1

¿A qué hora pasa el próximo tren?

Todo el que ha estado alguna vez esperando que llegue el tren se ha planteado lo que podría haber hecho si hubiese sabido a qué hora pasaría. Es entonces cuando envidiamos a los usuarios que lo utilizan con frecuencia y son capaces de llegar a la estación con solo un minuto de antelación. ¿Por qué siempre confiamos en la suerte y no comprobamos a qué hora pasará antes de vernos ya en el andén?

Hoy, al llegar a la estación y ver que de nuevo tendrás que esperar, has decidido hacer una aplicación que te permita saber en cualquier momento a qué hora pasará el próximo tren. Para empezar, en el tiempo que has estado esperando ya has conseguido el horario de los trenes de las estaciones que utilizas a menudo. Ahora, ya será muy fácil calcular cuándo pasará el próximo tren.

Requisitos de implementación

Se debe utilizar una clase `horas` con la representación que se considere más oportuna. Como mínimo se debe implementar la sobrecarga del operador < como función miembro de la clase. La sobrecarga de los operadores de inserción (<<) y extracción (>>) de las horas se hará con funciones externas a la clase. El constructor de la clase debe comprobar que los datos son correctos ($0 \leq$ *horas* ≤ 23; $0 \leq$ *minutos, segundos* ≤ 59) y lanzar una excepción si no lo son.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea con dos valores: el número de trenes que sale de la estación y el número de horas que se van a querer consultar. A continuación se muestra en una línea el horario de trenes de la estación en orden creciente. Para cada tren se dice la hora, minutos y segundos en que saldrá de la estación (en el formato `HH:MM:SS`). A continuación se listan las horas que se van a consultar, una en cada línea y no necesariamente ordenadas.

El número de trenes que sale de la estación es siempre mayor que cero y menor que 1000. Se garantiza que las fechas del horario de trenes son correctas y están en orden creciente; sin embargo, las fechas que se consultan pueden ser incorrectas.

La entrada termina con `0 0`.

Salida

Para cada caso de prueba se escribirá una línea por cada hora consultada, indicando la hora de salida (con el mismo formato que en la entrada) del primer tren con salida posterior o igual a la hora consultada, si existe . Si no existe, se escribirá `NO`. Si la fecha de entrada es incorrecta se escribirá `ERROR`. Cada caso termina con tres guiones (`---`).

Entrada de ejemplo

```
4 2
06:40:30 12:50:00 19:20:00 21:25:00
10:20:00
22:00:00
6 3
00:00:00 09:30:00 16:40:30 17:00:00 20:10:40 22:35:00
20:10:40
08:62:30
08:40:30
0 0
```

Salida de ejemplo

```
12:50:00
NO
---
20:10:40
ERROR
09:30:00
---
```

Autores: Isabel Pita y Alberto Verdejo.

2

Una tarde de sábado

Me han llamado unos amigos para ver una película en el cine este sábado. Entre todos han elaborado una lista con sus preferencias y me han dicho que decida cuál de ellas vamos a ver. Como ya he quedado el sábado para cenar con otros amigos tengo que elegir una película que termine antes de la hora de la cena. Al mirar la cartelera, he visto que solo pone la hora de comienzo y la duración de la película, pero no la hora en que termina. Como no quiero equivocarme le he pedido a mi hermano que me ayude a hacer un programa que lo calcule. Me ha dicho que si le doy la hora a la que empieza cada película y lo que dura me mostrará una lista de todas las películas ordenadas por la hora a la que terminan. De esta forma podré ver fácilmente cuáles son las que puedo ir a ver y el tiempo que me quedará libre con cada una.

Requisitos de implementación.

En la implementación del problema se debe utilizar la clase `horas` implementada en el problema anterior. Se le añadirá el operador suma (`+`), que lanzará una excepción cuando la suma de las horas exceda del día actual.

Se utilizará una clase `pelicula` que guardará los datos relativos a las películas. Esta clase necesita el operador menor para poder ordenar, por tiempo de finalización, el vector que almacenará las películas.

El vector se puede ordenar utilizando la función `sort` de la librería `algorithm`.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con el número de películas que vienen a continuación. Cada película se describe en una línea, primero se indica la hora de comienzo, a continuación su duración, y por último el título de la película. El formato de la hora de inicio y de la duración es el mismo: `HH:MM:SS`. La entrada termina con un caso con 0 películas.

Se supone que todas las horas son correctas y que ninguna película terminará después de media noche.

