I MARATON INTERNA DE PROGRAMACION

UNIVERSIDAD KONRAD LORENZ

14 de Abril 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)

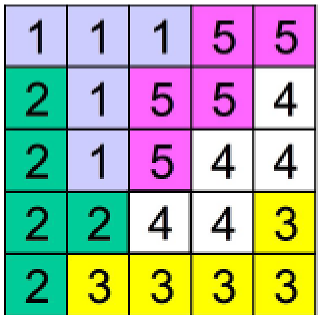
Sample input	**Sample Output**
3 ! 10 !!! 9 !!	6 280 945

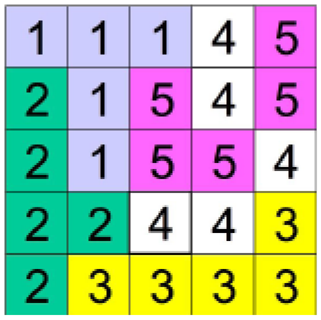
Problem 3

Equidivisions

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

An equidivision of an n × n square array of cells is a partition of the n2 cells in the array in exactly n sets, each one with n contiguous cells. Two cells are contiguous when they have a common side.
A good equidivision is composed of contiguous regions. The figures show a good and a wrong equidivision for a 5×5 square:





Note that in the second example the cells labeled with 4 describe three non-contiguous regions and cells labeled with 5 describe two non-contiguous regions. You must write a program that evaluates if an equidivision of the cells in a square array is good or not.

Input
It is understood that a cell in an n x n square array is denoted by a pair (i, j), with 1<= i, j <= n. The input file contains several test cases. Each test case begins with a line indicating n, 0 < n < 100, the side of the square array to be partitioned. Next, there are n − 1 lines, each one corresponding to one partition of the cells of the square, with some non-negative integer numbers. Consecutive integers in a line are separated with a single blank character. A line of the form: a1 a2 a3 a4 ... means that cells denoted with the pairs (a1, a2), (a3, a4), ... belong to one of the areas in the partition. The last area in the partition is defined by those cells not mentioned in the n − 1 given lines. If a case begins with n = 0 it means that there are no more cases to analyze.

Output

For each test case good must be printed if the equidivision is good, in other case, wrong must be printed. The answers for the different cases must preserve the order of the input.

Sample input	**Sample Output**

2
1 2 2 1
5
1 1 1 2 1 3 3 2 2 2
2 1 4 2 4 1 5 1 3 1
	wrong
	good
	wrong

4 5 5 2 5 3 5 5 5 4
2 5 3 4 3 5 4 3 4 4
5
1 1 1 2 1 3 3 2 2 2
2 1 3 1 4 1 5 1 4 2
4 5 5 2 5 3 5 5 5 4
2 4 1 4 3 5 4 3 4 4
0	

Problem 4

Odd or Even

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

There are several versions of Odd or Even, a game played by competitors to decide random issues. In one of the versions, for two players, the game starts with each player calling either odds or evens. Then they count to three (some people chant “Once, twice, three, SHOOT!”). On three, both players hold out one of their hands, showing a number of fingers (from zero to five). If the fingers add to an even number, then the person who called evens wins. If the fingers add to an odd number, then the person who called odds wins.
John and Mary played several games of Odd or Even. In every game John chose odds (and, Mary chose evens). During the games each player wrote down, in small cards, how many fingers he/she showed, using one card for each game Mary used blue cards, John used red cards. Their objective was to be able to re-check the results later, looking at the cards for each game. However, at the end of the day John dropped the deck of cards, and although they could separate the cards by color, they are now out of order. Given the set of numbers written on red cards and on blue cards, you must write a program to determine the minimum number of games that Mary certainly won.

Input

The input contains several test cases. The first line of a test case contains an integer N representing the number of games played ($1<N<100$). The second line of a test case contains N integers X_i, indicating the number of fingers shown by Mary in each of the games ($0<=X_i<=5$, for $1<=i<=N$). The third line of a test case contains N integers Y_i, indicating the number of fingers shown by John in each of the games ($0<=Y_i<=5$, for $1<=i<=N$). The end of input is indicated by N = 0.

Output

For each test case your program must write one line, containing one integer, indicating the minimum number of games that Mary certainly won.

Sample input	Sample Output

3
1 0 4
3 1 2
9
0 2 2 4 2 1 2 0 4
1 2 3 4 5 0 1 2 3
0
	0
	3

Problem 5

Uncle Jack

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

Dear Uncle Jack is willing to give away some of his collectable CDs to his nephews. Among the titles you can find very rare albums of Hard Rock, Classical Music, Reggae and much more; each title is considered to be unique. Last week he was listening to one of his favorite songs, Nobody's fool, and realized that it would be prudent to be aware of the many ways he can give away the CDs among some of his nephews.
So far he has not made up his mind about the total amount of CDs and the number of nephews. Indeed, a given nephew may receive no CDs at all.
Please help dear Uncle Jack, given the total number of CDs and the number of nephews, to calculate the number of different ways to distribute the CDs among the nephews.

Input

The input consists of several test cases. Each test case is given in a single line of the input by, space separated, integers N (1 <= N <= 10) and D (0 <= D <= 25), corresponding to the number of nephews and the number of CDs respectively. The end of the test cases is indicated with N = D = 0.

Output

The output consists of several lines, one per test case, following the order given by the input. Each line has the number of all possible ways to distribute D CDs among N nephews. The output must be written to standard output.

Sample input	**Sample Output**

1 20
3 10
0 0
	1
	59049

Problem 6

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
	<
	=
	=
	>