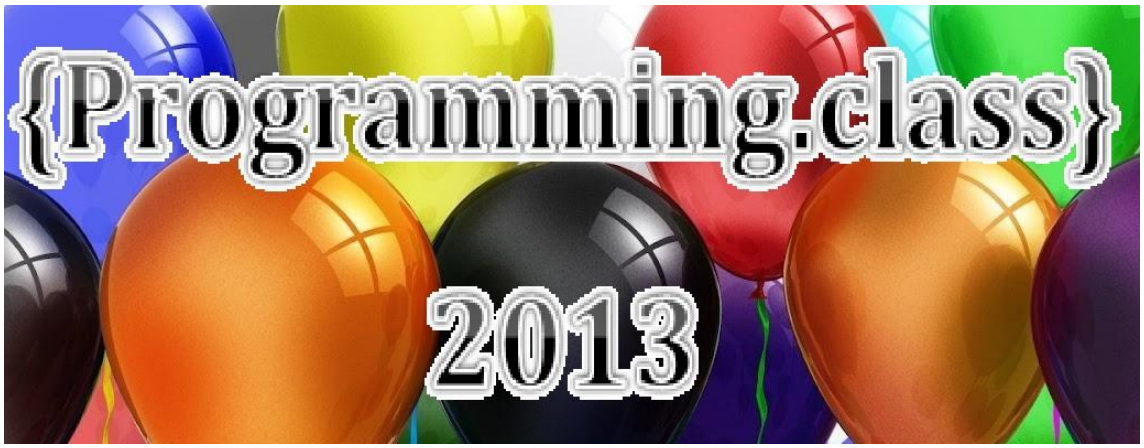

Ejercicios entrenamiento para el concurso regional Programming.class 2013 que se celebrará en el IES Augustóbriga de Navalmoral de la Mata en el mes de Abril del 2013

Con el apoyo de:

IES AUGUSTÓBRIGA

I.E.S. ALBARREGAS

Instituto de Educación Secundaria Ágora Cáceres

IES Donoso Cortés Don Benito

IES Instituto Castelar Badajoz

En este documento se especifican todos los detalles de cada uno de los ejercicios de la ***PRIMERA ENTREGA*** que están subidos al juez online.

Los ejercicios no están colocados por orden de dificultad, es decir, puede ser que el último ejercicio de este documento sea más sencillo de solucionar que el primero.

Cada ejercicio tiene un nombre y, precediéndole y entre paréntesis, un identificador. Este identificador es el que utilizaremos en el juez online para encontrarlo.

Para acceder al juez utilizaremos la siguiente dirección:

http://80.36.53.96/domjudge/team/

Los usuarios y contraseñas para poder subir ejercicios al juez se proporcionarán por correo electrónico.

TABLA DE EJERCICIOS

(SNAR) Sistema numérico en la Antigua Roma

Descripción del problema

Los números del antiguo sistema de notación romano aún se utilizan para algunos propósitos. Los símbolos básicos comunes y sus equivalencias decimales son:

- M = 1000
- D = 500
- C = 100
- L = 50
- X = 10
- V = 5
- I = 1

Algunas combinaciones posibles:

- IV es 4
- IX es 9
- XV es 15
- XL es 40
- LV es 55
- XC es 90
- CCC es 300
- CD es 400
- DV es 505
- CM es 900

La conversión a números romanos se realiza de acuerdo a las siguientes reglas:

- Llamaremos símbolos de tipo 5 a V, L y D. Llamaremos símbolos de tipo 1 a I, X, C y M
- Los símbolos se escribirán de izquierda a derecha, de mayor a menor valor, salvo en el caso que deba producir una resta.
- No se permite la repetición de un mismo símbolo de tipo 5: VV, LXL, DMD, no son opciones válidas.
- Solo se permite la resta de un símbolo de tipo 1 sobre el inmediato superior de tipo 1 o 5: IV, IX, XL, XC, CD, CM, son opciones permitidas
- No se puede repetir ningún dígito secuencialmente más de tres veces: IIII no esta permitido ha de ser IV, MMMM no es válido, CCCC no es válido, será CD
- Un símbolo de tipo 5 no puede estar a la izquierda de uno de mayor valor: DM, VM, LD no están permitidos

Ejemplos de conversiones:

1989 produce: *M* -(1000) *CM* -(900) *LXXX* -(80) *IX* -(9), o lo que es lo mismo: *MCMLXXXIX*

Desarrollar un programa que convierta un número entero positivo menor de 4000 en notación romana

Entrada

Se podrán introducir un número comprendido entre `1` y `3999`

El proceso concluirá si el número introducido está fuera del rango (menor de 1 ó mayor de 3999)

Salida

La salida reflejará el resultado de la conversión en un formato de: `numero_arabigo-numero_romano`.

