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

Programa-Me 2018
Regional de Madrid y Olot
Problemas

Ejercicios realizados por

Universidad Complutense
de Madrid

Realizado en el **CFTIC de Getafe (Madrid)**
y en el **Institut Bosc de la Coma (Olot, Girona)**
21 de marzo de 2018

21 de marzo de 2018
http://www.programa-me.com

Listado de problemas

Autores de los problemas:

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

El problema **A** está basado en una idea de Joan Rodríguez. La foto que acompaña al enunciado también es suya.

Revisores:

- Ferran Borrell Micola (Institut Baix Camp – Reus)
- Sergi Coll (Institut Bosc de la Coma – Olot)
- Cristina Gómez Alonso (Institut Sa Palomera – Blanes)
- Marc Nicolau (Institut Bosc de la Coma – Olot)
- Joan Rodríguez (Institut Bosc de la Coma – Olot)

A
Entrenamiento imperial

Para su entrenamiento, los integrantes de las tropas de asalto tienen que realizar grandes marchas a pie, para acostumbrarse al peso de su armadura metálica, practicar la autodisciplina y no tener miedo al sufrimiento.

Los entrenamientos son muy duros y no es extraño que duren varias semanas. Se especifican como una sucesión de tramos, cada uno dando su longitud y su desnivel, en metros.

No todos los soldados tienen las mismas capacidades. Los soldados exploradores (*scout troppers*) están especializados en todo tipo de terrenos, pero los soldados costeros (*shoretroppers*) llevan muy mal los desniveles, aunque aguantan largas distancias. En particular, la capacidad de un soldado se mide en función de esos dos parámetros: distancia y desnivel (de subida) máximos que puede cubrir en una jornada.

Entrada

Cada caso de prueba comienza con dos números, D y S indicando la máxima distancia y el máximo desnivel de subida que puede cubrir en una sola jornada un determinado soldado imperial.

A continuación se describen las características de un entrenamiento, como una sucesión de tramos, para cada uno indicando su longitud y desnivel. La descripción del entrenamiento termina con dos ceros. Ninguna distancia será mayor que 100.000, ni ningún desnivel será mayor que 1.000 en valor absoluto.

Por su parte la entrada termina, también, con dos ceros.

Salida

Por cada caso de prueba el programa escribirá el mínimo número de jornadas que necesita el soldado imperial para terminar el entrenamiento sabiendo que cada tramo debe completarse en una única jornada (no pueden partirse), y deben recorrerse en el orden en el que aparecen en la entrada. Si es imposible que finalice el entrenamiento, se escribirá "`DESTITUIDO`".

Entrada de ejemplo

```
2000 30
1000 0 1000 30 0 0
1900 40
1000 10 1000 20 0 0
5000 10
2000 7 2000 -10 0 8 0 0
1000 100
1100 0 0 0
0 0
```

Salida de ejemplo

```
1
2
2
DESTITUIDO
```

B
Participantes en una estadística

Es bien conocido que el 80% de las estadísticas son falsas[1], y pueden malinterpretarse. Por ejemplo, aproximadamente un tercio de los accidentes de tráfico con víctimas mortales se producen por infracciones de velocidad. Como los otros dos tercios se producen a velocidades permitidas, ¡es más seguro correr!

El error está, entre otros lugares, en *la muestra*. Hay muchos más desplazamientos *sin* infracciones de velocidad, y por tanto es normal que haya más accidentes en ese conjunto. La pregunta interesante es ¿qué *porcentaje* de las infracciones de velocidad acaban en accidentes? Seguramente será mayor que entre los desplazamientos sin infracciones.

El problema de la falta de información sobre el tamaño de las muestras ocurre debido al uso de porcentajes. Aunque resultan de gran utilidad a la hora de resumir los datos, es precisamente ese resumen el que puede ocasionar problemas de interpretación. Si nos dicen "El 100% de los suecos son morenos." seguramente diremos que es falso; pero si quien ha hecho esa estadística ha utilizado a *un único sueco* que, por casualidad, ha resultado ser moreno, la estadística es fiel a los datos.

