"First, solve the problem. Then, write the code" John Johnson.

Programa-Me 2017
Final Nacional
Problemas

Patrocinado por

Ejercicios realizados por

Universidad Complutense
de Madrid

Realizado en la **Facultad de Informática (U.C.M.)**
9 de junio de 2017

9 de junio de 2017
http://www.programa-me.com

Listado de problemas

A Sobre la tela de una araña **3**

B Ovejas negras **5**

C Claras y oscuras **7**

D Conseguir un cuadrado perfecto **9**

E Resta y amus **11**

F Michael J. Fox y el Pato Donald **13**

G Nana al bebé de papá y mamá **15**

H RENUM **17**

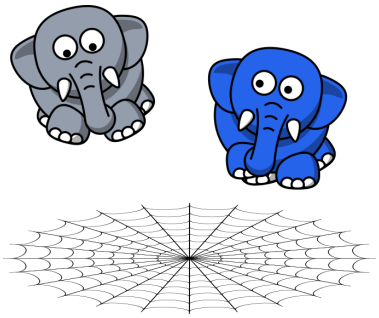
I Telesillas **19**

J El secreto de la bolsa **21**

Autores de los problemas:

- Marco Antonio Gómez Martín (Universidad Complutense de Madrid)
- Pedro Pablo Gómez Martín (Universidad Complutense de Madrid)
- Alberto Verdejo López (Universidad Complutense de Madrid)

A
Sobre la tela de una araña



Un elefante se balanceaba sobre la tela de una araña. Como veía que no se caía, fue a llamar a otro elefante.

Dos elefantes se balanceaban sobre la tela de una araña. Como veían que no se caían, fueron a llamar a otro elefante.

Tres elefantes se balanceaban...

Pero, ¿hasta cuándo? ¿Cuántos elefantes pueden llegar a balancearse encima de una tela de araña antes de que se rompa? La canción no arroja ninguna luz sobre este particular, pero quizá tú puedas ayudarnos.

Entrada

Cada caso, leído de la entrada estándar, consiste en dos líneas. La primera contiene el peso máximo soportado por la tela de araña. La segunda contiene los pesos de cada elefante, separados por espacio, en el orden en el que van llegando, y termina con un `0`. Todos los números son enteros positivos menores que 10^9.

La entrada acaba también con un `0`, que no deberá procesarse.

Salida

Para cada caso de prueba, el programa indicará, en la salida estándar, el número de elefantes que pudieron balancearse en la tela, antes de que ésta se rompiera o de que no hubiera más elefantes dispuestos a arriesgarse.

Los elefantes se van incorporando a la diversión por estricto orden de llegada.

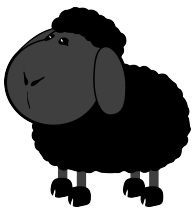
Entrada de ejemplo

```
10
1 2 3 4 5 0
20
3 3 0
30
10 10 20 10 0
0
```

Salida de ejemplo

```
4
2
2
```

B
Ovejas negras



Parece ser que en cierta ocasión estaban de viaje por Escocia un abogado, un físico y un matemático. Por la ventanilla del tren en el que viajaban vieron un campo con ovejas negras. Ninguno de los tres había visto ovejas negras nunca, por lo que se estableció un curioso diálogo entre ellos:

— ¡Vaya! ¡En Escocia las ovejas son negras! — dijo el abogado.

— Querrás decir que en Escocia *algunas* ovejas son negras... — corrigió el físico.

— Bueno, — no pudo evitar decir el matemático — con lo que hemos visto lo único que podemos decir es que en Escocia algunas ovejas son negras... *¡por un lado!*

Para comprobar si el abogado tenía razón o no, se han tomado fotografías a todas las ovejas de Escocia, y ahora hay que analizarlas para ver si en alguna foto sale alguna oveja blanca (al menos por un lado) o no.

Entrada

La entrada estará compuesta por distintos casos de prueba, cada uno siendo una instantánea de una o más ovejas escocesas.

Cada foto comienza con una línea con dos números indicando el ancho y el alto (en píxeles) de la imagen (ambos menores o iguales que 50). A continuación viene la imagen en blanco y negro en donde el carácter '`.`' representa el color blanco y '`X`' el negro.

Se puede asumir que:

- El fondo de la imagen es siempre blanco.
- Todas las ovejas tienen la silueta negra. Las ovejas blancas tienen algo blanco dentro de su silueta.
- Las ovejas nunca se solapan (es decir, en las fotos las ovejas nunca se tocan).
- Ninguna oveja entra en contacto con los bordes de la foto (es decir, en todas las fotos la primera y última fila y la primera y última columna son '`.`').
- En la foto sólo aparecen ovejas.

