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

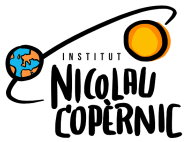
ProgramaMe 2023
Regional de Villaviciosa de Odón y Terrassa

Problemas

Ejercicios realizados por

Universidad Complutense
de Madrid

Realizado en la **Universidad Europea** (Villaviciosa de Odón) y en el **INS Nicolau Copèrnic** (Terrassa)
29 de marzo de 2023



29 de marzo de 2023
https://www.programame.com

Listado de problemas

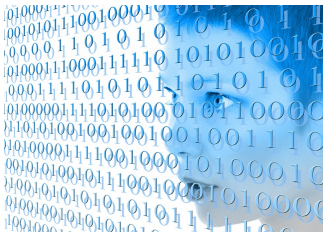
Autores de los problemas:

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

Revisores:

- Iván Cantón Sáez (Institut Nicolau Copèrnic – Terrassa)
- Albert Grau (Institut Nicolau Copèrnic – Terrassa)
- Manuel Orós (Institut Nicolau Copèrnic – Terrassa)
- Alberto Verdejo (Universidad Complutense de Madrid)

A
Binario en decimal



Cuando a Ann Osen Tera le explicaron los números binarios lo entendió todo mal. En clase, su profesor les dijo que con ellos se podía representar cualquier número usando únicamente los dígitos 0 y 1. También oyó que había sumas y multiplicaciones por algún lado, y con esas frases sueltas se hizo su propia idea de lo que eran.

Y lo que pasó era de esperar. Cuando llegó el día del examen y el profesor les pidió que convirtieran a binario varios números escritos en decimal, Ann dio una respuesta completamente disparatada:

```
2 = 1 + 1
6 = (1 + 1 + 1) * (1 + 1)
24 = 1 + 11 + 11 + 1
120 = 110 + 10
```

Ann pensaba que pasar un número a binario significaba ¡escribir una expresión de sumas y multiplicaciones de números que solo tuvieran los dígitos 0 y 1! Así por ejemplo, para escribir el 12 lo que hacía sería sumar 1 y 11.

Lo peor de todo es que aunque la respuesta estaba completamente mal, contestarla le costó mucho tiempo porque buscaba la expresión con el menor número de operaciones, y eso no siempre era fácil.

Entrada

La entrada estándar contendrá una sucesión de números, en decimal, entre 1 y 10.000 terminada por un 0, que no deberá procesarse.

El programa deberá ser capaz de procesar, en una sola ejecución, un considerable número de casos.

Salida

Por cada número de la entrada, el programa escribirá el menor número de operaciones que tendrá la "representación en binario" entendida por Ann. Si un número ya está "en binario" (solo tiene dígitos 0 y 1) se escribirá `0`.

Entrada de ejemplo

```
2
6
24
1111
0
```

Salida de ejemplo

```
1
4
3
0
```

B
Día Mundial del Piano

Todos los días del año están dedicados a algún tipo de celebración a través de los llamados *días internacionales* o *días mundiales*. Por ejemplo, el día 8 de marzo es el Día Internacional de la Mujer, y el 9 de marzo el Día Mundial de la Tortilla de Patata.

El piano no podía escaparse de esta costumbre y también tiene su Día Mundial del Piano. Pero es particular porque la fecha no siempre es la misma. Lo normal es que se celebre el 29 de marzo pero los años bisiestos se celebra el día antes, el 28 de marzo. La razón de esta fluctuación es muy sencilla. Se eligió como fecha de celebración el 29 de marzo porque es el día número 88 del año, que coincide con el número de teclas de un piano. Pero esta regla no se cumple los años bisiestos, por lo que en esos años la celebración se tiene que mover al día anterior.

Entrada

La entrada comienza con un número que indica cuántos casos de prueba deberán ser procesados.

Cada caso de prueba son dos números representando dos años entre 1600 y 2400. Se garantiza que el primer número no será nunca mayor que el segundo.