Cuando nos dan un porcentaje asociado a una estadística, deben darnos también del tamaño de la muestra. Si no lo hacen, al menos podremos averiguar *el mínimo* número. Por ejemplo, si nos dicen "El 80% de las mujeres llevan el pelo largo." sabremos que, al menos, han utilizado una muestra de 5 mujeres, de las cuales 4 llevaban el pelo largo. Con una cantidad inicial menor es imposible llegar a ese porcentaje.

Entrada

El programa recibirá, por la entrada estándar, un primer número indicando cuántos casos de prueba vendrán a continuación.

Cada caso de prueba consistirá en una línea con el dato de una estadística *en tantos por diez mil*, es decir el porcentaje multiplicado por 100. Así, para el 80% del ejemplo se recibirá `8000`.

Todos los casos de prueba serán números naturales entre 1 y 10.000.

Salida

El programa escribirá, para cada caso de prueba, el *mínimo* tamaño de la muestra que se habrá utilizado para calcular la estadística.

Entrada de ejemplo

```
4
8000
2500
1250
7
```

Salida de ejemplo

```
5
4
8
10000
```

[1] Incluída ésta.

C
Tarta Sacher

Para celebrar tu próximo cumpleaños has decidido sacar tu vena culinaria y, en lugar de comprar otro año más la misma tarta, hacerla en casa.

Como amante del chocolate que eres, has decidido hacer una tarta Sacher. La historia de esa tarta es digna de película, e incluye incluso disputas legales entre los herederos de Franz Sacher y la pastelería vienesa *Demel* por hacerse con el derecho a comercializar la tarta con el apelativo "*Tarta Sacher original*".

Después de mucho buscar, has conseguido encontrar la que parece ser la primera receta, la de verdad, la creada en 1832 y no los burdos intentos por imitarla. Las cantidades de los ingredientes vienen indicados en medidas bastante imaginativas, porque en aquella época los kilos como medida se usaban aún solo en Francia.

En particular, la cantidad de chocolate que hay que usar viene medida en *cuadritos*. Confías en que un *cuadrito* de 1832 siga siendo lo mismo que un *cuadrito* en las tabletas de chocolate de hoy en día, así es que ahora lo que necesitas es averiguar cuántas tabletas tienes que comprar, sabiendo cuántos entran en cada una.

Entrada

El programa deberá leer, de la entrada estándar, un primer número indicando cuántos casos de prueba vendrán a continuación.

Cada caso de prueba consiste en tres números separados por espacios, $1 \leq n, m \leq 1.000$, $1 \leq r \leq 10.000$. Los dos primeros indican el número de *cuadritos* que vienen en las tabletas del chocolate que vas a comprar, a lo ancho y a lo alto. El número r indica cuántos *cuadritos* necesitas según la receta original.

Salida

Por cada caso de prueba el programa escribirá la mínima cantidad de tabletas de chocolate que tienes que comprar.

Entrada de ejemplo

```
2
4 6 20
7 5 50
```

Salida de ejemplo

```
1
2
```

D
Lío con los casos de prueba

Tras estar mucho tiempo enganchado a los concursos de programación, has decidido que ha llegado el momento de pasar *al otro lado* y vas a hacer tus propios problemas con los que retar a tus compañeros.

Has empezado con uno fácil:

> “Cada caso de prueba comienza con un primer número $n \geq 1$ indicando la cantidad de números que vendrán a continuación. Tras él, vendrán n números mayores que 0. El programa deberá escribir, en una línea independiente, la suma de todos ellos.”
>
> **Entrada de ejemplo**
>
> ```
> 1 1
> 2 1 4
> ```
>
> **Salida de ejemplo**
>
> ```
> 1
> 5
> ```

Has hecho tus casos de prueba, tus soluciones en un montón de lenguajes, has medido tiempos de ejecución... ¡todo perfecto!

Pero, por desgracia, has sufrido un percance en tus datos y media hora antes de utilizar el problema en tu primer concurso de programación como organizador, te has dado cuenta de que el fichero con los casos de prueba se ha dañado y ha perdido el primer número de la entrada, es decir la longitud del primer caso de prueba. Y no solo eso; además todos los números se te han amontonado en la misma línea. Por ejemplo, para la entrada mostrada del enunciado el fichero se te ha quedado así:

```
1 2 1 4
```

Para que el concurso pueda realizarse, tienes que resolver el desaguisado rápidamente, y averiguar, dada la sucesión de números de cada fichero estropeado, el número ausente.