Salida

Para cada caso de prueba se escribirá `SI` si en la foto hay alguna oveja blanca y `NO` en caso contrario.

Entrada de ejemplo

```
22 7
......................
.XXX..................
.XXXXXXXX.............
...XXXXXXX........XX..
....XXXXX....XXXXXX...
....X...X..XXX.....XX.
......................
29 7
.............................
.XXX..............XXX....X...
.XXXXXXXX....XXXXXXXX...XXX..
...XXXXXXX..X....XX....X...X.
....XXXXX....XXXXX......XXX..
....X...X....X...X.......X...
.............................
```

Salida de ejemplo

```
NO
SI
```

C
Claras y oscuras

Después de muchos años ahorrando para la obra de su ruinosa casa, Paca Herse ha podido por fin contratar a una cuadrilla de obreros que están prácticamente haciéndole una casa nueva. Los gastos se han disparado y cualquier cosa es importante para ahorrar unos eurillos.

Para los suelos, quiere poner unas losetas grandes con dos colores alternos para que formen un patrón clásico similar al de los tableros de ajedrez. Para no desaprovechar material, que luego no se puede devolver, ha estado midiendo todas las estancias de la casa, para averiguar el número de losetas de ancho y de alto que necesita en cada una. Ahora quiere afinar más para saber exactamente cuántas necesita de cada color.

Entrada

La entrada comienza con un número indicando la cantidad de casos de prueba que el programa deberá procesar.

Un caso de prueba consiste en dos números naturales mayores que 0 y menores que 1.000 con el número de losetas que entran en una habitación a lo ancho y a lo alto.

Salida

Para cada caso de prueba, el programa escribirá una línea con el número de losetas que hay que comprar de cada tipo. Los colores aún no están decididos, pero para que la casa sea lo más luminosa posible, prefiere que, si hay que elegir, haya más losetas claras que oscuras. En la salida se indicará primero cuántas losetas claras se necesitan, seguido del número de oscuras.

Entrada de ejemplo

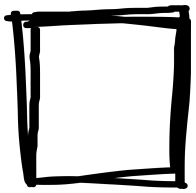
```
3
1 3
4 4
2 6
```

Salida de ejemplo

```
2 1
8 8
6 6
```

D

Conseguir un cuadrado perfecto



Un número es un *cuadrado perfecto* si su raíz cuadrada es un número exacto (sin decimales). Por ejemplo, el 4 es un cuadrado perfecto (2^2), al igual que lo son el 36 (6^2) y el 3.500.641 (1871^2).

Todos los números que *no* son cuadrados perfectos pueden multiplicarse por otros para conseguir serlo. Por ejemplo, el número 8 no es un cuadrado perfecto, pero al multiplicarlo por 2 se obtiene el 16, que sí lo es.

Entrada

La entrada comienza con un número que indica cuántos casos de prueba tendrán que procesarse.

Cada caso de prueba consiste en un número mayor que 0 y menor que 2^{31}.

Salida

Para cada caso de prueba, el programa escribirá por la salida estándar, en una línea independiente, el número más pequeño que al ser multiplicado por el número del caso de prueba da como resultado un cuadrado perfecto.

Entrada de ejemplo

```
3
4
8
12
```

Salida de ejemplo

```
1
2
3
```

E
Resta y amus

Intentando impresionar a su profesor, Alonso y Aldonza se han inventado un extraño juego con números. Se comienza con un valor positivo cualquiera seleccionado aleatoriamente. A partir de entonces, en cada turno los contrincantes deciden qué número jugar, y lo utilizan para modificar ese valor. La modificación se realiza restando el número elegido, y sumando ese mismo número tras invertir sus dígitos. Si resultara que el número elegido fuera *menor* que el opuesto, el proceso sería contrario, restándose el opuesto y sumándose el original, para garantizar que el valor final nunca crezca. Esto es importante, porque gana el jugador que tras modificar el número llega a cero; además, no se permite realizar una jugada que haga que el valor final quede por debajo. Se han tomado la libertad incluso de darle un nombre: "resta y amus" (suma, al revés).

Por ejemplo, en una partida han empezado con el número 36. Aldonza ha jugado el número 21, por lo que ha restado 21 y ha sumado 12, llegando al 27. Alonso, mucho más bruto, ha decidido jugar el 68. El opuesto es el 86, que es mayor, por lo que ha restado 86 y ha sumado 68, dando 9. Finalmente, Aldonza ha jugado el 10, alcanzando el buscado 0 que le ha dado la victoria.