Salida

Por cada caso de prueba el programa escribirá dos números, el primero indicando cuántos años del rango se celebró, o se celebrará, el Día Mundial del Piano el día 29 de marzo, y el segundo indicando en cuántos se hará el 28.

Recuerda que un año es bisiesto si es divisible por 4, salvo que sea divisible por 100, en cuyo caso para ser bisiesto debe también serlo por 400. Por ejemplo, el año 2000 fue bisiesto (es divisible por 100 y por 400) pero el 2100 no lo será (es divisible por 100, pero no por 400).

Entrada de ejemplo

```
4
2020 2023
1999 2001
2099 2101
1600 1699
```

Salida de ejemplo

```
3 1
2 1
3 0
75 25
```

C
El cuadrado del cinco

Todo el mundo sabe que el cuadrado del número 5 es 25. Pero que el 5 sea exactamente la mitad del 10, que es nuestra base de numeración habitual, ocasiona una propiedad curiosa que salta a la vista si escribimos los cuadrados de los números acabados en 5:

n	n^2
5	25
15	**2**25
25	**6**25
35	**12**25
45	**20**25

El cuadrado de todos los números que terminan en 5 acaba en 25. Y ¡no solo eso! Los dígitos que van delante de ese 25 se pueden calcular fácilmente a partir del número original quitando el último dígito. Por ejemplo, 35^2 es 1225. Si al 35 le quitamos el 5, nos queda un 3. Si lo multiplicamos por él mismo sumado 1 obtenemos un 12 (3×4) que es ¡el número resultante de quitar el 25 en el cuadrado del 35! Esto ocurre con el cuadrado de cualquier número que acaba en 5, y no es difícil demostrar por qué.

Entrada

La entrada comienza con un número que indica cuántos casos de prueba deberán ser procesados. Cada uno es un número terminado en 5 menor que 2×10^{10}.

Salida

Para cada caso de prueba n el programa escribirá su cuadrado, n^2.

Entrada de ejemplo

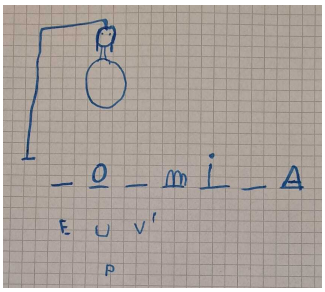
```
5
5
15
25
35
45
```

Salida de ejemplo

```
25
225
625
1225
2025
```

D

El verdugo



El ahorcado es un conocido juego de adivinanzas de lápiz y papel para dos jugadores. Uno de ellos piensa una palabra (a veces una frase completa) y el oponente trata de adivinarla. Para eso, el jugador adivinador va proponiendo letras, que el otro debe ir colocando en la palabra secreta. Si alguna de las letras no aparece en ella, cuenta como un fallo.

Para anotar el número de fallos acumulados, se va completando la figura de un hombre ahorcado, lo que da nombre al juego. Cuando está completamente formada, se considera que el número de fallos es excesivo y el adivinador pierde la partida. La imagen es de un gusto cuestionable, por lo que en algunos contextos se utilizan figuras alternativas, como manzanos a los que se le caen las manzanas.

El número de oportunidades para el adivinador también varía, dependiendo de los detalles que se añadan a la figura. Es habitual que al séptimo fallo el juego se considere terminado.

Dada una palabra secreta y las letras propuestas para descubrir, tu labor es hacer de verdugo y decidir si finalmente el hombre es o no ahorcado.

Entrada

Cada caso de prueba está formado por dos líneas, cada una con una secuencia de no más de 100 letras del alfabeto inglés sin espacios.

La primera representa la palabra oculta. La segunda contiene las letras propuestas por el adivinador, entre las que podría haber algunas repetidas.

Un punto (“.”) marca el final.

Salida

Para cada caso de prueba, el programa escribirá “`AHORCADO`” si el número de letras falladas es mayor o igual que 7. En otro caso, se escribirá “`SALVADO`” si la palabra secreta ha sido desvelada completamente, y “`COLGANDO`” si la partida está aún a medias.

