VII MARATON INTERNA DE PROGRAMACION

UNIVERSIDAD KONRAD LORENZ

26 de Febrero de 2010

PROBLEMAS

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

Problem 1

Prime Numbers

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

Definition

Prime number is which could be dividing only for 1 and for itself. You must find the biggest prime in a set of numbers

Input

There will be multiple cases. Each test case will be contained on one line. Each line will contain two integer numbers. First number is lower than second number

Output

For each test case you should print the prime numbers. If one case does not contain any prime number you must print the word No.

Sample input	**Sample Output**

5 20
20 25
8 10
	5 7 11 13 17 19
	23
	No

Problem 2

Marbles on a Tree

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

n boxes are placed on the vertices of a rooted tree, which are numbered from 1 to n, 1 <= n <= 10000. Each box is either empty or contains a number of marbles; the total number of marbles is n.

The task is to move the marbles such that each box contains exactly one marble. This is to be accomplished by a sequence of moves; each move consists of moving one marble to a box at an adjacent vertex. What is the minimum number of moves required to achieve the goal?

Input

The input contains a number of cases. Each case starts with the number n followed by n lines. Each line contains at least three numbers which are: v the number of a vertex, followed by the number of marbles originally placed at vertex v followed by a number d which is the number of children of v, followed by d numbers giving the identities of the children of v.

The input is terminated by a case where n = 0 and this case should not be processed.

Output

For each case in the input, output the smallest number of moves of marbles resulting in one marble at each vertex of the tree.

Sample input	**Sample Output**
9	7
1 2 3 2 3 4	14
2 1 0	20
3 0 2 5 6	
4 1 3 7 8 9	
5 3 0	
6 0 0	
7 0 0	
8 2 0	
9 0 0	
9	
1 0 3 2 3 4	
2 0 0	
3 0 2 5 6	
4 9 3 7 8 9	
5 0 0	
6 0 0	

7 0 0
8 0 0
9 0 0
9
1 0 3 2 3 4
2 9 0
3 0 2 5 6
4 0 3 7 8 9
5 0 0
6 0 0
7 0 0
8 0 0
9 0 0
0	

Problem 3

3n + 1

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

Consider the following algorithm to generate a sequence of numbers. Start with an integer n. If n is even, divide by 2. If n is odd, multiply by 3 and add 1. Repeat this process with the new value of n, terminating when n = 1. For example, the following sequence of numbers will be generated for n = 22:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured (but not yet proven) that this algorithm will terminate at n = 1 for every integer n. Still, the conjecture holds for all integers up to at least 1,000,000.

For an input n, the cycle-length of n is the number of numbers generated up to and including the 1. In the example above, the cycle length of 22 is 16. Given any two numbers i and j, you are to determine the maximum cycle length over all numbers between i and j, including both endpoints.

Input

The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 1,000,000 and greater than 0.

Output

For each pair of input integers i and j, output i, j in the same order in which they appeared in the input and then the maximum cycle length for integers between and including i and j. These three numbers should be separated by one space, with all three numbers on one line and with one line of output for each line of input.

Sample input	**Sample Output**

1 10
100 200
201 210
	1 10 20
	100 200 125
	201 210 89

Problem 4

Fair Power Off

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

During the last power crisis in Colombia (caused by a shortage of rain and hence low levels in the hydro dams), a contingency scheme was developed to turn off the power to areas of the country in a systematic, totally fair, manner. The country was divided up into N regions (Amazonas was region number 1, and Bogota number 13). A number m would be picked, and the power would first be turned off in region 1 (clearly the fairest starting point) and then in every m'th region after that, wrapping around to 1 after N, and ignoring regions already turned off. For example, if N = 17 and m = 5, power would be turned off to the regions in the order:1,6,11,16,5,12,2,9,17,10,4,15,14,3,8,13,7.

The problem is that it is clearly fairest to turn off Bogota last (after all, that is where the electricity headquarters are), so for a given N, the number m needs to be carefully chosen so that region 13 is the last region selected.

Write a program that will read in the number of regions and then determine the smallest number m that will ensure that Bogota (region 13) can function while the rest of the country is blacked out.

Input

Input will consist of a series of lines, each line containing the number of regions (N) with $13 \leq N < 100$. The file will be terminated by a line consisting of a single 0.

Output

Output will consist of a series of lines, one for each line of the input. Each line will consist of the number m, according to the above scheme.

Sample input	**Sample Output**
17 0	7

Problem 5

Age

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

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.

Output

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

Perfect Number

Source file name: perfect.c, perfect.cpp or perfect.java
Input: standar input
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