Entrada

Cada caso de prueba comienza con un primer número indicando la cantidad de números que hay en uno de tus ficheros dañados de casos (como mucho 250.000). A continuación aparecen esos números, separados por espacio. Ningún número es mayor que 10^9.

La entrada termina con un `0`.

Salida

Por cada caso de prueba, el programa escribirá el número del fichero que se ha perdido y debería ser la longitud del primer caso de prueba. Si hay varias opciones posibles, se dará la más corta.

Entrada de ejemplo

```
4
1 2 1 4
1
9
5
4 5 6 7 8
0
```

Salida de ejemplo

```
1
1
5
```

E
En la mente del timonel

La regata Cambridge–Oxford es una famosa competición de remo entre esas dos universidades, que se celebra todas las primaveras en el río Támesis. La primera edición tuvo lugar en el lejano 1829, y desde 1856 se ha celebrado todos los años con la excepción de los periodos de las Guerras Mundiales.

Aunque los remeros tienen que ser estudiantes con una buena complexión física, para el puesto de timonel se busca precisamente lo contrario. Cuanto más delgado, escuálido y poca cosa sea el que lleve el timón mucho mejor, porque no genera fuerza de empuje y lo único que aporta es peso.

Precisamente por eso le eligieron para formar parte del equipo de remo de Oxford. Lo que nadie entendió es por qué, pese a su inteligencia, dejó que una cuerda se enrollase en el timón nada más sonar el pistoletazo de salida, arruinando los meses de entrenamiento de sus compañeros.

Aun así le dieron otra oportunidad... que desaprovechó, llevando a su embarcación a una colisión frontal contra la de otro equipo.

Y es que Stephen Hawking solía tener la cabeza en otro sitio.

En particular, tenía la tonta costumbre de memorizar todos los números que veía desde que se levantaba por la mañana hasta que se ponía a los mandos del bote. Y en ese momento, en lugar de fijarse por dónde iba, empezaba a repasarlos todos mentalmente para calcular, de cabeza, el mayor valor que se pudiera conseguir con la multiplicación de dos de ellos.

Esta costumbre le hizo convertirse en uno de los peores timoneles que recuerda Oxford. Pero a cambio le permitió entrenar la memoria visual, algo que le vino muy bien cuando la enfermedad consumió sus músculos y tuvo que construir todas sus teorías físicas mentalmente.

Entrada

Cada caso de prueba está compuesto por dos líneas. En la primera aparecerá un valor $2 \leq n \leq 200.000$ indicando la cantidad de números que aparecerán en la segunda línea. Todos ellos serán valores enteros no mayores que 10^9 en valor absoluto. La entrada termina con un `0`.

Salida

Por cada caso de prueba, el programa escribirá el número más alto que se puede conseguir con la multiplicación de exactamente dos números de la entrada.

Entrada de ejemplo

```
4
10 40 20 20
4
-10 -40 -20 -20
5
-5 -1 0 1 2
2
-1 1
0
```

Salida de ejemplo

```
800
800
5
-1
```

F
Presupuestando semáforos

El tráfico en el pueblo se ha vuelto inmanejable. Hace unos años era raro ver dos tractores coincidir en una esquina. Ahora la salida del sol hace despertar a infinidad de coches, tractores, furgonetas, motos y otros medios de locomoción que llenan las calles de ruido y confusión. Si a eso le sumamos los peatones que zigzaguean por la falta de aceras y le añadimos aquellos que siguen moviéndose en burro, la situación es insostenible.

Como alcalde, Jacinto ha decidido coger el toro por los cuernos y, pese a las protestas de muchos vecinos, llenar el pueblo de semáforos. Pondrá, dice, un semáforo en cada calle que llegue a una intersección, trayendo la modernidad a este pueblo que hace bien poco no tenía ni las calles asfaltadas.

Eso sí, como el número de intersecciones en el pueblo es grande no podrá hacerlo de manera inmediata. El primer paso es incorporar la previsión del gasto en los presupuestos del año que viene. Y para eso necesita saber cuántos semáforos habrá que instalar, por lo que es hora de ponerse a contar.

Entrada

La entrada contiene distintos casos de prueba, cada uno con un plano en forma de cuadrícula.

La descripción de cada plano comienza con una línea con dos números con el ancho ($1 \leq a \leq 1.000$) y alto ($1 \leq l \leq 1.000$) del pueblo.

