Concurso de programación

Las 12 UVas 2017

Problemas


Patrocinado por

accenture technology

Ejercicios realizados por

Facultad de Informática
Universidad Complutense de Madrid

Organizado desde la **Facultad de Informática (UCM)**
31 de diciembre de 2017

Universidad Complutense Madrid

31 de diciembre de 2017
http://las12uvas.es

Listado de problemas

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
Racimos de uvas

Los racimos de uvas perfectos, los bonitos, los de pata negra, son los que tienen una uva abajo del todo, y luego, al ir subiendo, el número de uvas del nivel siguiente se incrementa, como mucho, en una uva cada vez. Dicen que los mejores sumilleres son capaces de identificar, no sólo los taninos, aromas de madera o corpulencia de un buen vino, sino también si los racimos que se usaron eran o no perfectos. Yo, francamente, no me lo creo.

Sin embargo, por si acaso, los enólogos y los responsables de las vides sí tienen en consideración la forma de los racimos. Puestos a elegir, prefieren aquellos que, además de ser perfectos, tienen el menor número de niveles posible para la cantidad de uvas que contienen. De esa forma se consiguen racimos más compactos, que son más fáciles de almacenar antes de ser prensados.

Entrada

El programa deberá leer de la entrada estándar múltiples casos de prueba. Cada uno contendrá un único número $1 \leq n \leq 1.000.000$ indicando el número de uvas de un racimo.

La entrada termina con un caso con 0 uvas que no deberá procesarse.

Salida

Para cada caso de prueba el programa escribirá el menor número de niveles posible que deberá tener un racimo perfecto con n uvas, de modo que tanto sumilleres como enólogos estén contentos con él.

Entrada de ejemplo

```
10
40
0
```

Salida de ejemplo

```
4
9
```

B
Romance en el palomar

Higinio es un abuelito venerable empeñado en mantener a flote su negocio de palomas mensajeras. Desde hace más de 60 años tiene en su pueblo perdido de las arribes del Duero un palomar al que dedica sus horas llenándolo de cuidados y cariño.

Para que el palomar no se vaya a pique es muy importante que la población de palomas se mantenga. Y para evitar tener que ir a ferias a comprar las aves, lo mejor es que los romances se den entre los ejemplares del propio palomar. Y la verdad es que ahí Higinio siempre se ha considerado privilegiado: todas las mañanas, invariablemente desde hace 60 años, descubre que en alguno de los habitáculos ha dormido más de una paloma.

Sin embargo hoy su hermana Leonor le ha puesto en alerta. Dice que ha leído en la güiquipedia (está seguro de que no se escribe así...) que los matemáticos tienen desde 1834 lo que llaman el "principio del palomar". Ese principio establece que si n palomas se distribuyen en m habitáculos y se tiene que $n > m$, entonces habrá al menos un habitáculo en el que haya más de una paloma. Es decir, que las noches en las que bloquea, por limpieza, el uso de demasiados habitáculos, igual en lugar de tener romances tiene a los animales amontonados por obligación...

Entrada

La entrada comienza con una línea con un número que indica el número de casos de prueba que siguen.

Cada caso de prueba, en una línea independiente, contiene los datos de una noche en el palomar: número de ejemplares que duermen en él (al menos dos) y número de cajas disponibles. Ninguno de los números será mayor que 1.000.000.

Salida

Sabiendo que siempre se encuentra dos palomas en al menos un habitáculo, por cada caso de prueba se escribirá una línea que indicará si Higinio puede estar seguro de que esa noche ha habido un romance (escribiendo `ROMANCE`) o puede ser cosa del principio del palomar (escribiendo `PRINCIPIO`).

Entrada de ejemplo

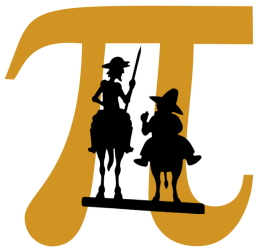
```
2
2 5
5 2
```

Salida de ejemplo

```
ROMANCE
PRINCIPIO
```

C

El Pijote



¿Se puede encontrar El Quijote entre los decimales del número π? La pregunta puede parecer extraña, expliquémonos. Ya sabes que los caracteres se pueden codificar con números, por ejemplo utilizando el código ASCII o UTF-8 o cualquier otro. Pues bien, si escribimos el número (bien largo) que consiste en la concatenación de los códigos de cada caracter del texto de El Quijote en ese mismo orden, ¿podríamos encontrar dicho número dentro de la lista de decimales del número π? Lo mismo podríamos preguntarnos con cualquier otro texto, como el del programa que resuelve este problema, o la concatenación de *todos* los programas que resuelven este problema.