Salida

Para cada caso de prueba se escribirá una línea por cada película, indicando la hora de finalización y el nombre de la película. Las películas se mostrarán ordenadas en orden creciente de finalización. Si varias películas terminan a la misma hora, se muestran ordenadas por orden alfabético de su título. Cada caso termina con tres guiones (`---`).

Entrada de ejemplo

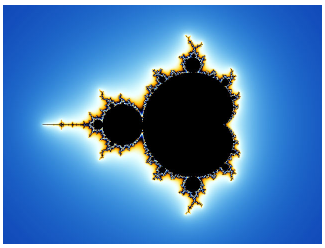
```
3
17:40:20 02:20:10 El hombre tranquilo
15:30:00 01:35:40 12 hombres sin piedad
20:40:10 01:55:10 Horizontes lejanos
3
15:10:00 01:20:00 Gremlins
13:10:00 03:20:00 Ben-Hur
12:00:50 00:59:10 La soga
0
```

Salida de ejemplo

```
17:05:40 12 hombres sin piedad
20:00:30 El hombre tranquilo
22:35:20 Horizontes lejanos
---
13:00:00 La soga
16:30:00 Ben-Hur
16:30:00 Gremlins
---
```

Autores: Isabel Pita y Alberto Verdejo.

3
El conjunto de Mandelbrot



El conjunto de Mandelbrot es uno de los fractales más conocidos y estudiados. Su nombre se debe al matemático Benoit Mandelbrot que lo estudió en los años 70.

Se define como el conjunto de los números complejos c para los cuales la siguiente serie no diverge.

$$z_0 = 0$$
$$z_n = {z_{n-1}}^2 + c$$

En la figura puede verse la representación en el plano complejo de este conjunto. Los puntos representados en negro pertenecen al conjunto; el color del resto de los puntos depende de la velocidad con que la serie diverge.

Queremos saber si un número complejo c pertenece al conjunto de Mandelbrot. Para ello iteraremos la recurrencia anterior n veces. Si en algún momento el módulo del número complejo obtenido es estrictamente mayor que 2 consideramos que la serie es divergente y dejaremos de iterar. Si, por el contrario se alcanza la n-ésima iteración sin que el módulo del número complejo sea mayor de 2 entonces supondremos que la serie no es divergente y que por lo tanto el número complejo pertenece al conjunto de Mandelbrot.

Requisitos de implementación.

En la implementación del problema se utilizará una clase genérica `complejo` cuya representación serán dos valores del tipo del parámetro. Se implementará en la clase un constructor que permita inicializar el número complejo, un operador suma (+), un operador producto (*) y un método que calcule el módulo del número complejo. Estas operaciones se definen como:

(a+bi) + (c+di) = (a+c) + (b+d)i
*(a+bi) * (c+di) = (a*c-b*d) + (a*d+c*b)i*
mod(a+bi) = sqrt(a^2 + b^2).

Entrada

La entrada comienza con un número entero mayor o igual que cero que indica el número de casos de prueba que aparecen a continuación. Cada caso de prueba consta de tres valores separados por blancos. Los dos primeros son números reales (`float`) con dos decimales que indican la parte real y la parte imaginaria del número complejo c. El tercer valor indica el número de veces que se quiere iterar la recurrencia para decidir si la serie es divergente.

Los datos de entrada cumplen que: la parte real e imaginaria del número c pertenecen al intervalo [–2..2] y el número de iteraciones pertenece al intervalo [100..500].

Salida

Para cada caso de prueba se escribe `SI` si el número complejo c pertenece al conjunto de Mandelbrot y `NO` si no pertenece.

Entrada de ejemplo

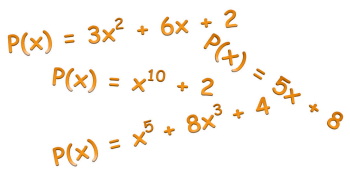
```
2
-1.80 -0.40 400
0.20 -0.20 400
```

Salida de ejemplo

```
NO
SI
```

Autores: Isabel Pita y Alberto Verdejo.

4
Evaluar un polinomio



Un polinomio en una variable es una expresión matemática de la forma $P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0 x^0$. Las constantes a_0,..., a_n se llaman los *coeficientes* del polinomio y los valores 0..n los *exponentes* (número naturales). Cada una de las sumas $a_i x^i$ se denomina un *monomio*. El grado del polinomio es el del monomio de mayor grado.

La operación más básica que podemos realizar con un polinomio es calcular su valor para cierto valor de la variable. En este problema se pide calcular el valor de un polinomio para diversos valores de la variable.

Requisitos de implementación.

