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

Programa-Me 2022
Calentamiento pre-navideño
Problemas

Concurso gestionado desde

IES Serra Perensixa, Torrent

Concurso on-line sobre **¡Acepta el reto!**
`https://www.aceptaelreto.com`
16 de diciembre de 2021

Universidad Complutense Madrid

16 de diciembre de 2021
http://www.programame.com

Listado de problemas

Los problemas están ambientados en videojuegos de las décadas de los 70 y 80, y en los ordenadores de 8 bits como recuerdo a Sir Clive Sinclair, creador del ZX Spectrum, fallecido en septiembre de 2021. Las imágenes de los enunciados son capturas de juegos de la época, que son propiedad de sus respectivos dueños. Las historias de las ambientaciones son reales.

Autores de los problemas:

- Sergi García Barea (IES Serra Perenxisa, Torrent)
- Pedro Pablo Gómez Martín (Universidad Complutense de Madrid)

Revisores de los problemas:

- Marco Antonio Gómez Martín (Universidad Complutense de Madrid)
- Alberto Verdejo López (Universidad Complutense de Madrid)

A
La pulga

"La pulga" es considerado el primer videojuego desarrollado en España. Creado por Paco Suárez y Paco Portalo, fue publicado en 1983 para ZX Spectrum y portado a otros ordenadores de la época, como MSX, Amstrad CPC y Commodore 64. Se exportó a muchos países con distintos nombres ("Bugaboo", "Roland in the Caves", "Il Paese Incantato", ...) suponiendo el inicio de la llamada edad de oro del software español.

El juego comienza con una de las primeras "cinemáticas" (*cut scenes*) de la historia. Muestra a la pulga protagonista, *Bugaboo*, tripulando su sonda Cebolla X7 y estrellándose en un planeta del sector Almak-1 donde había detectado vida. Termina en una gruta de la que debe escapar saltando hasta volver a la superficie.

Una de las mecánicas más novedosas fue el control. En los juegos anteriores las teclas se transformaban en acciones inmediatas. En "La pulga", sin embargo, la protagonista salta con mayor o menor intensidad dependiendo del tiempo que el jugador haya mantenido pulsada la tecla de salto. En la pantalla se muestra una barra que indica la potencia y que crece rápidamente mientras la tecla se mantiene pulsada. Dependiendo del momento en el que se suelte, la pulga saltará más o menos. Si se pasa un tiempo máximo, la barra vuelve a la posición inicial (potencia 0) y empieza a crecer de nuevo.

Esta novedosa mecánica afectaba de forma decisiva a la jugabilidad. Cuando el jugador necesitaba huir rápidamente del único enemigo del juego, querría hacerlo saltando mucho pero eso requería mantener la tecla pulsada más tiempo, aumentando el peligro. Es de suponer que los desarrolladores hicieron muchas pruebas hasta afinar la mecánica y los tiempos para conseguir una buena jugabilidad.

Entrada

La entrada comienza con un número indicando cuántos casos de prueba deberán ser procesados. Cada caso de prueba está compuesto por tres números, $1 \leq n \leq 100$, $1 \leq f \leq 100$ y $1 \leq t \leq 100.000$. El primero indica el número de pasos que tiene la *barra de potencia*. El segundo indica el *factor* por el que se multiplica la potencia indicada por la barra antes de aplicar el salto. Por último, t indica el tiempo, en centésimas de segundo, que se ha mantenido pulsada la tecla de salto.

Al principio, la barra está a 0 y se incrementa de uno en uno por cada centésima de segundo que la tecla se mantenga pulsada. Cuando se llena completamente, se pone a 0 y repite el ciclo.

Salida

Por cada caso de prueba el programa escribirá la potencia de salto que hay que aplicar a la pulga.

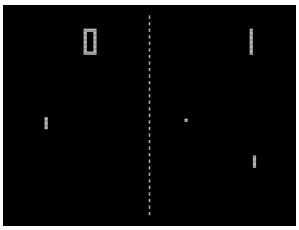
Entrada de ejemplo

```
3
32 1 32
32 1 33
40 10 46
```

Salida de ejemplo

```
32
0
50
```

B
Pic, poc, pic... pong!