El adivinador podría repetir algunas letras. En ese caso, las segundas apariciones no contarán nunca como fallo (lo fueran o no en su primera aparición). Además, en la entrada podrían aparecer letras posteriores dadas por el adivinador después de haberse salvado o de haber sido condenado, que deben ignorarse.

Entrada de ejemplo

```
ahorcado
ioaelmnps
jazz
aeiiiiii
abc
abcdefghijklmnopqrstuvwxyz
.
```

Salida de ejemplo

```
AHORCADO
COLGANDO
SALVADO
```

E
Facundo y el undo

Facundo se equivoca continuamente cada vez que escribe con el teclado. Tener que retroceder el cursor para corregir sus errores le resulta tan frustrante que al final, cada vez que escribe una palabra mal, prefiere borrarla entera y volver a escribirla desde cero.

Un día, Facundo descubrió en su editor de textos la combinación de teclas <Ctrl+Z>, que deshace la escritura de la última palabra introducida. De este modo, pulsando n veces seguidas la combinación <Ctrl+Z> se pueden borrar las n últimas palabras escritas. Eso sí, funciona siempre que aún queden palabras en el texto. Si no, la combinación de teclas no hace nada.

Facundo se entusiasmaba tanto con la combinación de teclas <Ctrl+Z> que a veces borraba más palabras de la cuenta. Por suerte, ha descubierto que su editor también tiene una funcionalidad que, mediante la combinación de teclas <Ctrl+Y>, le permite *rehacer* la última acción deshecha. Si justo después de deshacer la escritura de una palabra con <Ctrl+Z> pulsa <Ctrl+Y>, la palabra borrada reaparece otra vez. Puede hacer esto ¡hasta rehacer todo lo deshecho! Es más, si sigue pulsando <Ctrl+Y> cuando no queda nada que deshacer, la última operación *se repite*. Eso sí, las acciones que se pueden rehacer se olvidan si se añade una nueva palabra.

Entrada

Cada entrada comienza con un número indicando cuántos casos de prueba vienen a continuación.

Cada caso de prueba consiste en una línea que contiene las acciones realizadas por Facundo en el editor de texto. El símbolo < indica que ha pulsado la combinación <Ctrl+Z> (deshacer). El símbolo > indica que ha pulsado la combinación <Ctrl+Y> (rehacer). Cualquier otra secuencia de caracteres indica que Facundo ha introducido la palabra correspondiente en el editor. Las palabras están formadas por como mucho 20 letras del alfabeto inglés, en mayúscula o minúscula. La última palabra, que no debe procesarse, es un punto `.` que marca el final. Ningún caso de prueba tiene más de 200 palabras o teclas rápidas.

Salida

Para cada caso de prueba debe imprimirse una línea con las palabras que quedan en el editor tras realizar todas las acciones. Las palabras han de estar separadas por un espacio.

Entrada de ejemplo

```
4
Me llaom < llamo facundo < Facundo y boy a Guadalajara < Toledo < < < voy a Toledo .
Cuatro < < Cinco lobitos tiene < > la lova < < < > > loba .
> Ole ole > > > .
Ole ole < hola > .
```

Salida de ejemplo

```
Me llamo Facundo y voy a Toledo
Cinco lobitos tiene la loba
Ole ole ole ole ole
Ole hola hola
```

F
La carrera del salmón

Los salmones nacen en los lechos de grava de arroyos y de la parte alta de diferentes ríos. Tras una fase larvaria y de juventud que puede llegar a durar 3 años, sus cuerpos se transforman para adaptarse al agua salada y viajan, río abajo, hacia el océano. Allí pasan buena parte de su vida adulta, que puede durar 4 o 5 años, hasta que están preparados físicamente para afrontar *la carrera del salmón*.