A continuación vienen l líneas cada una con a caracteres (distintos de espacios). Cada carácter tiene un significado (los hay para simbolizar parques, casas, consultorio médico...), pero el que nos interesa es el `#` que representa una calle (cuyo ancho es siempre 1). Las calles van en horizontal o vertical.

Salida

Por cada caso de prueba se escribirá una línea con un único número conteniendo el número de semáforos totales que hay que presupuestar.

Entrada de ejemplo

```
5 4
--#..
--#..
####'
:::##
20 3
..#......#####...###
#######..#...#...#..
..#...####...#####..
7 4
..#....
#######
...#.#.
...#.#.
```

Salida de ejemplo

```
3
4
9
```

G

Base raíz de 10

Los sistemas de numeración posicionales son aquellos que asignan a cada dígito un valor distinto en base a su posición. La llamada base 10, que es la que utilizamos normalmente, utiliza 10 dígitos distintos que son multiplicados por una potencia de 10 dependiendo del lugar que ocupan en el número completo.

$$4.321 = 4 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$$

Dependiendo del número elegido como base se necesitarán más o menos dígitos para expresar el mismo número. Además la elección de ciertos números como base da resultados curiosos. Uno de ellos es $\sqrt[2]{10}$ utilizando los dígitos habituales $(0 \ldots 9)$.

Entrada

La entrada estará compuesta por distintos casos de prueba, cada uno de ellos en una línea.
En cada línea aparecerá un número n $(0 \leq n \leq 10^{100})$ que hay que convertir a base $\sqrt[2]{10}$.

Salida

Para cada caso de prueba se mostrará en una línea independiente el número n en su representación en base $\sqrt[2]{10}$.

Entrada de ejemplo

```
1
12
123
```

Salida de ejemplo

```
1
102
10203
```

H
Conquistando bases imperiales

Para su próxima ofensiva, el ejército rebelde ha decidido conquistar varias de las bases enemigas cercanas al planeta Polis Massa. Para eso ha estimado el mínimo número de soldados que necesitará para tener unas mínimas garantías de conquistar cada base enemiga, así como el número de bajas esperadas. Además, se espera que las fuerzas imperiales no se queden con los brazos cruzados ante cada conquista, por lo que es necesario dejar un retén militar permanente en las bases conquistadas, para impedir que sean recuperadas por el enemigo.

El número de efectivos para la ofensiva es limitado, por lo que se ha decidido atacar las bases imperiales de una en una. El orden no está fijado, y se quiere elegir aquél que minimice el número total de soldados necesarios para conquistarlas todas. Tu misión, si decides aceptarla, es determinar ese número.

Entrada

Cada caso de prueba comienza con el número $1 \leq N \leq 1.000$ de bases enemigas. A continuación vendrán N líneas, una por cada base.

Para cada una se indican tres valores, la cantidad mínima de soldados necesarios para conquistar la base, s, las bajas esperadas, b, y el número de soldados que habrá que dejar como retén tras la batalla, r. Se garantiza que $0 \leq b \leq s \leq 1.000$ y $0 \leq r \leq 1.000$.

Un caso sin bases, que no deberá procesarse, marca el final.

Salida

Para cada caso de prueba el programa escribirá la cantidad mínima de soldados necesaria para conquistar todas las bases.

Entrada de ejemplo

```
2
100 0 10
90 30 20
3
50 40 20
20 20 30
50 10 5
0
```

Salida de ejemplo

```
100
125
```

Notas

Para conseguir conquistar las tres bases del segundo ejemplo con sólo 125 soldados hay que empezar conquistando primero la última (`50 10 5`). El orden en el que se ataquen después las otras dos es indiferente.

I
Mundo limpio

Cuando una persona prevé que llega su final, se plantea muchas cosas. Si su vida ha tenido sentido, si habrá hecho feliz a la gente que le rodea, si podría haber tomado otros caminos distintos, qué cosas cambiaría si pudiera...

En su lecho de muerte, mi abuelo me sorprendió con unas declaraciones fuera de lo normal:

> Hijo[2], yo, igual que tú, llevo gafas desde los dos años. Estimo que eso significa que he visto el mundo limpio en mi vida unos míseros 27 días en total. Ahora que ha llegado mi hora me he dado cuenta de que debido a toda esa suciedad acumulada en mis cristales he dejado de ver muchas cosas preciosas, colores vivos, detalles... Por favor, no hagas lo que yo, y límpiate las gafas más a menudo.