Se implementará una clase `polinomio` para representar polinomios con coeficientes enteros. La representación del polinomio será un vector de pares que representen los monomios, de manera que cada par esté formado por un coeficiente y un exponente, y el vector esté ordenado de menor a mayor según el exponente, y todos los exponentes sean distintos.

La clase tendrá al menos un método para añadir un monomio al polinomio, dados su coeficiente y exponente, y otro para evaluar el polinomio dado el valor de la variable.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea en la que aparecen los monomios del polinomio, cada uno formado por un coeficiente y un exponente, con posibles repeticiones de exponentes. Esta línea acaba con `0 0`.

La siguiente línea contiene el número N de valores para los que se quiere evaluar el polinomio. La tercera línea contiene esos N valores.

Los coeficientes y los valores para los que se evalúa el polinomio son números enteros en el intervalo [–10..10]. El grado del polinomio es mayor o igual que 0 y menor que 9.

Salida

Para cada caso se escribirá una línea con los valores a los que se evalúa el polinomio. Estos valores pertenecerán siempre al intervalo [–2*10^9..2*10^9].

Entrada de ejemplo

```
3 2 6 1 2 0 0 0
2
-2 2
5 1 8 0 0 0
3
5 0 -1
1 5 8 3 4 0 0 0
2
2 3
```

Salida de ejemplo

```
2 26
33 8 3
100 463
```

Autores: Alberto Verdejo y Isabel Pita.

5
Los k elementos mayores

Dada una serie de elementos ordenables, queremos encontrar los k elementos distintos mayores.

Requisitos de implementación.

Se modificará la clase genérica `set` vista en clase para que el invariante de la representación imponga que los elementos del array estén ordenados.

Se extenderá la clase añadiendo, al menos, métodos para consultar y eliminar el menor elemento del conjunto.

La complejidad en espacio adicional de la resolución de un caso de prueba debe estar en $O(k)$.

Entrada

Cada caso de prueba está formado por tres líneas. La primera contendrá el carácter `N` si los elementos de la serie son números, o el carácter `P` si los elementos son palabras. La segunda línea contendrá el valor $k > 0$, que será menor o igual que el número de elementos (distintos) de la serie. La tercera línea contendrá los elementos de la serie (posiblemente con repeticiones). Si son números estarán en el rango $[0..10^9]$, y el fin de la serie vendrá indicado con un `-1`. Si son palabras, estarán formadas por no más de 30 caracteres de la '`a`' a la '`z`', y el final de la serie estará indicado con la palabra `FIN`.

Salida

Para cada caso de prueba se escribirá una línea con los k elementos mayores de la serie, sin repeticiones y ordenados de menor a mayor.

Entrada de ejemplo

```
N
3
1 8 3 14 5 -1
P
2
maria luis marta juan alberto FIN
N
3
1 2 3 4 5 6 6 6 -1
```

Salida de ejemplo

```
5 8 14
maria marta
4 5 6
```

Autor: Alberto Verdejo.

Problema número 139

Números cubifinitos

Tiempo máximo: 2,000-3,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=139

Se dice que un número es *cubifinito* cuando al elevar todos sus dígitos al cubo y sumarlos el resultado o bien es 1 o bien es un número cubifinito.

Por ejemplo, el número `1243` es cubifinito, pues al elevar todos sus dígitos al cubo obtenemos `100` que es cubifinito.

Por su parte, el `513` no es cubifinito, pues al elevar al cubo sus dígitos conseguimos el `153` que nunca podrá ser cubifinito, pues la suma de los cubos de sus dígitos vuelve a dar `153`.

Dado un número, se trata de determinar si éste es o no cubifinito.

Entrada

La entrada consta de una serie de casos de prueba terminados con un número `0`. Cada caso de prueba es una línea con un número positivo no mayor que 10^7.

Salida

Para cada caso de prueba se mostrará una única línea en la que aparecerá la serie de transformaciones del número original hasta el `1` o hasta la repetición de un número de la serie. Tras eso se indicará la conclusión a la que se llega: si el número es cubifinito o no.

Mira el ejemplo de la salida para ver el formato esperado exacto.

Entrada de ejemplo

```
1
10
1243
513
0
```

Salida de ejemplo

```
1 -> cubifinito.
10 - 1 -> cubifinito.
1243 - 100 - 1 -> cubifinito.
513 - 153 - 153 -> no cubifinito.
```

Autor: Marco Antonio Gómez Martín.

Revisor: Pedro Pablo Gómez Martín.

