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

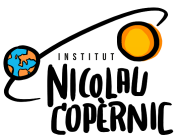
ProgramaMe 2026
Regional de Terrassa
Problemas

Patrocinado por

Ejercicios realizados por

Universidad Complutense
de Madrid

Realizado en el **Institut Nicolau Copèrnic (Terrassa)**



25 de marzo de 2026

25 de marzo de 2026
https://www.programame.com

Listado de problemas

Autores de los problemas:

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

Revisores:

- Iván Cantón (Institut Nicolau Copèrnic – Terrassa)
- Roberto Ferrero (Institut Nicolau Copèrnic – Terrassa)
- Albert Grau (Institut Nicolau Copèrnic – Terrassa)
- Xavi Hernández (Institut Nicolau Copèrnic – Terrassa)
- Manel Orós (Institut Nicolau Copèrnic – Terrassa)
- Eduard Santamaría (Institut Nicolau Copèrnic – Terrassa)

Imágenes y fotografías:

- Problema I, "*Evolución hacker*": creación propia.
- Resto de imágenes (salvo los logotipos): gratis para usos comerciales; no necesitan reconocimiento (licencia Pixabay)

A
Descomprimiendo el ADN

El ácido desoxirribonucleico, mucho más conocido como ADN, es una molécula de gran tamaño que almacena la información para crear otros componentes de la célula, como proteínas. Estructuralmente, son largas cadenas de *bases nitrogenadas*, que pueden ser adenina (A), timina (T), citosina (C) o guanina (G), unidas en parejas.

La longitud puede llegar a ser enorme. En los humanos, el cromosoma número 1 es el más largo, con aproximadamente 220 millones de parejas de bases. Esto hace que en la *secuenciación* del ADN se generen ficheros enormes compuestos únicamente por las letras A, T, C o G, para cada una de las cuatro bases.

Para reducir el tamaño, a veces se utiliza un mecanismo sencillo de compresión. Si una determinada cadena de bases se repite varias veces, se escribe entre corchetes poniendo delante el número de veces que se repite. Por ejemplo, `3[AGC]` representa la cadena `AGCAGCAGC`. Estas cadenas comprimidas pueden, por supuesto, concatenarse (poner varias consecutivas) y anidarse (poner unas dentro de otras). Así, por ejemplo, `C2[T]` es en realidad `CTT` y `2[C2[T]]` se convierte en `CTTCTT`.

Entrada

Cada caso de prueba es una cadena de no más de 10.000 caracteres representando una secuencia de ADN escrita con la notación comprimida descrita. Se garantiza que solo aparecen como letras las de las bases nitrogenadas (`A`, `T`, `C` o `G`) y que el número de veces que se repite cada cadena entre corchetes (el número que va delante) está entre 0 y 9.

Salida

Por cada caso de prueba el programa escribirá la longitud (número de bases nitrogenadas) de la versión "descomprimida" de la cadena. Se garantiza que el valor nunca será mayor de 10^{12}.

Entrada de ejemplo

```
ACG
3[AGC]
2[C2[T]]
0[TA1[TA]GTA]
9[9[9[9[9[9[9[9[9[AGCT]]]]]]]]]
```

Salida de ejemplo

```
3
9
6
0
13947137604
```

B
Tasa de rendimiento

Los responsables educativos suelen dedicar parte de su tiempo a analizar los resultados de los estudiantes, comparando el progreso entre distintas asignaturas, entre distintos cursos e incluso entre estudiantes en primera matrícula y repetidores. En su jerga hay términos como "tasa de rendimiento", "tasa de éxito" o "eficiencia de los egresados" cuyo significado y utilidad a veces es más difícil de entender que las propias asignaturas que analizan.

La *tasa de rendimiento* es el porcentaje de estudiantes matriculados en una asignatura que ha aprobado en una determinada convocatoria. Sabiendo el número de matriculados y la tasa de rendimiento, ¿cuántos han aprobado?

Entrada

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

Cada caso de prueba está compuesto por dos números. El primero, entre 1 y 200, indica el número de estudiantes matriculados en una asignatura. El segundo indica la tasa de rendimiento, un número entre 0 y 100 redondeado a dos dígitos decimales.

Salida

Por cada caso de prueba se escribirá el número de estudiantes que ha superado la asignatura. Los estudiantes ¡no pueden partirse! El número deberá escribirse sin decimales.