Pues la respuesta es que *no se sabe*. Y dirás tú, "Ya me imaginaba". Pero el asunto no termina ahí. Los matemáticos han determinado las tres condiciones que debería cumplir la lista de decimales de π para que cualquier texto aparezca en algún lugar de esa lista. La primera es que sea tan grande como queramos. Eso es fácil, π tiene infinitos decimales. La segunda es que los decimales no tengan un patrón que se repita. Eso también es fácil, ya que π es un número *irracional*, que significa justamente eso. La tercera condición pide que la lista de decimales sea un número *normal*, que significa que cualquier número (de 1, 2, 3,... cifras) aparezca dentro de la lista de decimales el mismo número de veces (es decir, que no hay preferencia por ningún número). Pues bien, no se sabe si los decimales de π son un número normal o no (hay bastantes indicios de que sí, pero aún no se ha demostrado).

Si estás un poco decepcionado con la conclusión, no te preocupes, sí hay números que cumplen las tres condiciones. Por ejemplo, el número de Champernowne, que es el 0 seguido de los decimales formados concatenando todos los números naturales, 0,123456789101112131415161 7...

Tranquilo, no te vamos a pedir que demuestres que los decimales de π son un número normal. Por ahora estamos interesados en una propiedad más sencilla: saber si una secuencia de esos decimales es *subnormal*, es decir, que aparecen en ella los dígitos del 0 al 9 el mismo número de veces. ¿Nos ayudas?

Entrada

La entrada está formada por una serie de casos de prueba. Cada caso consiste en una secuencia no vacía de hasta 1.000 dígitos del `0` al `9`.

Salida

El programa escribirá una línea por cada caso de prueba. Si la secuencia dada es subnormal se escribirá `subnormal` y, en caso contrario, se escribirá `no subnormal`.

Entrada de ejemplo

```
314159
9876543210
00112233554466779898
```

Salida de ejemplo

```
no subnormal
subnormal
subnormal
```

D
¡Reto superado!

Me encantan los programas de televisión en los que se proponen un reto, que muchas veces parece realmente imposible, pero que gracias a la tenacidad y perseverancia de los colaboradores del programa terminan finalmente consiguiendo. Y todos se ponen muy contentos.

En el programa de ayer pretendían superar la altura de un edificio simplemente doblando sucesivamente por la mitad y sobre sí misma una hoja de papel. El papel era bastante grande, pero tan fino como una hoja de escribir. Increible, ¿verdad?

Los primeros dobleces costaban por lo grande del papel, y varias personas tenían que tirar de un extremo para ir sobreponiendo una mitad sobre la otra. Pero cuando ya llevaban muchos pliegues hechos, lo difícil era poner una mitad sobre la otra y que quedaran planas. Tuvieron que recurrir a una apisonadora. ¡Pero lo lograron!

Dado el grosor de la hoja de papel, medido en micras (1 micra = 10^{-6} metros), y la altura del edificio, medida en metros, ¿sabrías calcular cuántos dobleces hacen falta para conseguir superar el reto?

Entrada

La entrada estará formada por una serie de casos de prueba, cada uno en una línea.

Un caso de prueba consiste en dos números: el grosor del papel (medido en micras), un número entre 1 y 1.000, y la altura del edificio (medida en metros), un número entre 1 y 10.000.

Salida

Para cada caso de prueba se escribirá una línea con el número de dobleces necesarios para superar la altura del edificio con un papel de ese grosor.

Entrada de ejemplo

```
1 1
2 1
1 10
```

Salida de ejemplo

```
20
19
24
```

E
El anuncio más caro del año

En España, la tradición de comer 12 uvas con las primeras (¿o son las últimas?) campanadas del año lleva a todo el país a conectar la televisión para ver la retrasmisión desde algún reloj emblemático. Eso supone unas *cuotas de pantalla* irrepetibles durante el resto del año, y las cadenas aprovechan para pedir a los anunciantes una gran cantidad de dinero por emitir su anuncio. Cuanto más cerca está el anuncio de las campanadas más gente lo verá, lo que lleva a que el último anuncio emitido del año se convierte normalmente en el más caro del año completo.

