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

ProgramaMe 2025
Regional de Reus
Problemas

Ejercicios realizados por

Universidad Complutense
de Madrid

Realizado en el **Institut Baix Camp (Reus)**
13 de marzo de 2025

13 de marzo de 2025
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:

- David Marti Batalla (Institut Baix Camp – Reus)
- Eduardo Sorita Calatayud (Institut Baix Camp – Reus)
- Jaime Bocio Sanz (Institut Baix Camp – Reus)

Imágenes y fotografías:

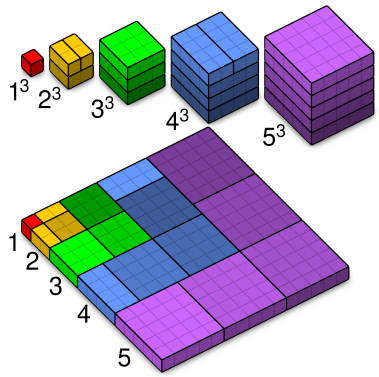
- Problema A, "*Números de Nicómaco*": imagen de la Wikipedia (usuario Cmglee), CC-BY-SA 3.0.
- Problema F, "*Suelo multicolor*" : creación propia.
- Problema G, "*Serpiente de dominó*" : creación propia.
- Problema J, "*Dragon's Lair*" : fotograma de la *película interactiva* "Dragon's Lair" en la que se ambienta el enunciado.
- Resto de imágenes (salvo los logotipos): gratis para usos comerciales; no necesitan reconocimiento (licencia Pixabay)

A
Números de Nicómaco

Nicómaco de Gerasa, un matemático y filosofo nacido en el siglo I d.C., fue el primero en darse cuenta de que el cuadrado de la suma de los n primeros números es igual a la suma de sus cubos:

$$(1 + 2 + \ldots + n)^2 = 1^3 + 2^3 + \ldots + n^3$$

Los números que cumplen esta propiedad se llaman *números de Nicómaco*. Los cinco primeros son el 1, 9, 36, 100 y 225. El noveno número de Nicómaco es el 2025. La figura siguiente muestra una demostración gráfica para el 5.



Entrada

El programa deberá leer un primer número que indica cuántos casos de prueba deberán ser procesados. A continuación, aparece una línea por cada caso de prueba, con un número entre 1 y 10^9.

Salida

Por cada caso de prueba c el programa escribirá el primer número de Nicómaco que es mayor o igual que c.

Entrada de ejemplo

```
3
200
2025
2026
```

Salida de ejemplo

```
225
2025
3025
```

B
Repartiendo para cuatro

Cuando el pequeño Alex cumplió 5 años, en casa hicieron una pequeña celebración con una tarta casera que tomaron para merendar, pese a que era un día de diario. A la hora de repartirla, su hermana Luna dijo que, como eran cuatro en casa, podían partirla primero en dos partes, y luego cada trozo otra vez en dos partes, para que fuera más fácil.

La verdadera celebración llegó el fin de semana siguiente, cuando fueron a casa los tíos con la prima Miriam. Además, para las velas había otra vez tarta. Alex, recordando el comentario de su hermana, dijo que, para repartirla entre los 7, podían primero partirla en 4 trozos, y luego cada uno otra vez en 3. Todos rieron la ocurrencia y le dijeron que no, que ese reparto no estaba bien, pero él no entendió por qué.

Entrada

Cada caso de prueba es una lista de números enteros positivos terminada en un 0.

La entrada termina con una lista vacía.

Salida

Por cada caso de prueba el programa escribirá en cuántas partes se trocea una tarta si se divide sucesivamente por cada número de la entrada.

Se garantiza que ningún resultado es mayor que 10^9.

Entrada de ejemplo

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

Salida de ejemplo

```
4
12
7
```

C Wheeler y el zumo de limón

En 1995, pocas horas después de robar dos bancos a cara descubierta, McArthur Wheeler fue detenido tras ser identificado gracias a las grabaciones de las cámaras de seguridad. Ante unos policías estupefactos, el detenido entró en cólera gritando que era imposible que pudieran saber que había sido él, pues se había rociado la cara con zumo de limón.

El ridículo plan de Wheeler de entrar en el banco sin taparse la cara "ocultándose" a las cámaras utilizando limón se debía a que éste se utiliza en ocasiones para crear *tinta invisible*. En una descabellada asociación de ideas, pensó que quizá funcionara también en personas. La mala suerte (para él) hizo que la prueba previa que realizó con una cámara Polaroid le confirmara su idea. Se fotografió con zumo por la cara y no salió retratado seguramente porque no apuntó bien al estar cegado por el escozor del limón en los ojos. Y asi, creyéndose transparente para los sistemas de grabación, asaltó, junto a un cómplice, dos bancos antes de ser detenido.