Nótese que no hay espacios antes ni después del guión

Entrada de ejemplo

```
4
93
387
718
1154
1582
2055
2437
2761
2934
3144
3353
3594
3861
0
```

Salida de ejemplo

```
4-IV
93-XCIII
387-CCCLXXXVII
718-DCCXVIII
1154-MCLIV
1582-MDLXXXII
2055-MMLV
2437-MMCDXXXVII
2761-MMDCCLXI
2934-MMCMXXXIV
3144-MMMCXLIV
3353-MMMCCCLIII
3594-MMMDXCIV
3861-MMMDCCCLXI
```

(BISIESTO) Año bisiesto

Descripción del problema

En el cómputo del tiempo, para hacer coincidir el calendario con el solar se estableció para el calendario juliano un periodo anual de 365 días más 1 día cada cuatro años lo que convertía el cómputo en 365,25 al año. Pero se producía un desfase frente al calendario solar (365,242198). Para solucionar el desfase, ya en la época gregoriana se estableció que cada 100 años no se acumularía un día, y finalmente para un mayor ajuste, cada 400 años si se haría. De esta forma se llega a un año de 365,2425 días, algo mucho más exacto.

Entrada

Un año es bisiesto si es divisible entre 4, salvo el último de cada siglo (el divisible por 100), a excepción de que este sea divisible por 400.

En la entrada se introducirán números naturales y positivos que corresponderán a años entre 1800 y 9999 hasta que se introduzca un 0.

Salida

La salida reflejará un SI si el número es bisiesto y un NO si no lo es.

La salida informará si el número introducido está fuera del rango de 1800 a 9999 con un FUERA DE RANGO y leerá la siguiente línea

Entrada de ejemplo

```
1999
1968
2000
1800
12
1900
0
```

Salida de ejemplo

```
NO
SI
SI
NO
FUERA DE RANGO
NO
```

(CAPICUA) Números capicúas

Descripción del problema

Un número es capicúa cuando al invertir dicho número continua siendo el mismo número. 7557 es capicúa ya que cuando sus cifras se invierten generan el mismo número, 7547 no es capicúa pues cuando se invierten sus cifras se obtiene el 7457 que, evidentemente, no es el mismo número.

Diseñar un programa que indique si un número es capicúa o no. Nos interesan los números de dos cifras o más, no se introducirán nunca en nuestro programas números de una cifra

Entrada

Se realizará la introducción de una serie de números comprendidos entre el 10 y el 50000.

El primer número introducido indicará cuantas comprobaciones se realizarán y los siguientes números enteros son los números a evaluar.

Las operaciones se realizaran siempre sobre valores absolutos y, por lo tanto no se considerará el signo en valores negativos.

Salida

Como salida se mostrará un SI si el número es capicúa o un NO si no lo es.

Entrada de ejemplo

```
5
236
-1221
6987
454
13
```

Salida de ejemplo

```
NO
SI
NO
SI
NO
```

(CIFRAS) Calcular el número de cifras de un número entero

Descripción del problema

Diseñar un programa que, dada una secuencia de números enteros, indique cuántos dígitos tiene cada uno de ellos. Las operaciones se realizarán siempre sobre valores absolutos.

Entrada

Se introduce una secuencia de números enteros que termina con un número de una sola cifra, que *No* se reflejará en la solución.

Salida

La salida reflejará los dígitos que tienen cada uno de los números introducidos, en un formato de: `numero_introducido=cantidad_digitos.`

Entrada de ejemplo

```
236
-1258
6987
458
13
-50
4
```

Salida de ejemplo

```
236=3
-1258=4
6987=4
458=3
13=2
-50=2
```

(LETR_DNI) Comprobación de DNI

Descripción del problema

Se desea realizar un programa que compruebe la validez de un conjunto de Documentos Nacionales de Identidad de España. Para comprobar la validez de cada uno de los documentos se debe verificar que sea un número de 8 dígitos, aplicar un algoritmo para obtener la letra que corresponde al número indicado y comprobar si la letra que se proporciona junto al número es correcta. En caso de que la comprobación sea válida el programa debe mostrar `SI` y en caso contrario, cuando no sea válida, deberá mostrar `NO`. La comprobación de los DNI terminará cuando la entrada sea únicamente un 0.

El Documento Nacional de Identidad se debe proporcionar sin espacios en blanco ni otros caracteres de separación.

Un DNI válido estará formado por ocho dígitos numéricos seguidos de una letra. Cualquier otra entrada que no siga esta especificación será considerada no válida. El número máximo de caracteres que puede tener cada entrada será 20. Además la letra que va al final de la entrada debe ser la que se corresponda con el cálculo de la letra del DNI del algoritmo que presentamos a continuación:

- La letra del DNI se obtendrá al calcular el resto de dividir el valor numérico del DNI entre 23. El valor del resto nos indicará la letra a emplear según la siguiente tabla:

Resto	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Letra	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

Entrada

Se pasarán como entradas cadenas alfanuméricas de longitud variable. La longitud máxima de la cadena será de 20 caracteres. El proceso terminará cuando la entrada sea únicamente un 0.

Salida

Para cada una de las entradas, excepto para la de terminación, deberá mostrar la palabra `SI` cuando la entrada sea un DNI válido y `NO` cuando la entrada no sea válida.