En la Nochevieja de 2014–15, los reponsables de Canal Sur intentaron acercar tanto el último anuncio a las campanadas que... se pasaron y lo pusieron *durante* las propias campanadas, dejando a decenas de miles de andaluces sin comer las uvas. Sabiendo que las campanadas comienzan exactamente a las 12:00 de la noche, y dada la duración del último anuncio, ¿cuál es la hora límite para comenzar su emisión?

Entrada

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

Cada caso de prueba indica la duración del último anuncio, en formato `HH:MM:SS` (horas, minutos y segundos). La entrada cumple las reglas de las horas ($0 \leq$ `HH` < 24, $0 \leq$ `MM`,`SS` < 60). Los anuncios duran al menos un segundo.

Salida

Para cada caso de prueba se indicará la hora más tardía a la que puede empezar el anuncio, en formato `HH:MM:SS`, siguiendo también las restricciones habituales de las horas.

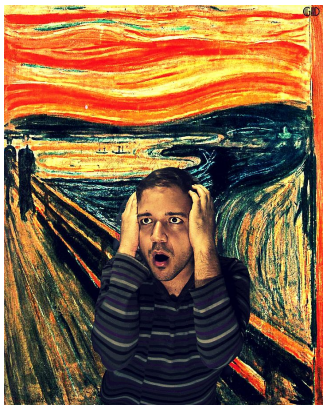
Entrada de ejemplo

```
2
00:00:30
01:00:59
```

Salida de ejemplo

```
23:59:30
22:59:01
```

F
Esgritura



Hoy en día se está perdiendo el uso del signo de exclamación al inicio de las frases. Quizá por el deseo de rapidez al escribir, por la influencia de otros idiomas (principalmente el inglés) o, simplemente, por pereza, mucha gente ha dejado de usar un símbolo único de nuestra lengua.

Resulta irónico que, por el contrario, a veces parezca que se compensa esta ausencia de los símbolos de apertura repitiendo el de cierre más allá de toda lógica.

Luis Piedrahita está siempre preocupado por encontrar palabras para dar nombre a los nuevos conceptos. Tras poner su atención en estos textos pródigos en signos de exclamación y parcos en contenido, ha propuesto para ellos la palabra *esgritura*, que define tal que así: "La *esgritura* es la acción o efecto de *esgritar*. Es la representación de conceptos o ideas, con más signos de exclamación que letras. Es un texto entusiasta; un grito escrito".

Estás desarrollando un nuevo sistema de mensajería instantánea y quieres detectar *esgritos* para ponerles un tamaño de fuente mayor, de acuerdo a su énfasis implícito. ¿Eres capaz de detectarlos?

Entrada

El programa deberá leer, de la entrada estándar, los mensajes recibidos desde el servidor por la aplicación de mensajería instantánea, cada uno en una línea. Estarán compuestos de entre 1 y 1.000 caracteres, formados únicamente por letras del alfabeto inglés, espacios, signos de exclamación (solo de cierre, por supuesto), y otros signos ortográficos.

Salida

Por cada línea de la entrada, el programa escribirá `ESGRITO` si el texto tiene más signos de exclamación que letras (del abecedario inglés), y `escrito` en caso contrario.

Entrada de ejemplo

```
Hola!!!!!
No puedo ir :(
22!
Uau!!!
Si!! Si!!!
:-O!!
```

Salida de ejemplo

```
ESGRITO
escrito
ESGRITO
escrito
ESGRITO
ESGRITO
```

G
Velocidad = desplazamiento/tiempo

Javier es profesor de física, y está comenzando a explicar a sus alumnos la *cinemática*, la rama de la física que describe el movimiento de los objetos sólidos sin considerar las causas que lo originan (es decir, las fuerzas), estudiando su trayectoria en función del tiempo. Para ello se basa en conceptos como *velocidad* y *aceleración*.

Por ahora les ha explicado que la velocidad se determina como el cociente entre el desplazamiento (espacio recorrido) y el tiempo empleado en recorrerlo, según la ecuación

$$velocidad = \frac{desplazamiento}{tiempo}$$

Para que se familiaricen con esta fórmula, les va a proponer una serie de preguntas cortas, donde les proporcionará dos de los datos (desplazamiento, tiempo o velocidad) y ellos tendrán que contestar con el tercero.

