MONEDA

Se desea elaborar un algoritmo para transformar una cantidad de euros al número mínimo de billetes y monedas necesarios para representarla. La cantidad siempre será positiva y sin decimales.

Entrada

El algoritmo recibirá números enteros positivos. El algoritmo terminará la transformación cuando reciba una cantidad de 0 euros.

Salida

Para cada uno de los valores que se reciban, excepto para el 0, la salida será la cantidad a transformar y en cada una de las líneas inferiores se colocará en número mínimo de billetes o monedas de cada clase que se necesitan para conseguir dicha cantidad. El formato de la salida será de la siguiente forma:

99
1 de 50
2 de 20
1 de 5
2 de 2

Entrada de ejemplo

354
1233
941
5995
4739
0

Salida de ejemplo

354
1 de 200
1 de 100
1 de 50
2 de 2
1233
2 de 500
1 de 200
1 de 20
1 de 10
1 de 2
1 de 1
941
1 de 500
2 de 200
2 de 20

1 de 1
5995
11 de 500
2 de 200
1 de 50
2 de 20
1 de 5
4739
9 de 500
1 de 200
1 de 20
1 de 10
1 de 5
2 de 2

ORDENACIÓN DE SECUENCIAS.

Se entiende por secuencias a un conjunto de números distintos, es decir la secuencia terminará cuando se encuentre con un número que ya pertenece a dicha secuencia, en cuyo caso comenzará una nueva secuencia, o se hayan leído todos los números.

Deseamos realizar un programa que reciba una sucesión de números enteros, el primero de los cuales indicará la cantidad de números que se van a leer, y realice la ordenación de las secuencias.

Cuando encontremos una secuencia ordenaremos, de menor a mayor, todos los elementos perteneciente a dicha secuencia.

Una vez ordenadas todas las secuencias se mostrarán por pantalla todos los números ordenados en sus secuencias.

Entrada

El programa recibirá una sucesión de números enteros, el primero de los cuales indicará cuantos números se van a introducir.

Salida

El programa deberá mostrar los números enteros ordenados en sus secuencias respectivas

Entrada de ejemplo

9
5
4
3
6
3
7
1
7
5

Salida de ejemplo

3
4
5
6
1
3
7
5
7

AÑO BISIESTO.

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. 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 que se acumularía el día. De esta forma se llega a un año de 365,2425 mucho más exacto.

Entrada

Un año es bisiesto si es divisible entre 4, excepto el último de cada siglo (el divisible por 100), salvo 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 solicitará nueva introducción

Entrada de ejemplo

1999
1968
2000
1800
12
1900
0

Salida de ejemplo

NO
SI
SI
NO
FUERA DE RANGO
NO

NÚMEROS CAPÍCUAS.

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

Entrada

Se realizará la introducción de una serie de números

El primer número introducido indicará cuantas comprobaciones se realizaran y los siguientes números enteros son los números incógnita.

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

Salida

Se habrán de informar con un *SI* o un *NO* indicando que el número es o no capicúa.

Entrada de ejemplo

5
236
-1221
6987
454
13

Salida de ejemplo

NO
SI
NO
SI
NO

COMPROBACIÓN DEL DNI.

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

MATRIZ CONCÉNTRICA.

Una matriz es concéntrica si, para todos sus anillos, los números correspondientes a cada anillo son iguales.

Ejemplo:

Matriz Concéntrica

3	3	3	3	3
3	7	7	7	3
3	7	5	7	3
3	7	7	7	3
3	3	3	3	3

Matriz NO Concéntrica

3	3	3	3	3
3	7	7	**2**	3
3	7	5	7	3
3	7	7	7	3
3	3	3	3	3

Realizar un algoritmo que reciba una matriz de 8 x 8 y muestre si se trata de una matriz concéntrica o no.

Entrada

La entrada constará de matrices de 8 x 8 hasta que el primer dato de una matriz sea un cero.

Salida

La salida será Si, si la matriz es concéntrica y No si no lo es.

Entrada de ejemplo

1 2 3 4 5 6 7 8
2 3 4 5 6 7 8 9
12 3 4 5 6 77 8 9
3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8
2 3 4 5 6 7 8 9
3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8
3 3 3 3 3 3 3 3
3 5 5 5 5 5 5 3
3 5 7 7 7 7 5 3
3 5 7 9 9 7 5 3
3 5 7 9 9 7 5 3
3 5 7 7 7 7 5 3
3 5 5 5 5 5 5 3

3 3 3 3 3 3 3 3
30 30 30 30 30 30 30 30
30 50 50 50 50 50 50 30
30 50 70 70 70 70 50 30
30 50 70 90 90 70 50 30
30 50 70 90 90 70 50 30
30 50 70 70 70 70 50 30
30 50 50 50 50 50 50 30
30 30 30 30 30 30 30 30
3 3 3 3 3 3 3 3
3 5 2 5 5 5 5 3
3 5 7 7 7 7 5 3
3 5 7 9 9 7 5 3
3 5 7 9 9 7 5 3
3 5 7 7 7 7 5 3
3 5 5 5 5 5 5 3
3 3 3 3 3 3 3 3
0

Salida de ejemplo

No
Si
Si
No

SISTEMA NUMÉRICO EN LA ANTIGUA ROMA.

Los números del antiguo sistema romano de notación 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

Ejemplos de conversiones:

1989 produce: *M* -(1000) *CM* -(900) *LXXX* -(80) *IX* -(9) *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 La conversión a números romanos se realiza de acuerdo a las siguientes reglas:

- 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 una misma letra de tipo 5 (V, L, D) (VV, LXL, DMD, no son opciones válidas).
- Solo se permite la resta de un símbolo de tipo 1 (I, X, C, M) sobre el inmediato superior de tipo 1 o 5 (IV, IX, XL, XC, CD, CM, son opciones permitidas)
- No se puede repetir ningún digito secuencialmente más de tres veces (IIII no está 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)

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"*.

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
```