Entrada de ejemplo

```
14238290W
003217283
04170564C
7283E2
23423415P
23423415J
234234P
0
```

Salida de ejemplo

```
SI
NO
SI
NO
SI
NO
NO
```

(ISOSCELE) Triángulo Isósceles

Descripción del problema

Realizar un programa que dibuje triángulos isósceles con caracteres que le introducirá el usuario. Dicho triángulo estará formado por tantas líneas de caracteres como el usuario le indique.

Entrada

Se le pasará como entrada el número de líneas de las que constará el triángulo, seguida del carácter que deberá imprimir para formarlo. El programa finalizará cuando se introduzca un triángulo formado por 0 líneas y cualquier carácter.

Salida

El programa devolverá los triángulos isósceles dibujados, con un vértice en la parte superior y la base abajo. No existen líneas en blanco entre los triángulos ni tampoco después del último triángulo.

Entrada de ejemplo

```
2 a
6 q
8 p
0 0
```

Salida de ejemplo

```
 a
aaa
     q
    qqq
   qqqqq
  qqqqqqq
 qqqqqqqqq
qqqqqqqqqqq
       p
      ppp
     ppppp
    ppppppp
   pppppppppp
  ppppppppppppp
 pppppppppppppp
ppppppppppppppp
```

(FIBONACI) Fibonacci

Descripción del problema

La serie de Fibonacci es una serie infinita de números enteros donde el primer y el segundo elemento de la serie tienen el valor 1, y el resto de los elementos se calcula sumando sus dos predecesores. De esta forma el tercer elemento de la serie, o Fibonacci de 3, es la suma del elemento 1 y del elemento 2. El cuarto elemento es la suma del tercer elemento y del segundo y así sucesivamente.

Serie de Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Realizar un programa que calcule tantos elementos de la serie de Fibonacci como indique la entrada. El proceso se repetirá hasta que se introduzca el número `0`.

Entrada

El programa tendrá como entrada el número de elementos de la serie de Fibonacci que desea que se muestren. Debe introducirse un número real entero positivo que estará comprendido entre `1` y `45`. La entrada finalizará cuando se introduzca el valor `0`

Salida

El programa mostrará tantos números, separados por un espacio en blanco, de la serie de Fibonacci como indique la entrada.

Entrada de ejemplo

```
1
2
3
7
0
```

Salida de ejemplo

```
1
1 1
1 1 2
1 1 2 3 5 8 13
```

(DIB_ROMB) Dibujar Rombos

Descripción del problema

Este ejercicio consiste en realizar un programa que imprima por pantalla rombos concéntricos `dibujados con el caracter '*'.`

Entrada

El programa solicitará, como dato de entrada, la longitud del lado del rombo más externo (es decir, su número de caracteres que como máximo será de 200).

Salida

El rombo siempre debe quedar ajustado a la izquierda, no habrá separación entre un rombo y el siguiente, ni espacios en blanco al final de cada línea (después del último asterisco). El programa no deberá imprimir nada para tamaños negativo y finaliza la ejecución cuando encuentra 0 a la entrada.

Entrada de ejemplo

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

Salida de ejemplo

```
*
 *
***
 *
  *
 ***
*****
 ***
  *
   *
  ***
 *****
*******
 *****
  ***
   *
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

(SUM_DIGI) Suma de dígitos

Descripción del problema

Hemos sido seleccionados como miembros de un nuevo cuerpo de programadores. Es necesario demostrar nuestros conocimientos realizando de forma eficaz distintos algoritmos de programación. Uno de ellos consiste en sumar los dígitos de cualquier número natural introducido.

Entrada

En la entrada, el valor de la primera línea, será la cantidad de números sobre los que calcular la suma de sus dígitos. El resto de valores, uno por línea, serán números naturales (es decir sin decimales y positivos) sobre los que realizar dicha operación.

Salida

Por cada número en la entrada, exceptuando el primero, el programa dará como resultado una línea con la suma de sus dígitos.

Entrada de ejemplo

```
4
29
7
0
1020
```

Salida de ejemplo

```
11
7
0
3
```

(DIB_TRIA) Dibujar Triángulos

Descripción del problema

Deseamos realizar un programa que dibuje triángulos rectángulos con caracteres que le introducirá el usuario. Dicho triángulo estará formado por tantas líneas de caracteres como el usuario le indique.

Entrada

Se le pasará como entrada el número de líneas de las que constará el triángulo, seguida del carácter que deberá imprimir para formarlo. El programa finalizará cuando se introduzca un triángulo formado por `0` líneas

Salida

El programa devolverá los triángulos dibujados. Entre un triángulo y otro deberá haber un *único salto de línea*. Deberá ocurrir lo mismo al final del último triángulo. *No deberá existir* ningún salto de línea antes del primer triángulo.

Entrada de ejemplo

```
2 a
6 q
9 *
17 p
0
```

Salida de ejemplo

```
a
aa

q
qq
qqq
qqqq
qqqqq
qqqqqq

*
**
***
****
*****
******
*******
********
*********

p
pp
ppp
pppp
ppppp
pppppp
ppppppp
pppppppp
ppppppppp
pppppppppp
ppppppppppp
pppppppppppp
ppppppppppppp
pppppppppppppp
ppppppppppppppp
pppppppppppppppp
```