Entrada de ejemplo

```
4
100 10.00
200 99.50
57 0.00
3 33.33
```

Salida de ejemplo

```
10
199
0
1
```

C
Números sin dígitos

Cuando se aprende a escribir los números en binario sorprende la longitud, en número de dígitos, que se necesita para representar cantidades relativamente pequeñas. Por ejemplo el número 9, que en decimal solo necesita un dígito, se convierte en 1001, ¡que necesita cuatro!

Ahora bien, la percepción cambia si utilizamos las pequeñas *cuentas* de un ábaco. Un ábaco es un instrumento usado desde la antigüedad para realizar operaciones aritméticas sencillas. Consiste en un soporte rectangular de madera a la que hay ancladas varillas paralelas con pequeñas *cuentas* (bolas típicamente de madera) que pueden desplazarse en ellas. Cada varilla representa una posición dentro del número (unidades, decenas, centenas, ...) y la posición de las *cuentas* de la varilla se utiliza para indicar el dígito concreto en esa posición.

Se cree que antes incluso de que se inventara el ábaco se representaban los números usando esas mismas *cuentas*, que podían ser piedrecillas, granos de maíz o, más macabro, dientes de animales abatidos. Para representar un número se colocaban las *cuentas* todas seguidas, pero era necesario marcar de alguna forma, con un tipo de *cuenta* diferente, el cambio de los dígitos. Por ejemplo, si representamos una *cuenta* normal con una `I` y una *cuenta* de separación como una `,` el número 123 quedaría `I,II,III` y el 201 `II,,I`.

Con esta representación el número de *cuentas* totales que se necesita ¡puede ser menor con bases de numeración más pequeñas! Por ejemplo, el número 9 sería `IIIIIIIII` en base 10 pero solo `I,,,I` en base 2, que, para ese número, resulta mucho más corto. Y, puestos a elegir, mejor en base 9, que sería simplemente `I,`.

Entrada

La primera línea de la entrada contiene un número n con la cantidad de casos de prueba que deberán ser procesados.

A continuación aparecen n líneas, cada una con un caso de prueba que consiste en un número entre 1 y 10^9.

Salida

Por cada caso de prueba se escribirá en qué base (entre 2 y 10) resulta más corto representar el número si se utilizan *cuentas* para indicar cada dígito y los separadores de los dígitos. Si hay varias bases en las que la longitud es mínima se indicará la menor.

Entrada de ejemplo

```
4
2
9
12
40
```

Salida de ejemplo

```
2
9
6
10
```

D
Pagando en el bazar

Curioseando entre los puestos de un bazar de un país de Oriente Próximo, un norteamericano se sintió atraído por unas baratijas que podía comprar para llevar de vuelta como regalo a sus hijas de múltiples exesposas. Después de regatear convenientemente, acordó un precio en libras, la moneda local.

El problema fue que, una vez conseguido el acuerdo, se dio cuenta de que no se podía pagar con tarjeta y solo tenía efectivo en dólares. La negociación tuvo que empezar otra vez para consensuar cuántas libras eran un dólar para hacer el cambio. Una vez aceptada la conversión, el norteamericano le dio unos pocos billetes.

Pero la historia ¡no terminó ahí! La cantidad entregada era mayor que el precio real con la conversión acordada y el tendero le tenía que devolver algo de dinero. El norteamericano se negó a recibir el cambio en libras y lo único que el tendero podía ofrecerle era dárselo en euros, dado que tenía algo de efectivo en esa moneda por el pago de otros turistas. ¡Y vuelta la burra al trigo! Tuvieron que acordar cuántas libras era un euro y luego encontrar una cantidad exacta en dólares para que pudiera haber un cambio exacto en euros de modo que lo pagado fuera el precio exacto en libras.

Entrada

Cada caso de prueba son tres números. El primero indica el precio de la compra en libras (entre 1 y 10^8), el segundo la cantidad de libras que son un dólar, y el tercero la cantidad de libras que son un euro. Todos los números son enteros y ni dólares ni euros valdrán más de 10.000 libras.

La entrada termina con tres ceros, que no deben procesarse.

Salida

Por cada caso de prueba el programa escribirá la mínima cantidad de dólares posible que debe entregar el norteamericano, y la cantidad de euros devueltos por el tendero de modo que el pago realizado sea el precio exacto, en libras, de la compra. Tanto el pago en dólares como el cambio en euros debe ser entero, sin centavos ni céntimos.

Se garantiza que todos los casos de prueba tienen solución y nunca hay que pagar más de 10.000 dólares.

Entrada de ejemplo

```
100 10 10
120 70 5
130 3 50
0 0 0
```

Salida de ejemplo

```
10 0
2 4
60 1
```

E
Suciedad en Manhattan

Aunque los vecinos de Nueva York están orgullosos de que Spiderman, su superhéroe local, sea famoso mundialmente, su costumbre de desplazarse lanzando telarañas a modo de lianas por todas partes está empezando a ser un problema de salud pública.