A Carlos, uno de sus alumnos, le gusta más programar que aplicar fórmulas, por lo que ha decidido escribir un programa que resuelva las preguntas por él.

Entrada

La entrada comienza por un número N que indica el número de preguntas que vendrán a continuación. Cada una de ellas aparece en una línea distinta.

En una pregunta aparecen dos datos, que corresponderán a un desplazamiento, un tiempo o una velocidad. Un dato consiste en una letra (`D` para deplazamiento, `T` para tiempo, o `V` para velocidad), el símbolo `=` y un valor entre 1 y 10.000. Los datos pueden darse en cualquier orden y están separados por un espacio.

Salida

Para cada pregunta, el programa escribirá la respuesta, utilizando la misma notación empleada para las preguntas. Se garantiza que todos los resultados son números enteros.

Entrada de ejemplo

```
3
D=32 T=4
T=4 V=8
V=8 D=32
```

Salida de ejemplo

```
V=8
D=32
T=4
```

H
Escalando el Everest

Escalar el Everest a mediados del siglo XX era una hazaña al alcance de muy pocos. Sin embargo, a partir del año 1985, comenzó la "comercialización del Everest" con una expedición que alcanzó la cima y que incluía a un hombre de negocios de 55 años y con tan solo 4 años de experiencia de escalada.

Hoy día hay numerosas compañías que ofrecen visitas guiadas a la montaña a ricos con ganas de emociones pero sin experiencia en escalada. Eso hace que la llegada a la cumbre sea ahora un hecho casi rutinario, pero cargado de peligros.

Cuando las condiciones meteorológicas son propicias, son muchos los que se agolpan en la base de la montaña para comenzar su aventura. Eso hace que se formen largas hileras de escaladores por las distintas rutas de ascensión, con habilidades dispares y, por tanto, velocidades de subida distintas. Esas distintas velocidades, unidas a la imposibilidad de "adelantamientos" en las zonas cercanas a la cima hace que se originen "atascos" que retrasan a los escaladores más veteranos. La fila de escaladores que comienza en la base de la montaña siendo única termina, debido a esos atascos, dividiéndose en distintas agrupaciones que suben a la velocidad del primer escalador que origina el atasco.

Entrada

La entrada está compuesta por distintos casos de prueba, cada uno describiendo, en dos líneas, una hilera de escaladores.

La primera línea tiene un único número indicando la cantidad total de escaladores que ascienden por la ruta ese día (hasta 10.000). En la segunda, aparece la velocidad máxima a la que podría ascender cada uno de esos escaladores; el primer número indica la velocidad de la persona que va en cabeza. Todas las velocidades son enteros positivos y ninguna supera 10^9.

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

Salida

Por cada caso de prueba se escribirá una línea con tres números, indicando el número de grupos distintos que se hacen en la fila debido a los atascos y el tamaño del grupo más pequeño y del grupo más grande.

Entrada de ejemplo

```
6
10 11 4 5 6 1
7
10 15 18 2 4 6 20
0
```

Salida de ejemplo

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

Contar hasta el final

El día que, con cuatro años, Clara averiguó que los números son infinitos no le gustó nada. Le había costado mucho tiempo aprender a contar hasta 100, estaba empezando a entender cómo llegar hasta 1.000, y de repente se había enterado de que siempre tendría más y más números que aprender.

Le gustó tan poco ese descubrimiento que no pudo aceptarlo. A cualquier adulto que veía le preguntaba, sin previo aviso, por qué los números nunca se acaban. Aunque la respuesta era invariablemente la misma (un inquietante "porque siempre se puede sumar uno más a cualquier número"), ella no perdía la esperanza de que finalmente se rindieran y confesaran que todo había sido una broma de mal gusto.

Un año después, en clase de religión, encontró la incrongruencia que estaba buscando. Esta tarde, en cuanto llegara a su casa pillaría a su padre en un renuncio y le obligaría a reconocer que, efectivamente, había un último número. La pregunta que le haría resonaba en su mente...

— Papá, ¿Dios puede contar hasta el final?

Entrada

La entrada está compuesta por distintos casos de prueba cada uno en una línea.