En septiembre de 1972 veía la luz la *Magnavox Oddysey*, la primera videoconsola de la historia, tras el esfuerzo de Ralph Baer para convertir su *caja marrón* en un producto comercial. A medio camino entre consola y juego de mesa (incluía una baraja de cartas, fichas de póker, una ruleta y láminas para adherir a la televisión), su éxito puso en marcha la industria del videojuego.

Solo dos meses después, Atari sacó su conocida *máquina de arcade* "Pong". Era un clon de uno de los juegos de la Odyssey, lo que desencadenó una batalla legal entre Magnavox y Atari.

Cada jugador controlaba una "raqueta" situada en extremos contrarios de la pantalla. Una bola se desplazaba de lado a lado y debía ser golpeada por el jugador correspondiente para evitar que superara el límite de la pantalla. Al ser golpeada, la pelota rebotaba en dirección opuesta, retando al jugador contrario. Si la bola impactaba con el borde superior o inferior de la pantalla, rebotaba manteniendo el sentido. Si un jugador fallaba al golpear la bola, esta superaba el límite de la pantalla y el contrincante conseguía un punto a favor. Al reaparecer, la bola se dirigía de nuevo hacia él.

Al empezar una partida, la bola aparecía en el centro de la pantalla y se desplazaba hacia la derecha, de modo que el jugador situado en ese lado era siempre el primero en golpear.

Pong incorporaba sonido y marcador de puntos, algo de lo que carecía la versión original de Odyssey. Tenía tres "beeps" característicos distintos, que sonaban cuando la pelota golpeaba una raqueta, cuando golpeaba el borde superior o inferior, o cuando un jugador no golpeaba la bola y el contrincante se llevaba un punto. Dados *los sonidos* reproducidos por el juego, ¿eres capaz de reconstruir el marcador?

Entrada

Cada línea de la entrada se corresponde con un caso de prueba que el programa deberá procesar.

Cada caso de prueba comienza con un número indicando el número de sonidos reproducidos en la partida en curso. La *onomatopeya* para el sonido de la bola golpeando una raqueta es "`PIC`". El sonido de la bola golpeando el borde superior e inferior es "`POC`". Por último, cuando la bola supera un jugador y su oponente gana un punto suena "`PONG!`". Se garantiza que todos los casos terminan con un "`PONG!`".

El programa debe terminar al leer un `0` (partida sin sonidos).

Salida

Por cada caso de prueba el programa escribirá el marcador, indicando primero el número de puntos ganados por el jugador de la izquierda y luego el de la derecha, separados por un espacio. En cada partida, la bola siempre comienza desplazándose hacia la derecha. Cuando alguien pierde un punto, la bola comienza yendo de nuevo hacia él, como si el contrario hubiera "sacado".

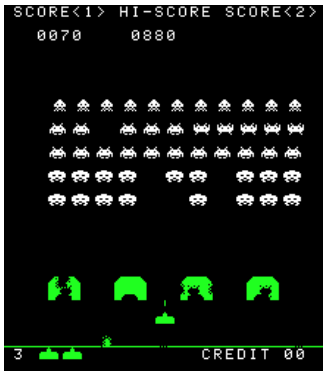
Entrada de ejemplo

```
4 PIC POC PIC PONG!
9 PIC POC PONG! PONG! PIC PIC PIC POC PONG!
0
```

Salida de ejemplo

```
1 0
1 2
```

C
Mensajes en Space Invaders



Space Invaders fue uno de los primeros *shooters* de la historia. Publicado en 1978 como máquina de arcade, cuenta la leyenda que en Japón ocasionó escasez de monedas de 100 yenes debido a su enorme éxito.

Su creación requirió el diseño de *hardware* específico para soportar las necesidades computacionales y gráficas del juego. Además de la propuesta jugable, una de las novedades que incorporó fue la tabla de mejores puntuaciones, en la que los jugadores podían registrar su nombre si conseguían una de las mejores puntuaciones del juego.