Por desgracia, tras jugar unas cuantas veces se han dado cuenta de que en la mayor parte de las partidas no son capaces de terminar en 0, y quedan en tablas. Desconcertados, han ido a pedir ayuda a Miguel, su profesor, para que les cuente cómo conseguir llegar a 0 o, al menos, con qué números empezar para garantizar que no terminarán en tablas.

Miguel, hombre de rostro aguileño, barbas de plata y más de pluma que de números, no les ha hecho mucho caso y los pobres críos andan por los caminos buscando algún bachiller que les ayude.

Entrada

La entrada al programa estará compuesta por múltiples casos de prueba. Cada uno será un número candidato a constituir el inicio de una partida del juego *Resta y amus*, mayor que 0 y menor que 2^{31}. Los valores de los números que los niños pueden jugar en cada turno no tienen límite, aunque no podrán ser negativos.

La entrada terminará con un `0`, que no deberá procesarse.

Salida

Para cada caso de prueba, el programa escribirá `SI` si el número es válido para una partida del juego, de modo que se garantice que hay al menos un modo de llegar a 0 y no terminar en tablas. En otro caso se escribirá `NO` (sin las comillas).

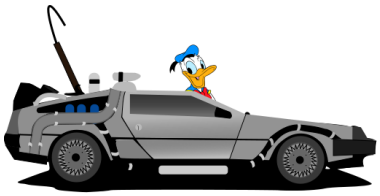
Entrada de ejemplo

```
36
1
0
```

Salida de ejemplo

```
SI
NO
```

F
Michael J. Fox y el Pato Donald



Si cogemos a los actores protagonistas de una película como la emblemática *Regreso al Futuro* y analizamos sus fechas de cumpleaños, vemos que ninguna de ellas coincide. Es decir, aunque durante el rodaje celebraron algún que otro cumpleaños, no hubo ningún día en el que soplaran las velas dos actores.

La cosa cambia si a ese elenco añadimos, por ejemplo, los personajes clásicos de Disney. Resulta que tanto Michael J. Fox (que interpretaba a Marty McFly) como el pato Donald celebran su cumpleaños el día 9 de junio[1].

En realidad, no es tan raro que cuando se junta un grupo de gente haya dos personas que cumplan años el mismo día. Es lo que se conoce como la *paradoja del cumpleaños*. Con un sencillo cálculo se puede comprobar que si juntamos a 23 personas elegidas al azar, la probabilidad de que haya dos que coincidan es de más del 50%. Si subimos a 57 personas, la probabilidad se dispara hasta el 99%.

Entrada

La entrada está formada por distintos casos de prueba, cada uno en dos líneas.

La primera línea de cada caso de prueba contiene el número de personas en el grupo. La segunda contiene las fechas de nacimiento de cada uno de ellos, con el formato `día/mes/año`. Todas las fechas serán válidas y estarán comprendidas entre el año 1600 y el 2000.

La entrada termina con una línea con un `0` que no debe procesarse.

Salida

Por cada caso de prueba se escribirá `SI` si hay algún cumpleaños repetido y `NO` en caso contrario.

Entrada de ejemplo

```
4
9/6/1961 22/10/1938 31/5/1961 20/4/1964
5
9/6/1961 22/10/1938 31/5/1961 20/4/1964 9/6/1934
0
```

Salida de ejemplo

```
NO
SI
```

[1] Siempre y cuando consideremos como nacimiento de Donald la fecha de estreno del primer corto en el que apareció en 1934 titulado "La gallinita sabia" (en inglés "*The Wise Little Hen*").

G
Nana al bebé de papá y mamá

A los padres primerizos se les cae la baba cuando su bebé dice sus primeras palabras: "*mamá*" y "*papá*".

Pero ¿por qué son esas dos palabras las primeras que aprendemos a decir? Es curioso que ambas sean la repetición de una sílaba dos veces. ¡Eso las hace más fáciles de vocalizar! De hecho, el vocabulario de la primera infancia está lleno de palabras así: mamá, papá, nana, bebé, tete, coco, baba, pipí...

A todas esas palabras las denominaremos *palabras cíclicas*, pues se construyen con la repetición, varias veces, de un conjunto de letras. Cuando alguien duerme escribimos "*Zzzzz*", que también es cíclica, pues es la repetición de la letra "z". También lo son "*toctoc*" o "*trantrán*".

Entrada

El programa debe aceptar una serie de palabras, cada una en una línea, con longitudes de entre 1 y 200 letras. Contendrán únicamente letras del alfabeto inglés, en mayúscula o minúscula, sin espacios ni ningún otro separador.