Antes de perder su popularidad y que sus amigos y vecinos empiecen a cogerle manía, Spiderman está intentando reducir el número de telarañas que dispara, aunque no es una tarea trivial. Cuando está volando colgado de una de sus lianas, puede lanzar otra en cualquier momento. Pero la distancia máxima que podrá avanzar con la nueva telaraña depende de la altura de los edificios cercanos al punto del lanzamiento. Si quiere reducir la suciedad que deja a su paso, tiene que elegir correctamente los puntos en los que lanza sus lianas.

Entrada

Cada caso de prueba comienza con un número $1 \leq n \leq 200.000$ indicando el número de manzanas que tiene la avenida que Spiderman tiene que recorrer. A continuación aparecen n números entre 0 y 10, con el número de manzanas que puede avanzar si lanza su telaraña en cada una. Por ejemplo, un 1 en la primera posición indica que al principio de la calle puede lanzar una liana con la que solo podrá recorrer la manzana actual (pasar a la siguiente), pero no más allá.

La entrada termina con un 0, que no debe procesarse.

Salida

Por cada caso de prueba el programa escribirá el mínimo número de telarañas que debe lanzar Spiderman para llegar desde el principio de la primera manzana hasta el final de la última. Si no se puede llegar al destino se escribirá `IMPOSIBLE`.

Naturalmente, Spiderman nunca va andando, porque le asaltarían los fans... o los encargados de la limpieza.

Entrada de ejemplo

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

Salida de ejemplo

```
3
2
2
IMPOSIBLE
```

F
Strongly Connected Components S.L.

En *Strongly Connected Components S.L.* llevan más de un siglo fabricando eslabones para cadenas. Armando Hierro, el abuelo del abuelo del dueño actual, empezó con un horno para fundir metal y unos pocos moldes, y con ellos forjó su imperio.

Aunque el negocio sigue yendo de maravilla, Bruno Hierro, el gerente actual de la empresa, está decidido a dar el salto y pasar de estar especializados en eslabones a montar cadenas completas. El problema es que siguen usando los moldes de hace 100 años y solo saben hacer eslabones cerrados que, naturalmente, no pueden abrirse. ¡No hay otros más fuertes!

Afortunadamente, en el mismo polígono industrial tiene su sede Union Find S.L., una pequeña empresa familiar especializada en la fabricación de eslabones abiertos. Bruno está convencido de que gracias a ellos y a la contratación de los hermanos Fuertes podrá comenzar con su línea de negocio de cadenas. Pero claro, tiene que buscar la forma de crearlas comprando la menor cantidad posible de eslabones abiertos, para no dar negocio a la competencia y para que los hermanos no tengan que cerrar muchos eslabones.

Entrada

La primera línea de la entrada contiene un número que indica cuántos casos de prueba deberán procesarse. Cada uno, en una línea independiente, es un número entre 1 y 20.000 indicando la longitud, en número de eslabones, de la cadena que se quiere fabricar.

Salida

Por cada caso de prueba el programa escribirá el número de eslabones cerrados y el número de eslabones abiertos que se necesitan para fabricar la cadena, sabiendo que se quiere minimizar este último número.

Entrada de ejemplo

```
2
10
75
```

Salida de ejemplo

```
5 5
38 37
```

G
Reinaldo el Egocéntrico

Reinaldo Naltrám es un egocéntrico. Hace lo que quiere, dice saber de todo, trata fatal a sus subordinados y cree que el mundo gira a su alrededor.

Su egocentrísmo llega hasta tal extremo que ha desarrollado una extraña habilidad: es capaz de ver su nombre aunque las letras estén desordenadas. Si en algún momento lee palabras como *riéndola*, *leridano* (natural de Lérida), *ladierno* (un tipo de árbol pequeño de hoja perenne) o *alindero* ("señalo los límites de un terreno") se identifica con ellas inmediatamente. Sus detractores dicen que también le ocurre con *orinadle* aunque eso nunca lo reconocerá.

Esta capacidad va más allá de las palabras individuales. Si lee la frase "El leridano iba riendo la gracia" ve su nombre ¡tres veces! Concretamente en "leridano", en "riendo la" ¡y también en "... a riendo l..."! En los tres casos aparecen las mismas letras que tiene la palabra "Reinaldo", ignorando orden, espacios y mayúsculas.

Entrada

La entrada comienza con un número que indica la cantidad de casos de prueba que hay que procesar.

