UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS

II MARATON INTERNA DE PROGRAMACION

UNIVERSIDAD DISTRITAL

27 de Mayo de 2008

PROBLEMAS

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

Problema 1

LISTA TELEFONICA

Nombre del archivo fuente: lista.c, lista.cpp or lista.java
Entrada: lista.in
Salida: standar output

Dada una lista de números telefónicos, determinar si es consistente en el sentido que ningún numero es prefijo de otro. Por ejemplo dados los siguientes números:

- Emergencia 911
- Pedro 97 625 999
- Caro 91 12 54 26

En este caso, no es posible llamar a Caro porque el número inicia con 911 y este es el número de emergencia. Entonces esta lista no es consistente.

Entrada

La primera línea tiene un numero entero, 1 <= t <= 40, que indica el numero de casos de prueba. Cada caso inicia con n, que indica el numero de números telefónicos de la lista donde 1 <= n <= 10000. Las siguientes n líneas con un único número telefónico en cada línea tiene los diferentes números telefónicos. Cada número telefónico tiene máximo 10 dígitos.

Salida

Para cada caso la salida es “YES” si la lista es consistente o “NO” si no lo es.

Ejemplo de Entrada	Ejemplo de Salida
2 3 911 97625999 91125426 5 113 12340 123440 12345 98346	NO YES

Problema 2

PARES O IMPARES

Nombre del archivo fuente: pares.c, pares.cpp or pares.java
Entrada: pares.in
Salida: standar output

Dado un conjunto de números enteros, determinar si la cantidad de números pares es mayor que la cantidad de número impares.

Entrada

La entrada de compone de multiples casos. Cada caso inicia con un numero entero n donde 1 <= n <= 100, que indica la cantidad de números de cada caso. Las n siguientes líneas, corresponden a los n números de cada caso. Después del último caso, esta el numero 0.

Salida

Para cada caso, se debe imprimir el mensaje CASO n: donde n es el número del caso. Si la cantidad de números pares es mayor que de impares, se debe imprimir el mensaje “Hay mayor cantidad de numeros pares”. Pero si es al revés se debe imprimir el mensaje “Hay mayor cantidad de numeros pares”. Finalmente si hay igual cantidad de números pares e impares se debe imprimir el mensaje “Igual cantidad de numeros pares e impares”

Ejemplo de Entrada	Ejemplo de Salida
4 10 15 20 25 3 11 12 13 2 10 20 0	CASO 1: Igual cantidad de numeros pares e impares CASO 2: Hay mayor cantidad de numeros impares CASO 3: Hay mayor cantidad de numeros pares

Problema 3

FACTORIAL

Nombre del archivo fuente: factorial.c, factorial.cpp o factorial.java
Entrada: factorial.in
Salida: Salida Estándar

Definición

n!!..! = n(n-k)(n-2k)..(n mod k), si k no divide a n
n!!..! = n(n-k)(n-2k)..k, si k divide n (Hay k marcas ! en ambos casos).

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

Dado un número n y k, se debe calcular un valor de la expresión de la definición dada.

Entrada

Hay múltiples casos. Cada caso esta contenido en una línea. Cada línea empieza con el número n, seguido de espacio y luego de k marcas de exclamación.

Salida

Por cada caso se debe imprimir un numero que contenga el resultado de n!!..! (Donde hay k marcas !)

Ejemplo de Entrada	Ejemplo de Salida
3 ! 10 !!! 9 !!	6 280 945

Problem 4

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 5

CHECK THE CHECK

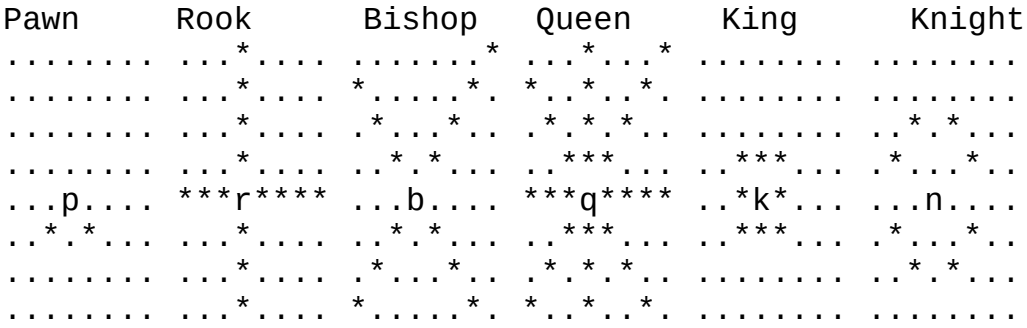
Source file name: check.c, check.cpp or check.java
Input: check.in
Output: standar output

Your task is to write a program that reads a chessboard configuration and identifies whether a king is under attack (in check). A king is in check if it is on square which can be taken by the opponent on his next move.

White pieces will be represented by uppercase letters, and black pieces by lowercase letters. The white side will always be on the bottom of the board, with the black side always on the top.

For those unfamiliar with chess, here are the movements of each piece:
Pawn (p or P): can only move straight ahead, one square at a time. However, it takes pieces diagonally, and that is what concerns you in this problem.
Knight (n or N): has an L-shaped movement shown below. It is the only piece that can jump over other pieces.
Bishop (b or B): can move any number of squares diagonally, either forward or backward.
Rook (r or R): can move any number of squares vertically or horizontally, either forward or backward.
Queen (q or Q): can move any number of squares in any direction (diagonally, horizontally, or vertically) either forward or backward.
King (k or K): can move one square at a time in any direction (diagonally, horizontally, or vertically) either forward or backward.

Movement examples are shown below, where "*" indicates the positions where thepiece can capture another piece:



Remember that the knight is the only piece that can jump over other pieces. The pawn movement will depend on its side. If it is a black pawn, it can only move one square diagonally down the board. If it is a white pawn, it can only move one square diagonally up the board. The example above is a black pawn, described by a lowercase "p". We use "move" to indicate the squares where the pawn can capture another piece.

Input

There will be an arbitrary number of board configurations in the input, each consisting of eight lines of eight characters each. A "." denotes an empty square, while upperand lowercase letters represent the pieces as defined above. There will be no invalid characters and no configurations where both kings are in check. You must read until you find an empty board consisting only of "." characters, which should not be processed.
There will be an empty line between each pair of board configurations. All boards, except for the empty one, will contain exactly one white king and one black king.

Output

For each board configuration read you must output one of the following answers:
Game #d: white king is in check.
Game #d: black king is in check.
Game #d: no king is in check.
where d stands for the game number starting from 1.

Sample input	Sample Output
..k..... ppp.pppp .... .R...B.. .... PPPPPPPP K....... rnbqk.nr ppp..ppp ....p... ...p.... .bPP.... .....N.. PP..PPPP RNBQKB.R	Game #1: black king is in check. Game #2: white king is in check.

Problem 6

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