Cuando los expertos en psicología social David Dunning y Justin Kruger oyeron la anécdota, se preguntaron cómo era posible que Wheeler tuviera tanta confianza en sí mismo, pese a su flagrante estupidez. Decidieron probar la frase, atribuída a Darwing, "la ignorancia genera confianza más frecuentemente que el conocimiento".

Con un sencillo experimento descubrieron el que hoy se conoce como el *efecto Dunning–Kruger*. Se trata del sesgo cognitivo que lleva a personas con capacidades limitadas en ciertas áreas a sobreestimar su propia capacidad en esas áreas. Pidieron a un grupo de personas que dijeran cuál era el nivel de destreza que ellos consideraban que tenían en un tema concreto para acto seguido pasarles un test de evaluación sobre él. Los datos sacaron a la luz que todos ellos se sobreestimaban dándose puntuaciones mayores de las que habían sacado en el test y, además, aquellos que menos sabían se sobreestimaban más que los más competentes. La razón del efecto se atribuye principalmente a que los conocimientos necesarios para saber hacer algo correctamente son los mismos que se necesitan para evaluar si algo está bien hecho.

Entrada

Cada caso de prueba comienza con un número $1 \leq n \leq 300.000$ indicando el número de personas evaluadas para analizar si en un determinado ámbito se produce el efecto Dunning–Kruger.

A continuación aparecen dos líneas, cada una con n números entre 1 y 10^9. En la primera aparece la evaluación de cada una de las n personas en lo referente a su desempeño en el ámbito analizado. En la segunda aparece la percepción que cada una de esas personas tiene de sí misma.

Se garantiza que no hay dos personas con la misma evaluación en el test (la primera lista no tiene elementos repetidos) y que todas las personas se sobreestiman algo (para cada persona el valor de la segunda lista es siempre mayor que el de la primera).

Salida

Por cada caso de prueba el programa escribirá `SI` si, en ese grupo de personas, se cumple el efecto Dunning–Kruger y `NO` en otro caso. Se considera que se cumple el efecto si, para cada persona, todas las que hacen mejor la tarea que ella (tienen una puntuación mayor en la primera lista) se sobreestiman menos.

La entrada termina con un `0`.

Entrada de ejemplo

```
4
1 5 10 14
15 14 13 15
4
1 5 10 14
15 14 13 17
4
14 1 5 10
15 15 14 13
0
```

Salida de ejemplo

```
SI
NO
SI
```

D
Cuerdas en un piano

Aunque a muchos profanos les sorprenda descubrirlo, el piano, igual que la guitarra, el violín o el arpa, es un instrumento de cuerda. La forma en la que las cuerdas se usan para producir sonidos en esos instrumentos varía. La familia de los violines (junto a violas, violonchelos y contrabajos entre otros) usan *cuerdas frotadas*. El clavecín, las guitarras, arpas, bandurrias e instrumentos similares usan *cuerdas pulsadas*. El clavicordio y el piano usan *cuerdas percutidas*.

Cuando Johann Sebastian Bach tuvo acceso al predecesor de los pianos, el *fortepiano*, se quejó porque las notas agudas sonaban con mucha menor intensidad que las graves. Para resolverlo, hoy los macillos del registro agudo golpean simultáneamente tres cuerdas, afinadas al unísono, para incrementar el volumen. Por su parte, las notas intermedias golpean solo dos, y las más graves muchas veces solamente una. Eso hace que aunque un piano tenga 88 teclas para las 88 notas de su registro, cuando se afina haya que ajustar muchas más cuerdas.

Entrada

El programa comienza con un número indicando cuántos casos de prueba deben procesarse.

Cada caso de prueba está compuesto por tres números con la información sobre las cuerdas de un determinado piano. El primero indica cuántas teclas golpean sobre 3 cuerdas, el segundo cuántas lo hacen sobre 2 cuerdas y el último cuántas sobre una cuerda. La suma de los tres números es igual a 88.

Salida

Por cada caso de prueba se escribirá el número de cuerdas del piano.

Entrada de ejemplo

```
3
0 0 88
0 88 0
40 30 18
```

Salida de ejemplo

```
88
176
198
```

E
Espectáculo nocturno

Mariluz Sión ha estado de viaje en Suiza y ha venido entusiasmada con un espectáculo nocturno que vió allí. Sobre un edificio proyectaban imágenes en movimiento, sincronizadas con música que animaba a todos los espectadores a bailar e integrarse con el ambiente festivo del momento.