Cada línea, de como mucho 1.000 caracteres, contiene un número entero utilizando el punto para separar los millares.

Salida

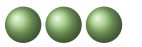
Por cada caso de prueba se escribirá el resultado de sumar uno al número leído usando el mismo formato que en la entrada.

Entrada de ejemplo

```
1
9
1.345
1.234.567
```

Salida de ejemplo

```
2
10
1.346
1.234.568
```



Camellos, serpientes y kebabs

GoodCode
is_like_a
GoodJoke
it-needs-no-explanation

Lador, el "compi" con quien hago las prácticas de programación, me dice que hay que poner nombres de variables significativos, que ayuden a entender qué guarda cada una. Siempre me critica si uso identificadores como `i`, `otra` o `aux`.

Para darle un escarmiento, intenté poner una variable que se llamaba `suma de los impares menores que n`. Pero el compilador me gritó cosas muy feas que no entendí. Cuando preguntamos a la profesora nos dijo que no se podían poner espacios en los nombres. Si queríamos poner nombres con varias palabras (aunque nos dijo que no pusiéramos tantas) entonces teníamos que usar algún truco para que el nombre se leyera bien sin los espacios.

Nos contó que hay varias formas, y se usa una u otra dependiendo de las preferencias personales, o del convenio usado en el lenguaje. Y nos soltó un sermón sobre las mayúsculas del camello, serpientes y kebabs que nos dejaron muy confundidos. Buscando luego en Internet vimos que hay principalmente tres opciones:

- *CamelCase*: la primera letra de cada palabra se escribe en mayúscula. Hay dos alternativas, *UpperCamelCase* y *lowerCamelCase* dependiendo de si la primera letra de la primera palabra va también en mayúscula o no. ¡Esta es la que más se utiliza!

- *snake_case*: todas las letras van en minúscula, y las palabras se separan por un guión bajo (_). Se utilizaba en C, y todavía sobrevive en algunos sitios.

- *kebab-case*: como antes, todas las letras van en minúscula, pero ahora las palabras se separan por un guión medio (-). En muchos lenguajes de programación no se puede usar, porque el `-` se confunde con el signo de la resta y no es válido. Se usa en Lisp (el lenguaje de los paréntesis).

Lo peor de todo fue que a Lador, después de que la profesora nos contara todo esto, se le ocurrió la feliz idea de preguntarle cuántos espacios utilizar para sangrar el código, y en qué línea colocar las llaves. Al final, terminamos perdiendo el autobús.

Entrada

Cada caso de prueba es un nombre de variable en alguna de las tres notaciones anteriores seguida de la notación a la que se quiere convertir (`CamelCase`, `snake_case` o `kebab-case`).

Ningún nombre de variable tendrá más de 20 caracteres y se garantiza que será correcta en alguna de las notaciones.

Salida

Para cada caso de prueba se escribirá el nombre de la variable en la notación solicitada. Tanto en la entrada como en la salida se utilizará *UpperCamelCase* (y no *lowerCamelCase*).

Entrada de ejemplo

```
MiVar snake_case
es_primo kebab-case
suma-de-impares CamelCase
j CamelCase
```

Salida de ejemplo

```
mi_var
es-primo
SumaDeImpares
J
```

K
Abanico de naipes

Mi hermano pequeño ha encontrado una baraja de cartas y está haciendo como que juega con ellas con sus amigos. Como tiene las manos pequeñas y poca práctica, le cuesta mantener en una mano una serie de cartas colocadas en forma de abanico como el de la imagen.

Cuando nos ha visto jugar se ha fijado que al coger las cartas primero las colocamos en abanico y luego las ordenamos de alguna manera, haciendo parejas o tríos, o poniendo juntas las del mismo palo. Para evitar que se le caigan todas al suelo, mi hermano ordena su abanico de cartas realizando movimientos solamente de este estilo: elige una carta, la saca del abanico y la coloca a la derecha del todo, encima de las demás.

Para practicar esta forma de ordenar, ha numerado las cartas, ha elegido al azar un conjunto de ellas, las ha colocado en forma de abanico y ha empezado a ordenarlas de manera que los números en las cartas queden ordenados de menor a mayor. Lleva un buen rato y se está empezando a poner nervioso, porque cuando ya tiene unas pocas ordenadas hace un movimiento más y se le vuelven a descolocar algunas.