Salida

Para cada cadena, el programa escribirá la longitud de la subcadena más corta posible que puede usarse para crear la palabra original repitiéndola varias veces, ignorando las diferencias entre mayúsculas y minúsculas.

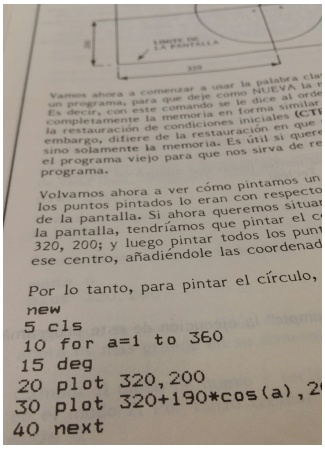
Entrada de ejemplo

```
Mama
TranTran
ZzZzZzZzZzZz
juguete
```

Salida de ejemplo

```
2
4
1
7
```

H
RENUM


Vamos ahora a comenzar a usar la palabra cla
un programa, para que deje como NUEVA la m
Es decir, con este comando se le dice al orde
completamente la memoria en forma simular
la restauración de condiciones iniciales (CTR
embargo, difiere de la restauración en que
sino solamente la memoria. Es útil si quere
el programa viejo para que nos sirva de re
programa.

Volvamos ahora a ver cómo pintamos un
los puntos pintados lo eran con respecto
de la pantalla. Si ahora queremos situar
la pantalla, tendríamos que pintar el c
320, 200; y luego pintar todos los punt
ese centro, añadiéndole las coordenad

Por lo tanto, para pintar el círculo,

```
new
5 cls
10 for a=1 to 360
15 deg
20 plot 320,200
30 plot 320+190*cos(a),2
40 next
```

En los albores de la informática, cuando los entornos de desarrollo estaban por inventar, los lenguajes de alto nivel eran una rareza, la programación funcional estaba naciendo y la programación orientada a objetos por nacer, apareció el lenguaje que marcaría la infancia a toda una generación de informáticos: el BASIC.

Una de las peculiaridades del lenguaje original era la existencia de *números de línea*. Debido a la ausencia de editores de texto para construir el código fuente, cada línea que se deseaba añadir debía ir antecedida del número de línea que ocuparía. Esa numeración (que podía tener huecos entre una línea y la siguiente) se utilizaba también a modo de etiqueta en las instrucciones de salto.

Un ejemplo de programa en BASIC es el siguiente:

```
 5 REM Euclid's algorithm for greatest common divisor
 6 PRINT "Type two integers greater than 0"
10 INPUT A,B
20 IF B=0 THEN GOTO 80
30 IF SGN(A-B) = 1 THEN GOTO 60
40 LET B=B-A
50 GOTO 20
60 LET A=A-B
70 GOTO 20
80 PRINT A
90 END
```

Como se ve, cada línea de código comienza con el número que la identifica, de forma que otras instrucciones de salto pueden referenciarlas (`GOTO xxx` o `GOSUB xxx`).

Durante la edición era habitual que la numeración fuera múltiplos de 10, para, en caso de necesidad, poder añadir líneas de código adicionales cómodamente. Además, llegado el momento, siempre se podía utilizar la orden `RENUM` para *renumerar* las líneas de nuevo automáticamente.

Entrada

La entrada comienza con un entero indicando el número de casos de prueba que vendrán a continuación.

Cada caso de prueba consiste en un programa de como mucho 100 líneas en un BASIC muy simplificado en donde cada línea de código:

- Comienza con el número de línea al que representa. La numeración es ascendente con números positivos entre 1 y 10^9.
- Tras el número de línea aparece un espacio en blanco.
- A continuación aparece una única instrucción, que podrá ser únicamente `GOTO`, `GOSUB` o `RETURN`. Las dos primeras van seguidas del número de línea al que saltar (con un único espacio de separación) y se garantiza que éste siempre existe.

El listado del programa termina siempre con una línea con numeración 0, que no se considera parte de él.

Salida

Por cada caso de prueba se escribirá el programa en BASIC equivalente tras la renumeración de las líneas.

Esa renumeración asignará el número `10` a la primera línea de código e irá incrementando éste de 10 en 10.

Separa cada listado con una línea con tres guiones (`---`).

Entrada de ejemplo

```
2
2 GOTO 5
5 GOSUB 50
50 RETURN
0
10 GOSUB 50
11 GOTO 10
50 RETURN
0
```

Salida de ejemplo

```
10 GOTO 20
20 GOSUB 30
30 RETURN
---
10 GOSUB 30
20 GOTO 10
30 RETURN
---
```

I
Telesillas

A pesar de tener que abandonar su país por culpa de la guerra durante la adolescencia, a Wan As Kai la vida le ha tratado bien. Se ha convertido en un hombre muy rico, que hoy dedica parte de su tiempo a buscar formas de devolver a la gente que le acogió todo lo que le han dado.