Está decidida a hacer lo mismo en su pueblo, y ha conseguido convencer a un paisano de Torregorrinos de que le deje usar la fachada de su granero, el edificio más grande de los alrededores, durante las fiestas del próximo verano.

Para planificar su espectáculo, ha medido la fachada rectangular, y está colocando cañones de proyección de algunos patrocinadores locales a los que ha engañado para que la apoyen económicamente. Cada cañón proyecta una imagen rectangular de tamaño arbitrario sobre la fachada. Como tiene que ir colocándolos de forma disimulada sobre árboles, farolas y otros elementos existentes para que no se vean, a veces esos rectángulos se solapan y ni siquiera sabe si cubren o no la fachada completa.

Entrada

Cada caso de prueba comienza con tres números. Los dos primeros, $1 \leq w, h \leq 200$ indican, respectivamente, el ancho y el alto de la fachada del granero. El tercero $0 \leq n \leq 200$ indica cuántos cañones de proyección ha conseguido Mariluz.

A continuación aparecen n líneas describiendo la información de cada uno de esos cañones. Un cañón se describe con cuatro números. Los dos primeros, $0 \leq x_i < w$ y $0 \leq y_i < h$, indican la posición en la fachada de la esquina inferior izquierda del rectángulo proyectado por el cañón. Los dos últimos, $1 \leq w_i, h_i \leq 10.000$ indican el ancho y alto del rectángulo.

Todos los números son números enteros sin decimales.

La entrada termina con tres ceros.

Salida

Por cada caso de prueba el programa escribirá el tamaño de la superficie de la fachada que falta por cubrir con algún cañón.

Entrada de ejemplo

```
10 10 1
0 0 10 10
10 10 2
0 0 5 5
5 5 10 10
6 6 2
1 1 4 4
3 3 4 4
0 0 0
```

Salida de ejemplo

```
0
50
15
```

F
Suelo multicolor

Paquito Céspedes está cansado de tener que estar cuidando día sí y día también el jardín de su chalet y está decidido a quitarlo. Llevaba tiempo mirando losetas de exterior para solar el terreno, pero no se decidía hasta que ha visto recientemente un camino de baldosas de distintos colores colocadas en diagonal que le ha gustado.

El problema es que la obra la va a hacer él mismo, y es bastante desastre. Para no equivocarse, quiere tener una representación del resultado que le permita estar seguro de que coloca en cada lugar la loseta que corresponde.

Entrada

El programa deberá leer, de la entrada estándar, múltiples casos de prueba. Cada uno comienza con dos números $1 \leq f, c \leq 40$ con el número de filas y de columnas que tendrá el suelo de losetas. A continuación aparece una cadena con como mucho 20 caracteres ASCII indicando el patrón de colores que se repite. En la cadena no aparecerá ningún espacio.

Salida

Por cada caso de prueba el programa "dibujará" el resultado de colocar las losetas siguiendo el patrón de colores.

En la primera fila, el patrón se repite una y otra vez hacia la derecha. En las filas sucesivas, se repite tras "desplazarlo" una posición hacia la derecha. La primera loseta de la nueva fila será del "color" anterior en el patrón a la que fue la primera en la fila superior. Por ejemplo, en la segunda fila, la loseta de la izquierda será del color de la última loseta del patrón.

Entrada de ejemplo

```
2 2 .O
2 4 .aA
6 10 .*o*
```

Salida de ejemplo

```
.O
O.
.aA.
A.aA
.*o*.*o*.*
*.*o*.*o*.
o*.*o*.*o*
*o*.*o*.*o
.*o*.*o*.*
*.*o*.*o*.
```

G
Serpiente de dominó

El dominó es un juego ancestral cuyos orígenes se remontan al siglo X. Aunque hay muchas variantes, en una de las más difundidas se juega con un grupo de 28 fichas rectangulares distintas, cada una con dos mitades en las que aparecen un grupo de entre 0 y 6 puntos negros. El juego tiene 28 fichas porque hay 28 parejas diferentes de los números del 0 al 6, asumiendo que no importa el orden, pues las fichas no requieren ser usadas con una orientación específica.

Los jugadores comienzan con 7 fichas, que colocan de pie frente a ellos. Deben irlas situando sobre la mesa de forma alterna formando una fila, de manera que dos fichas adyacentes tengan, en las dos mitades que se tocan, el mismo número de puntos negros.