Cada caso de prueba está compuesto por dos líneas, cada una con una secuencia de letras del alfabeto inglés en minúscula. Se garantiza que la primera no tendrá más de 500.000 letras y la segunda no tendrá más que la primera ni más de 200.000.

Salida

Por cada caso de prueba se escribirá cuántas veces aparece la segunda cadena como subcadena dentro de la primera, aunque sea con las letras en un orden distinto.

Entrada de ejemplo

```
3
elleridanoibariendolagracia
reinaldo
alinderolaslindesconladierno
reinaldo
amamantarla
naltram
```

Salida de ejemplo

```
3
4
1
```

H

Encendido y apagado automático

Hoy en día, todos los sistemas de calefacción del hogar tienen entre sus elementos un *termostato*, que activa y desactiva la calefacción en función de la temperatura ambiente. Esto proporciona confort a los habitantes manteniendo el gasto bajo control.

Aunque los primeros termostatos eran analógicos y, en ocasiones, con partes móviles, hoy en día la gran mayoría son electrónicos, lo que permite un mayor grado de control. Incorporan por ejemplo un programador horario, soporte inalámbrico para medir la temperatura en el lugar de la casa que se prefiera, o incluso control remoto a través de Internet.

Sea como sea, el funcionamiento básico sigue siendo el mismo. Cuando la temperatura ambiente detectada cae por debajo de un umbral, el termostato notifica al sistema de calefacción que debe ponerse en marcha. Es más dudoso cuando se debe volver a parar. Si el termostato desconecta la calefacción en cuanto se alcanza esa misma temperatura umbral, es muy probable que los ciclos de encendido y apagado sean muy cortos. La temperatura ambiente será seguramente muy estable, pero todo el sistema sufrirá mucho por el encendido y apagado continuos y apenas se aprovechará el calor residual.

Una solución es tener dos umbrales diferentes. Si la temperatura cae por debajo de una temperatura, el termostato pone en marcha la calefacción, y la dejará encendida hasta que la temperatura llegue a un máximo, que se considera aceptable, y en ese momento detendrá el sistema hasta que la temperatura vuelva a caer. De esta forma la temperatura en el hogar fluctua ligeramente, pero el resultado es más eficiente energéticamente.

Entrada

Cada caso de prueba comienza con tres números. El primero indica el número n de minutos en los que se ha estado midiendo la temperatura del hogar (entre 1 y 10.000). Los dos siguientes, e y a indican, respectivamente, las temperaturas de encendido y apagado de la calefacción. Si la temperatura cae por debajo de e, el termostato pondrá en marcha el sistema. Si se alcanza la temperatura a, lo detendrá. En otro caso lo dejará como estuviera.

Tras esos tres primeros valores, el caso de prueba continuará con n números, en otra línea, con la temperatura detectada a lo largo de n minutos.

Se garantiza que todas las temperaturas serán números enteros entre 1 y 40 y que $e < a$.

La entrada termina con tres ceros que no deben procesarse.

Salida

Por cada caso de prueba se indicará cuántos minutos está encendida la calefacción asumiendo que en el minuto anterior a la primera muestra estaba apagada.

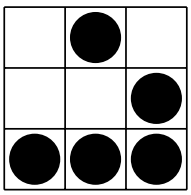
Entrada de ejemplo

```
5 18 20
19 17 19 20 19
4 19 20
19 20 20 19
6 16 25
20 14 20 23 25 25
2 18 19
17 23
0 0 0
```

Salida de ejemplo

```
2
0
3
1
```

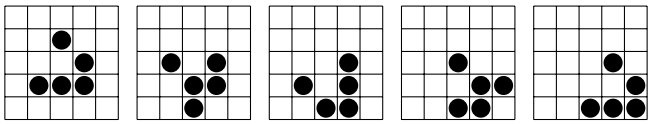
Evolución hacker



Eric S. Raymond propuso como emblema para la subcultura *hacker* una cuadrícula de 3×3 celdas con 5 de ellas ocupadas por círculos negros. Lejos de ser una figura arbitraria y caprichosa, representa un *glider* ("caminador"), un patrón en el *juego de la vida* del matemático británico John H. Conway, un *juego de cero jugadores* creado en la misma época que Internet y que Unix.

El *juego de la vida* es un autómata celular, es decir una rejilla en el que el estado de cada celda evoluciona en función de su estado y el de sus vecinas. En el juego de la vida, las celdas pueden estar en solo dos estados: "vivas" o "muertas". En cada paso de simulación (todas las celdas evolucionan a la vez) una celda *muerta* pasa a estar *viva* si tiene exactamente 3 vecinas *vivas* entre sus 8 adyacentes, y una celda *viva* muere si a su alrededor hay menos de 2 celdas *vivas* (por aislamiento) o más de 3 (por superpoblación).

