IV MARATON INTERNA DE PROGRAMACION

UNIVERSIDAD KONRAD LORENZ

6 de Septiembre de 2008

PROBLEMAS

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

Problem 1

Ink Blots

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

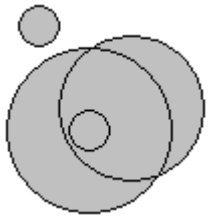


Figure 1:
1 White Region

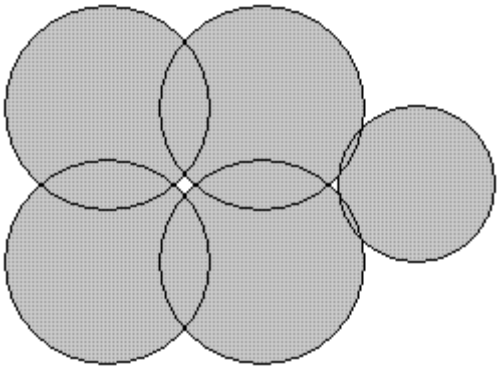


Figure 2:
3 White Regions

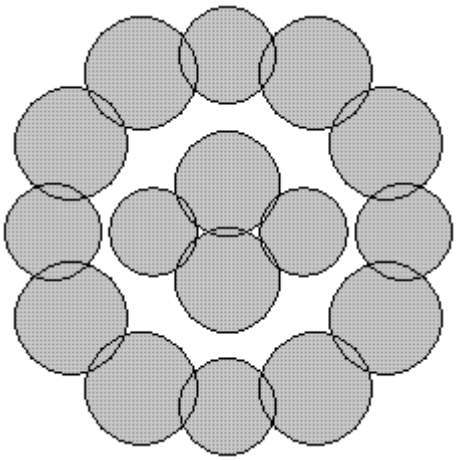


Figure 3:
4 White Regions

Drops of dark ink can fall on a white piece of paper creating a number of round ink blots. Three examples are shown above. The blots can create multiple distinct white regions. In the first figure, there is just one white region. In the second figure there is the outer white region plus a small white region bounded by the left four blots and an even smaller white region bounded by the right three blots. In the third figure, there are four white regions, one on the very outside, one inside the outer ring of blots and outside the four blots in the middle, and two tiny ones each formed between three of the four inner blots.

Two points are in the same white region if a path can be drawn between them that only passes through white points. Your problem is to count the number of white regions given the centers and radii of the blots.

Math Formulas: If circles
C1 with center (x1, y1) and radius r1, and
C2 with center (x2, y2) and radius r2

intersect in exactly two distinct points, let

d equal the distance between the centers of C1 and C2,
A = atan2(y2-y1, x2-x1), and
B = acos((r12 + d2 - r22)/(2*r1*d));

then the intersection points on C1 are at angles A+B and A-B radians counterclockwise from the ray extending to the right from the center of C1. The function atan2 is the inverse tangent function with two arguments, and acos is the inverse cosine function, both available in the math libraries of C, C++, and Java.

Input:

There are from one to 15 data sets, followed by a final line containing only 0. A data set starts with a line containing a single positive integer n, which is no more than 100. It is the number of blots in the dataset. Then 3n positive integers follow, with a single blank or a newline separating them. Each group of three give the data for the circular boundary of one blot: x and y coordinate of the center of the blot and its radius, in that order. Each of these numbers will be no larger than 1,000,000. All blots lie entirely on a piece of paper, and no blot touches any edge of the paper. No two circles in a dataset will be identical. Given any two distinct circles, they will either intersect at exactly two distinct points or not intersect at all. If two circles in the input intersect, then they overlap by at least one unit. More precisely, if they have radii r1 and r2, where $r1 \leq r2$, and if d is the distance between their centers, then $r2 - r1 + 1 \leq d \leq r1 + r2 - 1$.

Three or more circles will never intersect at the same point. If C is a circle in the input that intersects at least one other input circle, and p and q are any of the intersection points of C with any of the other input circles, with p distinct from q, then p and q will be separated on C by at least 0.001 radians of arc. The restrictions on radii and angles ensure that standard double-precision arithmetic is sufficient for the calculations suggested above.

The sample input below corresponds to the figures above, though the scale is different in each figure.

Output:

The output contains one line for each data set. The line contains only the number of white regions for the dataset, which is never more than 200.

Warning: Brute force raster methods of solving this problem will take up too much memory and be too slow.

Sample input	Sample Output
4 45 45 40 65 55 35 45 45 10 20 95 10 5 30 30 20 30 60 20 60 30 20 60 60 20 90 45 15 16 200 120 65 300 100 55 400 120 65 480 200 65 500 300 55 480 400 65 400 480 65 300 500 55 200 480 65 120 400 65 100 300 55 120 200 65 300 245 60 300 355 60 385 300 51 215 300 51 0	1 3 4

Problem 2

Decomposition

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

Definition

70	**2**
35	**5**
7	**7**
1	

Input

There will be multiple cases. Each test case will be contained on one line. Each line contain one number n where 2<n<1000.

Output

For each test case you should print one line of output which contains the answer of the decomposition with the format showed below.

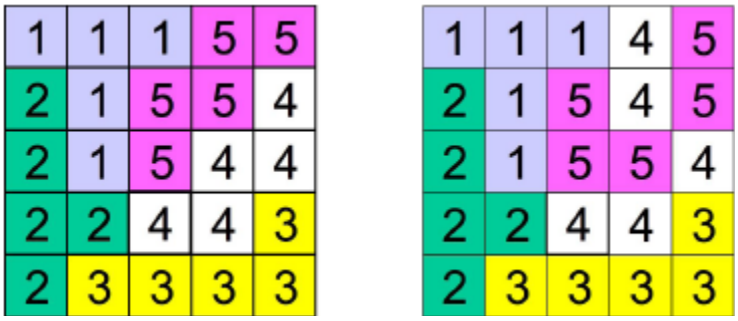
Sample input	Sample Output
70 100 150	2^1 * 5^1 * 7^1 2^2 * 5^2 2^1 * 3^1 * 5^2

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 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	wrong good wrong

Problem 4

Perfect Number

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

In mathematics, a perfect number is defined as a positive integer which is the sum of its proper positive divisors, that is, the sum of the positive divisors excluding the number itself.

The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6.

The next perfect number is 28 = 1 + 2 + 4 + 7 + 14.

Input

The input contains several test cases. Each line contain one number n where 1<n<100000.

Output

For each test case your program must write one line, containing the word “Perfect Number”, indicating a perfect number or the word “No Perfect Number” indicating a number that is not a perfect number.

Sample input	Sample Output
6 8 20 28	Perfect Number No Perfect Number No Perfect Number Perfect Number

Problem 5

Age

Nombre del archivo fuente: age.c, age.cpp or age.java
Entrada: age.in
Salida: Salida Estándar

Juan wants to calcúlate the age of the people he knows given a particulary date. Based in two dates, one is the birth date and otherone a date considerated the current date, make a program to calculate the age in each case.

Input

The input contains several test cases. Each line contains one case. There are 2 dates at each line. The first date is the birth date and the next is the current date. The date format is yyy-mm-dd. Those dates are separated by one space. The current date always will be greater than the birth date.

Salida

For each case, calculate the age

Sample input	Sample Output
1980-02-05 2008-05-06 1940-05-07 2000-03-04	28 59

Problem 6

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