Ésta consiste en la vuelta de los salmones al lugar donde nacieron, haciendo uso de un asombroso sentido de la orientación, para desovar. Significa tener que recorrer un trayecto que puede alcanzar miles de kilómetros, río arriba, a contracorriente, ascendiendo varios miles de metros. Para poder conseguir semejante proeza necesitan disponer de altas capacidades de natación y la energía suficiente para superar los rápidos de los ríos y otros muchos peligros, como los depredadores o los obstáculos artificiales.

Durante el ascenso, se ven obligados a hacer pequeños descansos que, debido a la corriente, les hacen retroceder parte de lo avanzado.

Entrada

El primer número de la entrada indica cuántos casos de prueba deberán ser procesados. Cada uno está compuesto por tres números, todos entre 1 y 10.000, indicando, respectivamente, la longitud del recorrido que debe realizar, río arriba, un salmón, la longitud que es capaz de avanzar sin descansar, y cuánto retrocede durante el descanso.

Salida

Por cada caso de prueba el programa escribirá cuántos ciclos de avance–descanso necesita el salmón para terminar su ascenso. Ten en cuenta que el salmón siempre empieza ascendiendo, y que en la posición destino no hay corriente por lo que, una vez alcanzada, el descanso ya no hace retroceder al salmón.

Si es imposible llegar al final del recorrido, se escribirá `IMPOSIBLE`.

Entrada de ejemplo

```
3
100 100 10
10 5 2
10 5 6
```

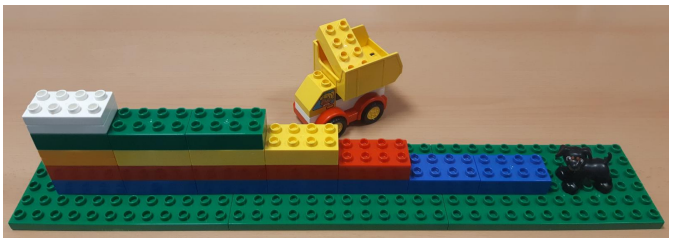
Salida de ejemplo

```
1
3
IMPOSIBLE
```

G
La escalera de Alba

A sus 6 años, Alba Ñil es muy aficionada a los juegos de construcción. Tiene un montón de *ladrillitos* de diferentes colores que utiliza para crear infinidad de cosas. Hoy ha decidido que va a hacer una escalera para que su perrillo Cem Ento (en realidad, una pieza del mismo juego) haga ejercicio.

Para eso ha cogido sus 7 piezas azules y las ha puesto formando una hilera. Luego ha colocado encima, empezando por un lateral, las 5 piezas rojas que tiene. El siguiente "peldaño", amarillo, lo ha hecho con 4 piezas. Tiene una quinta amarilla pero no puede ponerla. ¡Cem Ento no puede subir dos peldaños de golpe! Después ha puesto las 3 piezas verdes que tiene y finalmente la única blanca.



Se ha puesto a buscar en su baúl y ha encontrado más piezas, algunas de los colores anteriores y otras de colores nuevos. Alba quiere hacer la escalera con más piezas posible de modo que cada nivel sea de un único color, y que todos sean diferentes entre sí. Además, un nivel tiene que tener siempre menos piezas que el que tiene debajo, porque de otro modo sería una escalera mal formada.

Entrada

Cada caso de prueba comienza con un número que indica de cuántos colores distintos tiene Alba piezas en el juego de construcción. A continuación va, en otra línea, el número de piezas que tiene de cada uno de esos colores. Salvo por el color, todas las piezas son iguales. El número de colores y el de piezas de cada color es menor o igual que 200.000.

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

Salida

Por cada caso de prueba el programa escribirá el máximo número de piezas que se pueden utilizar para hacer una escalera siguiendo las restricciones de Alba.

Entrada de ejemplo

```
5
7 5 5 3 1
3
1 2 3
4
5 7 7 5
0
```

Salida de ejemplo

```
20
6
22
```

H
Metidos en un cajón

Hay un famoso acertijo para niños que dice "tengo 5 patos metidos en un cajón; ¿cuántos picos y patas son?". La cantidad de patos depende de la edad de la víctima a la que se le dispare la pregunta. Y, de hecho, aunque el objetivo es que el acertijo se entienda tal y como se ha escrito, lo que el adulto está pensando es "tengo 5 patos; *metí dos* en un cajón", que cambia completamente la respuesta correcta.

