ! here)

Sample input	Sample Output
3 ! 10 !!! 9 !!	6 280 945

Problem 3 Making Book

Source file name: book.c, book.cpp or book.java
Input: book.in
Output: standar output

A printer - who still uses moveable type - is preparing to print a set of pages for a book. These pages are to be numbered, as usual. The printer needs to know how many instances of each decimal digit will be required to set up the page numbers in the section of the book to be printed.

For example, if pages 10, 11, 12, 13, 14and15 are to be printed, computing the number of digits is relatively simple: just look at the page numbers that will appear, and count the number of times each digit appears. The digit 0 appears only once, the digit 1 appears 7 times, the digits 2, 3, 4 and 5 each appear once, and 6, 7, 8 and 9 don't appear at all.

Your task in this problem is to provide the printer with the appropriate counts of the digits. You will be given the numbers of the two pages that identify the section of the book to be printed. You may safely assume that all pages in that section are to be numbered, that no leading zeroes will be printed, that page numbers are positive, and that no page will have more than three digits in its page number.

Input

There will be multiple cases to consider. The input for each case has two integers, A and B, each of which is guaranteed to be positive. These identify the pages to be printed. That is, each integer P between A and B, including A and B, is to be printed. A single zero will follow the input for the last case.

Output

For each input case, display the case number (1, 2, . . .) and the number of occurrences of each decimal digit 0 through 9 in the specified range of page numbers. Display your results in the format shown in the examples below.

Sample input	Sample Output
10 15 912 912 900 999 0	Case 1: 0:1 1:7 2:1 3:1 4:1 5:1 6:0 7:0 8:0 9:0 Case 2: 0:0 1:1 2:1 3:0 4:0 5:0 6:0 7:0 8:0 9:1 Case 3: 0:20 1:20 2:20 3:20 4:20 5:20 6:20 7:20 8:20 9:120

Problem 4 Quicksum

Source file name: quicksum.c, quicksum.cpp or quicksum.java
Input: quicksum.in
Output: standar output

A checksum is an algorithm that scans a packet of data and returns a single number. The idea is that if the packet is changed, the checksum will also change, so checksums are often used for detecting transmission errors, validating document contents, and in many other situations where it is necessary to detect undesirable changes in data.

For this problem, you will implement a checksum algorithm called Quicksum. A Quicksum packet allows only uppercase letters and spaces. It always begins and ends with an uppercase letter. Otherwise, spaces and letters can occur in any combination, including consecutive spaces.

A Quicksum is the sum of the products of each character's position in the packet times the character's value. A space has a value of zero, while letters have a value equal to their position in the alphabet. So, A = 1, B = 2, etc., through Z = 26. Here are example Quicksum calculations for the packets "ACM" and "MID CENTRAL":

ACM: $1 \times 1 + 2 \times 3 + 3 \times 13 = 46$
MID CENTRAL: $1 \times 13 + 2 \times 9 + 3 \times 4 + 4 \times 0 + 5 \times 3 + 6 \times 5 + 7 \times 14 + 8 \times 20 + 9 \times 18 + 10 \times 1 + 11 \times 12 = 650$

Input

The input consists of one or more packets followed by a line containing only # that signals the end of the input. Each packet is on a line by itself, does not begin or end with a space, and contains from 1 to 255 characters.

Output

For each packet, output its Quicksum on a separate line in the output.

Sample input	Sample Output
ACM MID CENTRAL REGIONAL PROGRAMMING CONTEST ACN A C M ABC BBC #	46 650 4690 49 75 14 15

Problem 5 Base Comparator

Source file name: bcomp.c, bcomp.cpp or bcomp.java
Input: bcomp.in
Output: standar output

DigiCircuits Inc. is a software company that develops software simulators for digital circuits. A very frequently used component of its software, named the comparator, is a simulated circuit that compares numbers expressed in different numerical bases. More exactly, this component receives two numbers, each one in a possible different base, and decides if the first number is less than, equal to or greater than the second number. The numerical bases that may appear vary from 1 to 9. Remember that a number expressed in base b uses only digits less than b. Your task is to develop a program that simulates the function of the comparator component.

Input

The input file contains several test cases, each one of them in a separate line. Each test case has four numerical strings, each two of them separated by a blank character, say `s b t c`. Strings `b` and `c` are one-character strings. They represent the bases for the first and third strings `s` and `t`, respectively.
The end of the input is denoted by the end of the input file.

Output

Output text for each input case is presented in the same order that input is read. For each test case the answer must be a left aligned answer-character corresponding to the input. This character must be `<` , `=` or `>` , accordingly to the fact that the first string represents a numerical value less than, equal to or greater that the represented by the third string.

Sample input	Sample Output
54 6 71 8 110 2 6 7 3 4 3 9 14 7 1000 2	< = = >

Problem 6 Making Money

Source file name: money.c, money.cpp or money.java
Input: money.in
Output: standar output

A trick sometimes used by parents to teach their children the value of money is to give then a penny - just a penny! - and the promise that for each day they don't spend it, the parent will double it. All students of computing know that long before a month has elapsed without spending a cent, the parents will not likely be able to make good on their promise. 100-percent compound daily interest on an investment is, of course, unattainable in normal financial dealings, but we are all continually reminded of the power of compound interest, even with the relatively low interest rates available today.

But exactly how much money can be made with compound interest? Assume, for example, an initial investment of $100.00 (US or Canadian), an annual interest rate of 6.00 percent, and that interest is compounded monthly. That is, the interest earned during the preceding month is added to the principal at the end of the month. (For our purposes, we'll assume a month is exactly 1/12th of a year.)

At the end of the first month, the money will have earned 0.5 percent interest (1/12th of 6.00 percent), or $0.50. This is added to the $100.00 invested, so that during the next month, interest is paid on $100.50. During the next month another 0.5 percent interest is earned, which is exactly $0.5025. We will assume that the bank, being conservative, will not pay any interest less than $0.01, so our investment is credited with an additional $0.50 at the end of the second month, for a whopping total of $101.00. Continuing in the same manner, at the end of 12 months our investment will total $106.12, $0.12 more than simple 6.00 percent interest for a year with no compounding. Given an amount P to be invested for a year with I percent interest, compounded C times during the year at equal intervals, what is total return on the investment?

Input

There will be multiple cases to consider. The input for each case is a single line containing the initial investment amount, P, given in dollars and cents (but no fractional cents, and no larger than $100,000.00), the annual interest rate (I) given as a real number with two fractional digits representing a percentage, greater than zero but less than 100, and the number of compounding intervals per year (C), an integer between 1 and 365. The last case will be followed by a line containing "0.00 0.00 0".

Output

For each input case, display the case number (1, 2, . . .), the initial investment (P), the annual interest rate (I), the number of compounding intervals per year(C), and the value of the investment at the end of a year. Your output should follow the format shown in the examples below.

Sample input	Sample Output
100.00 6.00 1 100.00 6.00 12 1000.00 6.00 12 0.00 0.00 0	Case 1. $100.00 at 6.00% APR compounded 1 times yields $106.00 Case 2. $100.00 at 6.00% APR compounded 12 times yields $106.12 Case 3. $1000.00 at 6.00% APR compounded 12 times yields $1061.63