La evolución de la rejilla está determinada por la de sus celdas individuales, pero surgen "comportamientos emergentes" que han entretenido a matemáticos, informáticos y científicos en general desde su creación. Existen así infinidad de "patrones" conocidos, como el del *glider* que sirve de emblema *hacker*. Si se simula su evolución, el patrón "avanza" en diagonal por la rejilla del autómata, "desplazándose" una celda en cada eje tras 4 pasos de simulación.



Lo llamativo es que este avance ocurre de forma emergente. El *glider* no existe como tal en la definición del juego, sino que surge como resultado de la evolución individual de cada celda de la rejilla.

Entrada

Cada caso de prueba comienza con tres números, f, c y p. Los dos primeros indican, respectivamente, el número de filas y de columnas de una rejilla del juego de la vida ($1 \leq f, c \leq 15$). El tercero (entre 1 y 100) indica el número de pasos de simulación que se quiere hacer evolucionar el autómata.

A continuación aparecen f líneas con c caracteres cada una representando el estado inicial del autómata celular. Un caracter "`.`" indica que la celda en esa posición está *muerta* y un caracter "`O`" (vocal "o" mayúscula) indica que está *viva*.

Se debe considerar que la rejilla actúa de manera circular, es decir que las celdas del borde derecho son adyacentes a las del lado izquierdo y las del borde superior a las del inferior.

Salida

Por cada caso de prueba se escribirá el estado del autómata tras el número de pasos de simulación indicados, utilizando el mismo formato para las celdas *vivas* y *muertas* que en la entrada. Tras cada caso de prueba se escribirá una línea con tres símbolos de igual ("`===`").

Entrada de ejemplo

```
3 7 1
.......
.O.OOO.
.......
5 5 4
.....
..O..
...O.
.OOO.
.....
```

Salida de ejemplo

```
....O..
....O..
....O..
===
.....
.....
...O.
....O
..OOO
===
```

J
Ordenación en el ballet

En la escuela de baile "Mucho Giro" tienen infinidad de grupos, de distintas edades y niveles. Flor Denada y Jessy Metría dan clase a un grupo de niñas de entre 9 y 13 años y están preparando la actuación de final de curso. Tienen casi toda la coreografía montada y en la parte final todas las niñas se quedan en fila mirando hacia el público.

Aunque no queda mal, como las bailarinas tienen diferentes alturas la composición resultante es un poco caótica. Flor quiere que terminen, antes del saludo final, con alturas crecientes, la más bajita en el lado izquierdo y la más alta en el derecho. Pero no pueden cambiar ahora toda la coreografía, de modo que las niñas deben recolocarse partiendo de su posición final actual.

Jessy está a favor del cambio pero solo si se puede hacer integrado en la coreografía y no se rompe la simetría del conjunto. Lo único a lo que está dispuesta es a realizar cambios de posición grupales. No hace falta que todas las niñas se muevan en cada cambio, pero el grupo que se mueva debe ser contiguo y dejar a cada lado la misma cantidad de compañeras sin moverse. Así, con una fila de 9 niñas, o bien se mueven todas, o bien se mueven todas salvo la primera y la última, o bien todas salvo las dos de cada extremo, etcétera.

Además, esos movimientos no pueden ser erráticos. ¡El ballet es orden, precisión y belleza! Los intercambios deben ser simétricos de forma que la primera bailarina del grupo que se mueve intercambiará su posición con la última, la segunda con la penúltima y así sucesivamente. De esa forma, el grupo parecerá girar sobre sí mismo mientras intercambian posiciones, manteniendo la armonía visual del conjunto.

Entrada

Cada caso de prueba comienza con un número n (entre 2 y 200.000) indicando la cantidad de niñas que forma la fila. A continuación aparecen n números con las alturas de las niñas dadas en una unidad que proporciona una gran precisión (números entre 1 y 10^6).

La entrada termina con un 0 que no debe procesarse.

Salida

Por cada caso de prueba el programa escribirá `SI` si es posible recolocar a las bailarinas para que terminen con alturas no decrecientes siguiendo las restricciones de Flor y de Jessy. Si no es posible, se escribirá `NO`.

Entrada de ejemplo

```
3 1 2 3
6 1 6 4 3 1 7
6 1 6 3 4 1 7
3 1 3 2
0
```

Salida de ejemplo

```
SI
SI
SI
NO
```