Problema número 185

Potitos

Tiempo máximo: 1,000-4,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=185

"¡Javi! ¡Parece que éste tampoco le gusta!" se convirtió en cantinela habitual a la hora de la comida, antes de que el pequeño dejara definitivamente de comer potitos una tarde. Darle de comer era misión imposible; pese a la paciencia de sus padres, el niño había heredado el carácter cabezón del padre y mantenía la boca cerrada, inmune a los vuelos de la cucharita y a las monerías de quienes le rodeaban. Era necesario encontrar una solución.

Se les ocurrió que, quizá, el aparentemente caprichoso rechazo a los potitos se debía a algún ingrediente concreto. Comenzaron a anotar, cuidadosamente, todos los ingredientes de cada potito que le daban, junto con un comentario de si el pequeño se lo había tomado o no. Tras varios días, estaban convencidos de que podrían averiguar cuáles eran los ingredientes que no le gustaban, y así comprar aquellos potitos que fuera a comerse. ¿Puedes ayudarles a encontrarlos?

Entrada

La entrada estará compuesta de múltiples casos de prueba. Cada uno comienza con un número indicando el número de potitos que se han intentado dar al bebé (como máximo 25).

A continuación aparece una línea por cada potito. La línea comienza por `SI:` o `NO:` dependiendo de si el pequeño se comió o no el potito. Después aparece la lista de los ingredientes del potito separados por espacios. La lista se cierra con la palabra `FIN`, que no debe considerarse un ingrediente. Ningún potito tiene más de 10 ingredientes, y todos los ingredientes están compuestos por una única palabra de hasta 20 letras minúsculas.

La entrada acaba con un caso de prueba sin potitos.

Salida

Para cada caso de prueba se deben mostrar, en una línea, los ingredientes que no le gustan al niño, ordenados alfabéticamente y separados por espacio. Si todos los ingredientes le gustan, se dejará la linea en blanco.

Entrada de ejemplo

```
3
SI: patata maiz tomate FIN
NO: patata puerro guisantes pollo FIN
SI: tomate zanahoria puerro pollo calabacin arroz FIN
2
SI: tomate zanahoria pollo calabacin arroz FIN
NO: tomate ternera puerro FIN
0
```

Salida de ejemplo

```
guisantes
puerro ternera
```

Autores: Pedro Pablo Gómez Martín, Marco Antonio Gómez Martín y Patricia Díaz García.

Revisores: Ferran Borrell Micola, Cristina Gómez Alonso y Roger Meix Mañá.

Problema número 232

¡Feliz no cumpleaños!

Tiempo máximo: 2,000 s Memoria máxima: 4096 KiB

http://www.aceptaelreto.com/problem/statement.php?id=232

Cuando Alicia se encontró con el Sombrerero Loco, la Liebre de Marzo y el Lirón, los tres estaban en mitad de la celebración de una fiesta de no cumpleaños. Al principio Alicia no entendía nada, pero, gracias a las explicaciones del Sombrerero Loco, finalmente entendió que un no cumpleaños es una fecha que *no* coincide con la del cumpleaños. Alicia cayó en la cuenta de que también ese día era su no cumpleaños. ¡Qué pequeño es este mundo!

La dificultad del no cumpleaños es ¡saber cuántos se cumplen! Por eso, para no complicarse, en el País de las Maravillas las tartas solían tener siempre una sola vela. Pero a la Reina de Corazones no le gustaba, y, cuando decidió cortar la cabeza a cualquiera que no pusiera el número correcto de velas, se dejaron de celebrar fiestas de no cumpleaños.

¡¡Hay que poner fin a esta sequía de celebraciones!!

Entrada

El programa leerá, de la entrada estándar, múltiples casos de prueba, cada uno en una línea.

Para cada caso de prueba se proporciona la fecha de nacimiento de alguno de los habitantes del País de las Maravillas, junto con la fecha actual, que siempre será posterior. Cada fecha está compuesta del día, mes y año, separados por espacio. Las dos fechas se separan también por un espacio, y siempre serán fechas válidas, aunque podría ocurrir que la segunda no fuera un día de no cumpleaños. El habitante más anciano del País de las Maravillas nació en 1862, y no hay ninguna consulta que supere el año 9999.

La entrada termina con un línea con seis ceros.

Salida

Para cada caso de prueba, el programa escribirá en la salida estándar cuántos no cumpleaños cumple el personaje en la fecha dada. Si el día no es un no cumpleaños, se escribirá `0`.

Ten en cuenta que en el País de las Maravillas, no existen los años bisiestos.

Entrada de ejemplo

```
4 7 1862 24 5 1865
23 6 1912 7 6 1954
17 2 2014 17 2 2015
0 0 0 0 0 0
```

Salida de ejemplo

```
1052
15273
0
```

Autores: Pedro Pablo Gómez Martín y Marco Antonio Gómez Martín.