Dada una configuración inicial de cartas, ¿cuál es el mínimo número de movimientos (como los que hace mi hermano, sacar una carta del abanico y colocarla a la derecha) que hay que realizar para ordenarlas de menor a mayor? Para ponertelo difícil, imaginaremos abanicos desmesuradamente grandes.

Entrada

La entrada está formada por una serie de casos de prueba. Cada caso ocupa dos líneas. En la primera apacere un número entre 1 y 100.000 que representa el número de cartas en un abanico. En la segunda línea aparecen los números escritos en cada una de esas cartas (números entre 1 y 1.000.000, todos distintos); el primer número corresponde a la carta más a la izquierda, debajo de todas, y el último a la carta más a la derecha, la que inicialmente está encima de todas.

La entrada termina con una línea con un cero, que no deberá ser procesada.

Salida

Para cada caso de prueba el programa escribirá el mínimo número de movimientos que hay que hacer para ordenar de menor a mayor los números en las cartas.

Entrada de ejemplo

```
4
1 2 4 3
4
9 7 5 3
3
8 9 10
0
```

Salida de ejemplo

```
1
3
0
```

L
La digestión de las serpientes

Por todos es sabido que las serpientes son animales de "sangre fría", que significa que no son capaces de regular su temperatura corporal y que por tanto dependen de la temperatura ambiental para sobrevivir[1].

Para incrementar su temperatura interna suelen tomar el sol en rocas u otros tipos de suelos que transmitan bien el calor. De esta forma consiguen subir su temperatura en un corto espacio de tiempo y pueden volver a lugares seguros donde son menos visibles.

Aunque son capaces de sobrevivir con temperaturas bajas, en ciertos momentos les resulta imprescindible mantener una temperatura corporal elevada. Uno de esos momentos es durante la caza: la temperatura elevada les proporciona la energía necesaria para la actividad. Otro momento importante es inmediatamente después, durante la digestión. En caso de que sus cuerpos sufrieran un enfriamiento, la presa engullida podría pudrirse y terminar matándolas "desde dentro".

Debido a esto, las serpientes deben planificar muy bien cuándo salir de caza. Necesitan buscar el momento del día en el que la temperatura ambiente se mantiene por encima de un umbral por más tiempo; eso les garantiza que tendrán tiempo suficiente para calentarse, buscar la presa, cazarla y digerirla.

Como miembros del equipo de apoyo de los biólogos australianos que estudian la fauna local, debemos comprobar si las serpientes escogen el momento de caza de forma óptima o no. Nuestra primera tarea es analizar la secuencia de temperatura ambiente de un día y encontrar la duración del periodo más largo en la que esa temperatura se mantiene lo suficientemente elevada. Eso sí, las serpientes tienen cierta tolerancia y admiten pequeños intervalos de tiempo con temperaturas por debajo del umbral deseado.

Entrada

La entrada estará compuesta por distintos casos de prueba, cada uno ocupando dos líneas.

La primera línea de cada caso de prueba tendrá dos enteros, el primero indicando el número de mediciones realizadas ($1 \leq n \leq 100.000$), y el segundo el número k de mediciones consecutivas por debajo del umbral que una serpiente es capaz de soportar entre dos mediciones con temperaturas altas.

La segunda línea contiene n números que serán siempre ceros o unos. Un cero indica una temperatura demasiado baja para la serpiente para calentarse, mientras que un uno indica una temperatura aceptable.

La entrada termina con una línea con dos ceros, que no deberá procesarse.

Salida

Por cada caso de prueba se escribirá la longitud del periodo más largo (medido en número de muestras) que tiene la serpiente para comer.

Un periodo es válido si comienza y termina por unos y entre dos lecturas con temperaturas por encima del umbral no hay más de k medidas consecutivas con temperaturas por debajo.

Entrada de ejemplo

```
6 0
0 1 0 1 0 0
6 0
0 1 0 1 1 0
8 1
1 0 0 1 0 1 0 1
0 0
```

[1] En realidad en un contexto científico, el término "sangre fría" y "sangre caliente" ya no se utiliza directamente, pues se ha visto que los tipos de temperatura corporal no son tan simples como para utilizar únicamente esas dos categorías.

Salida de ejemplo

```
1
2
5
```