Debido a las restricciones del *hardware* de la época, por "nombre" se entendía únicamente a *tres letras*, de modo que los jugadores tenían que conformarse con poner sus apodos o sus siglas.

Solo los jugadores más experimentados podían dejar mensajes más largos si conseguían todas las mejores puntuaciones de la máquina. Para eso, elegían las tres letras de cada mejor puntuación de modo que si se leían *todas en vertical* pudiera leerse algo significativo.

Entrada

El programa leerá, de la entrada estándar, un primer número que indica cuántos casos de prueba deberá procesar.

Cada caso de prueba comienza con un número indicando cuántas mejores puntuaciones almacena una determinada máquina de arcade (al menos 1 y no más de 20). A continuación aparecen, en líneas separadas, los nombres asociados a cada una de esas puntuaciones, que estarán compuestos por exactamente tres letras del alfabeto inglés o símbolos de puntuación sin el espacio.

Salida

Por cada caso de prueba el programa escribirá el mensaje que se puede ver en la tabla de *records* si se leen las letras de arriba hacia abajo, empezando por la primera letra de cada entrada.

Entrada de ejemplo

```
1
5
S.D
PIE
ANR
CVS
EA!
```

Salida de ejemplo

```
SPACE.INVADERS!
```

D
Buscando el nivel

Los ordenadores domésticos de los 80 guardaban los datos en *cintas de cassette*, que son un soporte *secuencial*. El ordenador no podía elegir qué datos leer, sino que el usuario pulsaba el botón *Play*, la cinta comenzaba a girar, y si los datos que pasaban por debajo del lector eran los esperados, el ordenador los leía y, si no, los descartaba y seguía esperando a que llegaran los correctos. Como la búsqueda de los datos era manual, los lectores solían disponer de un contador, que avanzaba con la cinta. Al rebobinarla, el usuario podía usar el contador para saber dónde parar, siempre que hubiera anotado con antelación las posiciones de cada cosa.

Normalmente los programas tenían todos los datos seguidos en la cinta y no había que hacer nada especial. Pero algunos juegos necesitaban más información de la que entraba en memoria y, al superar un nivel, había que pulsar *Play* y esperar a que se cargara el siguiente. Al acabar la partida había que rebobinar la cinta hasta la posición donde estaba guardado el nivel 1 para que el programa lo leyera y se pudiera jugar otra vez. En juegos como "Out Run", donde al acabar un nivel se podía ir a dos distintos, era necesario buscar también al avanzar y no solo al acabar la partida.

Entrada

Cada caso de prueba comienza con un número $1 \leq n \leq 100$ indicando la cantidad de niveles que tiene un determinado juego. A continuación aparecen n números indicando lo que ocupa en la cinta la información de cada uno de esos niveles. Los niveles aparecen guardados uno detrás de otro en orden.

El caso de prueba continúa con un número $1 \leq q \leq 10.000$ indicando cuántas subconsultas se harán sobre los datos anteriores. Cada una está compuesta por dos números $1 \leq a,b \leq n$, indicando el último nivel que se ha cargado (la cinta está colocada al final de sus datos) y el siguiente que se tiene que cargar.

La entrada termina con un caso de un juego sin niveles, que no deberá procesarse.

Salida

Por cada subconsulta, el programa escribirá el desplazamiento que hay que realizar sobre la cinta para colocarla al principio de los datos del nuevo nivel a cargar (un número positivo significará avanzar y uno negativo retroceder). Después de cada caso de prueba se escribirán tres guiones (`---`).

Entrada de ejemplo

```
5 10 20 40 7 3
3
1 2
2 5
4 1
0
```

Salida de ejemplo

```
0
47
-77
---
```

E
Pantallas de carga

Durante los 80, las portadas de los juegos tenían que vender el juego en una época en la que el realismo gráfico era inexistente. Los estudios encargaban las portadas a ilustradores como Alfonso Azpiri o Luis Royo que crearon verdaderas obras de arte.

El problema era cuando el juego tenía que mostrar esas imágenes durante las pantallas de carga para aliviar la tediosa espera. Además de tener muy bajas resoluciones, la riqueza de color era mínima.