La forma, característica, de las fichas ha llevado a que se utilicen también para crear "serpientes" de dominó. Se colocan de pie formando una fila de modo que cuando una cae golpea a la que tiene delante, que cae también, generando una reacción en cadena que acaba tirando todas.

Paz Iente dedica los ratos libres a montar, en su propia casa, serpientes de dominó enormes. Pone un cuidado escrupuloso, pues un mal golpe tiraría todas las fichas. El trabajo necesita una planificación minuciosa. Antes de empezar, tiene que decidir cómo será la construcción completa para poder empezar por las zonas que terminarán siendo inaccesibles al ir colocando más y más fichas. También situa con mucha antelación las cámaras desde las que grabará la caída de todas las fichas cuando llegue el gran día.

Por si todo ese trabajo no fuera suficiente, Paz insiste en que las fichas tienen además que estar colocadas siguiendo las reglas del juego original. La mitad que toca el suelo de una ficha tiene que tener el mismo número de puntos negros que la mitad superior de la ficha siguiente, o sufrirá pensando que, cuando llegue el momento, algo fallará y no caerán correctamente.

Entrada

Cada caso de prueba comienza con un número $1 \leq n \leq 300.000$ que indica cuántas fichas forman una determinada "serpiente" de dominó. En la línea siguiente aparecen n números de dos dígitos entre 0 y 6 indicando cada ficha. El dígito de la izquierda (el más significativo) indica el número de puntos de la mitad superior de la ficha, y el segundo el de la mitad inferior. Las fichas aparecen en la entrada de tal manera que la primera que se lee va detrás de la segunda y, por tanto, cae antes.

La entrada termina con un 0.

Salida

Por cada caso de prueba el programa escribirá `ERROR` si la "serpiente" de fichas no cumple las reglas del dominó. En otro caso, escribirá cuántos dominós distintos ha tenido que comprar Paz para poder hacerla.

Entrada de ejemplo

```
4
12 23 34 45
3
12 32 34
5
12 21 11 13 31
0
```

Salida de ejemplo

```
1
ERROR
2
```

H
Subiendo y bajando por la muralla

La Gran Muralla China, Patrominio de la Humanidad por la Unesco desde 1987, es una fortificación que comenzó a construirse en el siglo V para proteger el Imperio Chino de sus enemigos del norte. Ha pasado por múltiples reconstrucciones y restauraciones a lo largo de los milenios. En 2012, el gobierno chino anunció que su longitud fue de algo menos que 21.200 kilómetros, contando todas sus ramificaciones y construcciones secundarias, aunque hoy no se conserva entera.

La Gran Muralla serpentea sobre cordilleras, por lo que recorrerla supone subir y bajar continuamente, algo que puede asfixiar a cualquiera. Los responsables del Club de Atletismo local *Khala Khol* quieren poner a prueba a sus socios organizando una carrera en la Gran Muralla en la que los corredores tengan que ascender lo más posible. Para eso, quieren que la salida esté en un lugar bajo, la llegada en un punto alto, y la diferencia entre esas dos posiciones sea la máxima posible.

Entrada

Cada caso de prueba comienza con un número $1 \leq n \leq 400.000$ indicando en cuántos tramos está dividida la Gran Muralla China. En la línea siguiente aparecen n números, no mayores que 2.000 en valor absoluto, indicando el desnivel de cada tramo. Un valor positivo indica que el tramo asciende esa cantidad de metros, y un valor negativo que desciende. Los tramos aparecen en el orden en el que deben recorrerse en la carrera. Se garantiza que al menos uno es ascendente.

La entrada termina con un `0`.

Salida

Por cada caso de prueba el programa escribirá la máxima diferencia posible entre la altura de la salida (que aparece antes en la entrada) y la meta.

Entrada de ejemplo

```
3
10 10 10
5
2 -5 8 -2 6
6
-3 3 -3 3 -3 3
3
1 -10 -10
0
```

Salida de ejemplo

```
30
12
3
1
```

Fregando por no pagar

No es la primera vez que Rodrigo Ron va a restaurantes a comer o a cenar y, cuando llega el momento, se marcha corriendo dejando la cuenta sin pagar. Pero esta vez le han interceptado a mitad de la carrera y, además de la correspondiente denuncia y multa, le han puesto a fregar los platos para compensar los gastos.

Y no es una tarea menor. Según van recogiendo las mesas de los clientes, los camareros apilan más y más platos a su lado. Rodrigo va cogiendolos de arriba de uno en uno. Algunos vienen más sucios que otros y tarda un tiempo variable en limpiar cada uno, dependiendo de su estado.