Pero si entendemos la pregunta en su forma original, dado un número de patos ¿cuántos picos y patas son?

Entrada

La entrada comienza con 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 100.000.

Salida

Por cada caso de prueba el programa deberá escribir, separados por un espacio, cuántos picos y cuántas patas tienen en total el número de patos indicado en la entrada.

Entrada de ejemplo

```
3
5
6
1
```

Salida de ejemplo

```
5 10
6 12
1 2
```

Termoclastismo

Las grandes rocas pueden fracturarse de forma natural debido a un proceso conocido como *meteorización física*. Hay muchos motivos para que ocurra, como la *gelifracción* (el agua mantenida en cavidades interiores se congela y dilata, fracturando la piedra), el *haloclastismo* (las sales solubles se introducen en el interior de la roca y por cambios físicos y químicos se expanden causando la rotura) o el *termoclastismo*.

Este último tipo ocurre debido a cambios bruscos de temperatura (meteorológicos o por fuego) que ocasionan que la propia roca se contraiga y se dilate bruscamente, lo que puede terminar ocasionando que se fragmente. Para eso, la temperatura debe fluctuar de manera pronunciada en un tiempo relativamente corto.

Entrada

Cada caso de prueba comienza con un número entre 1 y 200.000 indicando el número de muestras de temperatura que se han recogido. A continuación, en otra línea, aparecen dichas temperaturas, números enteros entre –50 y 300.

La entrada termina con un caso de prueba sin muestras.

Salida

Por cada caso de prueba el programa escribirá, separadas por espacio, la menor y mayor temperatura alcanzadas, así como la menor distancia entre ellas. Se entiende por "distancia" a la diferencia entre sus posiciones en la lista de muestras.

Entrada de ejemplo

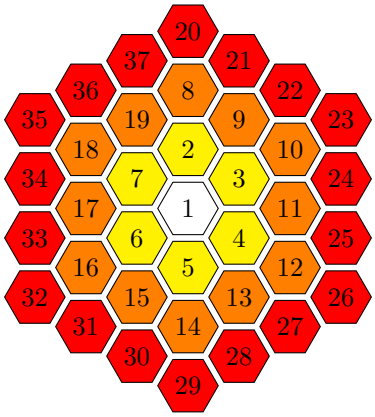
```
4
1 2 3 4
5
25 10 20 40 10
3
30 30 30
0
```

Salida de ejemplo

```
1 4 3
10 40 1
30 30 0
```

J
Vecinas del panal

Cuando construímos los panales, las abejas hacemos la primera celda en el centro y le damos el número 1. Luego vamos construyendo "*anillos*" de celdas a su alrededor, y las numeramos de manera consecutiva en el sentido de las agujas del reloj. Cada vez que hay que comenzar un nuevo anillo, lo hacemos colocando una nueva celda encima de la primera celda del anillo anterior (lo que los humanos llamaríais "*a las 12*" o "*al norte*").



Este proceso es simple, sistemático y controlado, como nos gusta a las abejas. Pero la verdad es que orientarse en el panal es complicado. Yo, que vivo en el extrarradio, ni siquiera tengo muy claro cuáles son los números de celda de mis vecinas.

Entrada

El programa deberá leer, de la entrada estándar, varios casos de prueba cada uno en una línea. Un caso de prueba es un número entre 1 y 1.000.000.000 representando una celda dentro de un panal que sigue la numeración anterior.

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

Salida

Para cada caso de prueba el programa escribirá, por la salida estándar, las celdas adyacentes a la dada, ordenadas de menor a mayor y separadas por espacio.

Entrada de ejemplo

```
1
2
3
0
```

Salida de ejemplo

```
2 3 4 5 6 7
1 3 7 8 9 19
1 2 4 9 10 11
```