Los recuerdos más felices de su infancia, allá lejos en las montañas, están atados a las blancas pendientes y al suave contacto de unos esquíes sobre la superficie nevada. Aunque ahora habita en un país cálido, quiere que todos los niños, incluídos sus propios hijos, puedan vivir esas mismas experiencias, por lo que ha financiado *Es Nou Parc*, un parque de nieve con dos pequeñas pistas, un telearrastre e incluso un telesilla, para que pequeños, y no tan pequeños, puedan comenzar a amar el deporte de su vida.

Por desgracia, la falta de experiencia de la constructora ha hecho que el telesilla instalado tenga una inquietante limitación de peso. Aunque cada silla tiene espacio para dos personas, sus usuarios no pueden superar un peso máximo, o se corre el riesgo de que la barra que une la silla con el cable se parta.

Este problema resulta crítico para el buen funcionamiento del parque. El número de usuarios es enorme (naturalmente, ¡todo el mundo quiere esquiar!) y el tiempo de espera en el acceso al telesilla es un cuello de botella para el disfrute de la nieve. Para solucionarlo, hay que mejorar el uso de los recursos, minimizando el número de sillas (dobles) que deben usarse para que todos los usuarios que quieren usar el telesilla suban hasta la cima. Como ayuda, conocemos el peso de cada uno de ellos y están dispuestos a reordenarse siempre que se consiga reducir el tiempo de espera total.

Entrada

La entrada del programa contendrá múltiples casos de prueba. Cada uno comienza con una línea con dos números mayores que cero. El primero indica el máximo peso que soporta cada silla (de dos plazas), y el segundo la cantidad de usuarios que desean utilizarlo (como mucho 10.000).

A continuación, vendrá una línea con los pesos de todos esos usuarios, separados por un espacio. Se garantiza que el peso soportado por cada silla será como mínimo el mayor de tales pesos, y no será nunca mayor que 10^9.

La entrada termina con una línea con dos ceros.

Salida

Para cada caso de prueba, el programa escribirá el número mínimo de viajes que se necesitan para que todos los usuarios suban hasta la cima.

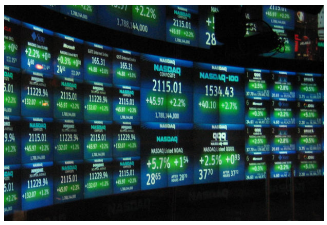
Entrada de ejemplo

```
20 4
10 10 10 10
30 5
12 20 10 16 8
40 6
30 30 30 30 30 30
0 0
```

Salida de ejemplo

```
2
3
6
```

J
El secreto de la bolsa



Jonás es un paranoico. Cuando ve los ríos de números con la cotización de la bolsa, cree que es el modo que tienen los gobiernos de enviar información oculta a sus agentes secretos.

Dice que el primer número que se emite en el momento de la apertura es el que hace las veces de *suma clave*. Una vez identificado, todos los demás números del día se ponen seguidos, formando una gran secuencia. Lo importante de la secuencia es cuántas veces se puede encontrar la suma clave en la suma de dígitos *consecutivos*.

Dependiendo de cuántas veces esté escondida la suma, los agentes infiltrados sabrán si tienen que mantenerse escondidos o entrar en acción de alguna forma.

Para poder demostrar su teoría, Jonás necesita poder contar rápidamente cuántas veces aparece la suma en las cotizaciones del día. Cuando lo hace a mano, le lleva cosa de una semana averiguarlo, y para cuando lo sabe la información está demasiado obsoleta como para ser útil.

Entrada

La entrada comienza con un primer número indicando cuántos casos de prueba tendrán que procesarse.

Cada caso está compuesto por una única línea, que comienza con un número con la *suma clave*. A continuación, tras un espacio, vendrá la secuencia de dígitos. La cotización del día puede ser muy larga (para poder esconder mucha información), por lo que la secuencia puede llegar a tener hasta 250.000 dígitos.

Como lo importante de este supuesto sistema de codificación son las sumas, los ceros no aportan nada y son eliminados con antelación. Gracias a esto, la *suma clave* no será nunca 0, ni aparecerá dicho dígito en la secuencia.

Salida

Por cada caso de prueba el programa escribirá el número escondido en la cotización, es decir, cuántos bloques de dígitos consecutivos de la secuencia suman exactamente la suma clave.

Entrada de ejemplo

```
3
20 1234
10 91234
3 1214212
```

Salida de ejemplo

```
0
2
4
```