El ZX Spectrum, por ejemplo, solo era capaz de mostrar 15 colores[1], y no todos a la vez. En concreto, la pantalla se dividía en bloques de 8×8 píxeles, y en cada bloque solo podían usarse dos colores distintos. De esta forma, con su resolución de 256×192 la pantalla quedaba dividida en una cuadrícula de 32×24 celdas de 8×8 píxeles, y cada una solo podía mostrar 2 de los 15 colores.

Conseguir disimular esta limitación era también todo un arte.

Entrada

Cada caso de prueba comienza con dos números que indican el ancho y alto, en número de píxeles, de una imagen. Se garantiza que ambas dimensiones son múltiplo de 8 y no serán mayores que 48.

A continuación viene la imagen, con una letra para cada píxel entre la `A` y la `O` (con el alfabeto inglés, 15 letras distintas).

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

Salida

Por cada caso de prueba el programa escribirá "`SI`" si la imagen puede representarse en un ZX Spectrum, es decir si cada uno de los bloques de 8×8 píxeles utiliza como mucho 2 colores (letras). Si, por el contrario, algún bloque de 8×8 usa más de dos colores se escribirá "`NO`".

Entrada de ejemplo

```
16 8
AAAAAAAIBBCCCBBC
AAIIIAAICBBCBBCC
AIIIAAIICCBBBCCC
IIIAAIIICCBBBCCC
IIAAIIAICBBCBBCC
IAAIIAAIBBCCCBBC
AAAAAAAIBBCCCBBC
IIIIIIIICCCCCCCC
0 0
```

Salida de ejemplo

```
SI
```

[1] Cada canal rojo, verde y azul podía estar activado o desactivado, lo que da 8 opciones distintas. Luego había dos versiones de cada uno, la "básica" y la "brillante". Ambas versiones eran igual para el negro, de ahí los 15 colores en lugar de 16.

F
Juegos en cintas

La mayoría de los ordenadores de 8 bits utilizaban *cintas de cassette* como almacenamiento de sus programas y juegos. Las cintas, usadas como soporte de audio para música, guardaban la información binaria a través de sonidos. Su capacidad se medía en tiempo (normalmente minutos) y tenían *dos caras* de la misma capacidad. Se utilizaba una u otra dependiendo de cómo se insertara la cinta en la unidad lectora. Las cintas eran lentas, de acceso serie (tenían que rebobinarse para colocarlas en el punto donde comenzara el programa que se quería leer) y con el uso se estropeaban y fallaban.

Como los juegos se guardaban, literalmente, como audio, era fácil copiarlos de una cinta a otra con aparatos de *doble pletina*.

Bárbara Surada es aficionada a los juegos antiguos de Spectrum y Amstrad pero teme que las cintas de los juegos originales se le estropeen y los pierda. Quiere hacer copias en cintas vírgenes como copia de seguridad. Para no tener que andar rebobinando mucho para encontrar el juego que quiere, como mucho meterá 8 juegos en cada cinta (entre las dos caras). Lo que no sabe es si, dado lo que dura cada juego, podrá o no meterlos en la misma cinta.

Entrada

El programa deberá leer múltiples casos de prueba de la entrada estándar.

Cada caso de prueba comienza con una línea con dos números. El primero indica el tamaño de cada una de las dos caras de la cinta virgen donde se quiere hacer la copia de los juegos (entre 1 y 10^8). El segundo indica el número de juegos que se quieren copiar (entre 1 y 8).

A continuación aparece una segunda línea con la duración de cada uno de los juegos (entre 1 y 10^8).

Salida

Por cada caso de prueba el programa escribirá "`SI`" si es posible grabar los juegos en la cinta, y "`NO`" en otro caso. Los juegos *no* pueden cortarse, de modo que cada uno debe estar completo en una única cara.

No es necesario dejar separación entre dos juegos consecutivos en la cinta.

Entrada de ejemplo

```
10 4
7 7 3 3
10 3
4 7 7
```

Salida de ejemplo

```
SI
NO
```