Para darle un sentido épico a la historia, me gustaría poder decir que se lo prometí allí mismo, que nos fundimos en un abrazo final y que en ese momento nos dejó. La realidad es bien distinta. Mi cabeza se puso a intentar estimar cuántos días habría visto yo el mundo limpio y me perdí sus últimas palabras.

Entrada

La entrada comienza con el número de casos de prueba que deben procesarse.

A continuación aparece cada caso de prueba en una línea independiente. Cada línea comienza con un número que indica el número total de veces que una persona se ha limpiado las gafas en su vida (nunca más de 100.000 veces, que sumarían más de 270 años a razón de una limpieza diaria). A continuación, en formato `HH:MM:SS`, se indica el tiempo aproximado que las gafas se mantienen limpias (siempre menos de un día).

Salida

Por cada caso de prueba se escribirá el tiempo total de "mundo limpio" que se estima que ha tenido la persona. El formato será `d HH:MM:SS`, donde `d` representa el número de días completos, y `HH`, `MM` y `SS` las horas, minutos y segundos del último día inconcluso (cada uno debe ocupar dos dígitos).

Entrada de ejemplo

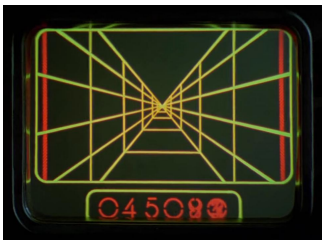
```
2
60 00:00:01
3200 00:12:09
```

Salida de ejemplo

```
0 00:01:00
27 00:00:00
```

[2] Nunca le gustó llamarme *nieto*. Bien pensado, a mí tampoco.

J
Tomas inéditas



Se está preparando una nueva versión, totalmente remasterizada, de "*Star Wars: Episodio IV*", con mejor audio, mejor imagen y nuevas tomas inéditas.

Bueno, en realidad todo el material de la película original de 1977 ha sido ya publicado y republicado, por lo que la única forma de saciar la sed de los *fans* es crear contenido nuevo, manteniendo el estilo original. Para la nueva reedición, se ha decidido añadir alguna toma adicional del interior de las naves durante la gran batalla final de la película.

En particular, se va a enseñar el monitor del panel de apoyo a la navegación que mostraba gráficas temporales sobre alguna característica de la nave como la altura, velocidad, o inclinación. Como el estilo visual de los paneles de control de la película era, por ser finos, "*vintage*", se quieren generar esas gráficas usando sólo caracteres alfanuméricos para imitar el aspecto. En particular se usarán únicamente los caracteres "`/`", "\" y "`_`" (guión bajo) en función de si la gráfica sube, baja o se mantiene constante en un determinado punto.

Como amante de la "saga galáctica" has decidido crear (¡y donar!) el *software* para que las nuevas tomas vean la luz. Te darán los movimientos de la gráfica en cada paso, y la pintarás con los caracteres anteriores, enmarcándola, además, entre "`#`" para simular el monitor CRT.

Entrada

Cada caso de prueba es una cadena de entre 1 y 100 letras, indicando la evolución de la gráfica. Una "`I`" indica que la gráfica se mantiene igual que en el paso previo ("`_`"), una "`S`" indica que sube una línea hacia arriba ("`/`") y una "`B`" indica que baja una posición "\". Se debe considerar que la "altura" inicial en la gráfica es 0 y a partir de ahí se mueve en función de lo indicado. Se garantiza que el valor nunca descenderá de ese mismo 0 y que no habrá letras distintas a las descritas.

Salida

Por cada caso de prueba se dibujará el panel de ayuda utilizando los caracteres alfanumericos descritos. La gráfica se enmarcará en un cuadro formado por el carácter "`#`". El interior del marco tendrá tantas columnas como letras la entrada, y las filas necesarias para incluir exactamente la gráfica.

Entrada de ejemplo

```
IISSBSIIIII
SISSBIB
```

Salida de ejemplo

```
#############
#      _____#
#   /\/     #
#__/        #
#############
#########
#   /\_ #
# _/   \#
#/      #
#########
```