Entrada

La entrada comienza con un número $1 \leq n \leq 200.000$ indicando cuántos platos han ido llegando a la cocina para que Rodrigo los friegue. A continuación aparecen, en una o varias líneas, n parejas de números, cada uno con la información de un plato, en orden de llegada. El primer número de la pareja (entre 0 y 10^9) indica el instante en el que el plato es apilado para ser fregado. El segundo indica cuánto tardará Rodrigo en fregarlo (entre 1 y 10^9).

Es posible que dos platos lleguen en el mismo instante. Cada vez que Rodrigo termina de fregar un plato coge inmediatamente el situado en la parte de arriba y comienza a limpiarlo. Si en el momento en el que Rodrigo termina de fregar un plato llegan platos nuevos, cogerá el último que llegue en ese momento.

Se garantiza que el primer plato llega en el instante 0, y que el momento de llegada de los platos está ordenado de menor a mayor.

La entrada termina con un caso sin platos.

Salida

Por cada caso de prueba el programa escribirá, separados por espacios, el número de platos que tiene aún pendientes de fregar cada vez que Rodrigo va a coger el siguiente plato. Cuando los camareros dejan de llevar platos a fregar, el número de platos pendientes irá bajando siempre de uno en uno. Por ese motivo, el programa dejará de escribir números después de que Rodrigo empiece a fregar el último plato llevado por los camareros.

Se garantiza que siempre que Rodrigo termina de fregar un plato hay al menos otro esperándole.

Entrada de ejemplo

```
3
0 1
1 1
2 1
3
0 3 0 3 1 2
0
```

Salida de ejemplo

```
1 1 1
2 2
```

J
Dragon's Lair

En 1983, Don Bluth, exanimador de Disney, publicó el videojuego *Dragon's Lair* para laserdisc. En aquella época el *hardware* de los ordenadores y videoconsolas era muy limitado. El laserdisc, un antiguo soporte para películas, permitió hacer uso de *vídeo de movimiento completo*. El juego estaba troceado en infinidad de pequeñas secuencias de animación pregrabada, con un estilo y calidad que recordaba a las películas de Disney de la época. El jugador podía elegir cada pocos segundos las acciones del protagonista, para que tomara un camino u otro, o atacara o no a los enemigos. En función de las decisiones tomadas, se saltaba a una sección distinta, haciendo evolucionar la historia.

Hoy se conoce a este género de videojuegos como *películas interactivas* y la idea ha ido olvidándose y volviendo a surgir varias veces a lo largo de las décadas. Crearlas supone construir un guion que se ramifica y, en función de él, crear pequeñas secciones de vídeos que, unidos, hacen que el espectador pueda ver la película que él mismo construye. En función del guion, puede haber distintas formas de llegar a un mismo punto intermedio y acabar en múltiples finales, algunos favorables y otros negativos.

Entrada

Cada caso de prueba comienza con un número n entre 2 y 20.000 indicando en cuántas secuencias se encuentra dividido el guion de una película interactiva. A continuación aparecen n líneas describiendo cada una de ellas, por orden.

Una secuencia se describe comenzando con un número e que indica entre cuántas opciones se puede elegir el siguiente paso. A continuación aparecen e números distintos entre 1 y n con el índice de la secuencia a la que la película salta por cada opción. En total no habrá nunca más de 200.000 opciones.

Las secuencias que se corresponden con un final de la película no ofrecen opciones. En ese caso, tras el `0` que indica que el jugador no puede elegir el siguiente paso, aparece el texto `FELIZ` o `AMARGO` para indicar si en ese punto la película interactiva acaba o no de forma feliz para el espectador.

La primera secuencia descrita, la 1, es el punto de inicio de la película. Se garantiza que todas las secciones son alcanzables desde ese inicio. Además, aunque puede haber varias formas de llegar a la misma secuencia del guion, la película *siempre avanza*. Se garantiza que no hay *ciclos* que hagan que el espectador pueda volver una y otra vez a la misma secuencia.

Salida

Por cada caso de prueba el programa escribirá cuántas *visualizaciones completas distintas* pueden conseguirse con el guion de la película interactiva indicada, proporcionando primero cuántas terminan en un final feliz y luego cuántas lo hacen en un final amargo. Como los números pueden ser muy altos se proporcionará el resto de dividir cada uno por 1.000.000.007.

Entrada de ejemplo

```
3
2 2 3
0 FELIZ
0 AMARGO
5
2 4 5
0 FELIZ
0 AMARGO
2 2 3
1 3
```

Salida de ejemplo

```
1 1
1 2
```