Pregunta **1**

Sin finalizar

Puntúa como 1,00

BANNER

Hacer un banner según Ejercicio U4_B4A_E7

Letras necesarias: A,E,I,C,D,M,S,T en matrices de 5x5. Caracter de impresión #.

El formato concreto de cada letra obsérvalo en los ejemplos

Por ejemplo:

Entrada	Resultado
DAM	## #### # # # # # # ## ## # # #### # # # # # # # # # ## # # # #

TELMAAAA	##### #### # # # #### #### #### ####
	  # # # ## ## # # # # # # #
	  # ### # # # # #### #### #### ####
	  # # # # # # # # # # # #
	  # #### #### # # # # # # # # #

MADA DE CIMOS	# # #### ## #### ## #### #### # # # ##### #### ## ## # # # # # # # # # # # ## ## # # # # # # #### # # #### # # ### # # # # # # # #### # # # # # # # # # # # # # # # # # # # # # ## # # ## #### #### # # # ### # ####

Respuesta: (sistema de penalización: 0 %)

```
1 |
```

Comprobar

Pregunta **2**

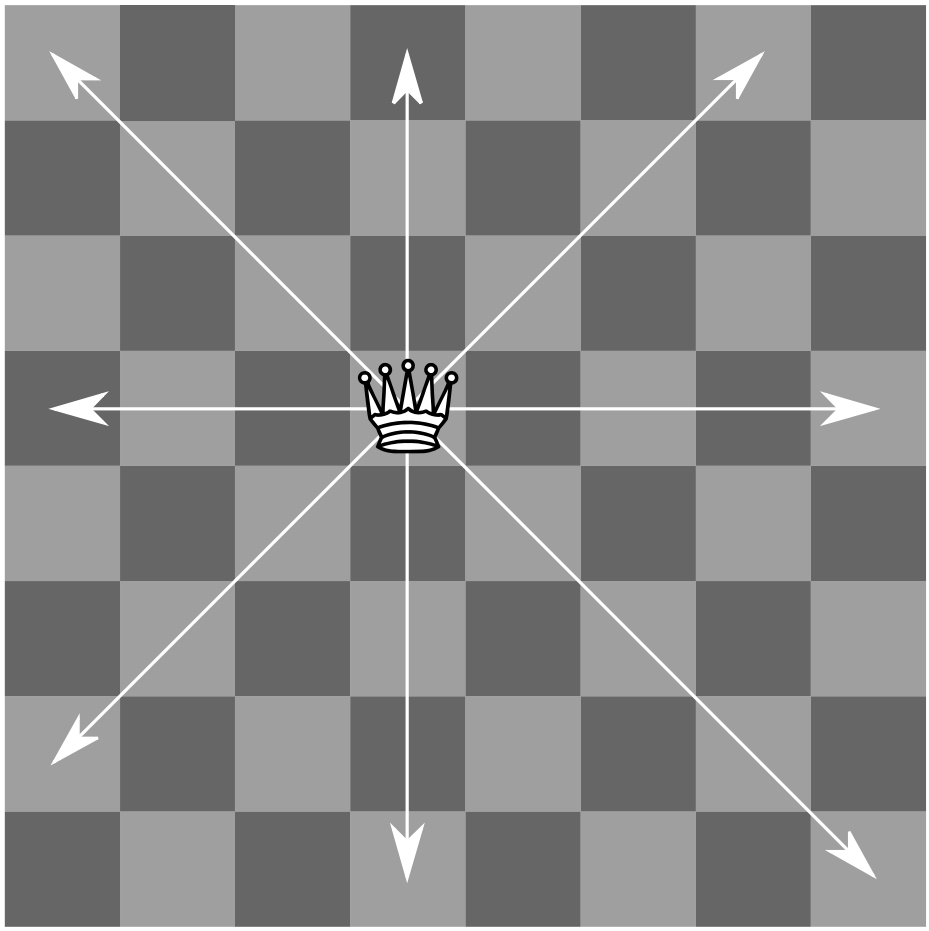
Sin finalizar

Puntúa como 1,00

REINAS ATACADAS

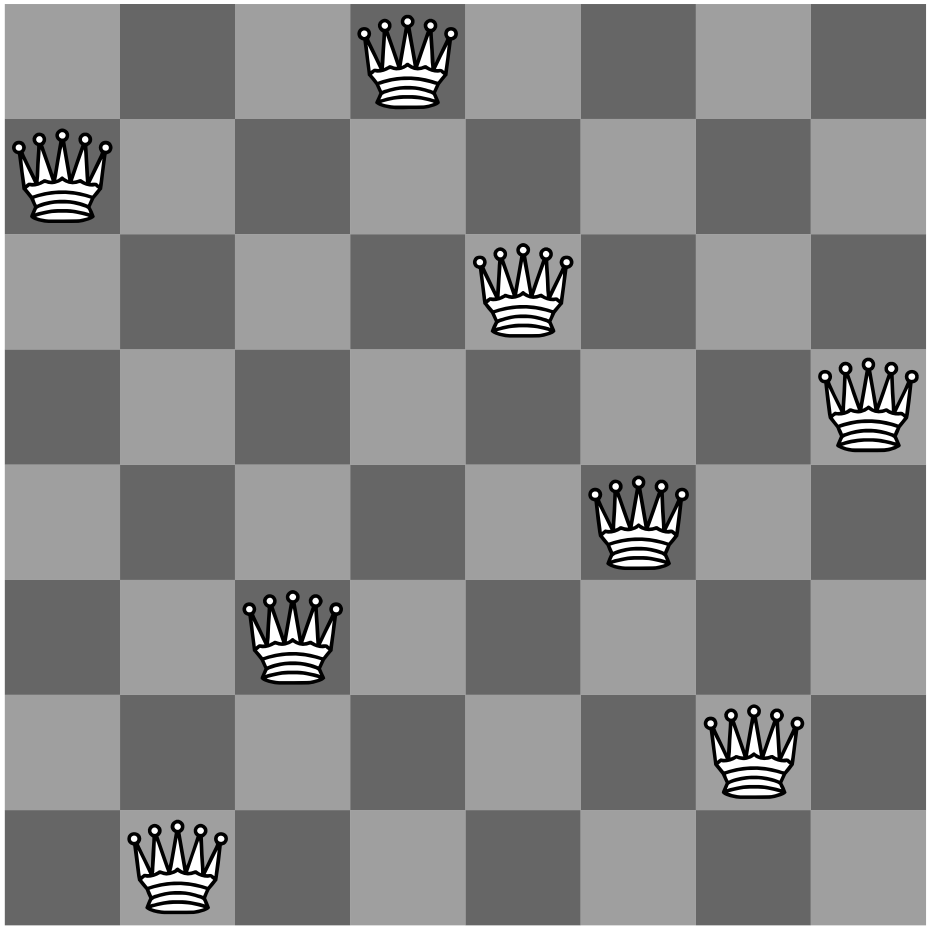
https://www.aceptaelreto.com/problem/statement.php?id=244

En el ajedrez, la reina es la pieza más poderosa, al poderse mover cualquier número de escaques en vertical, horizontal, o diagonal.



Movimientos de la reina

En 1848, el alemán Max Bezzel planteó el *puzzle de las 8 reinas*, en el que retó a colocar 8 reinas sobre un tablero sin que se atacaran entre sí. Dos años después, se dieron algunas de las 92 soluciones.



Una de las soluciones posibles

Desde entonces, matemáticos y aficionados de todo el mundo han estudiado el problema, generalizándolo a tamaños de tableros de ajedrez de N×N. En 1972, Dijkstra, en plena *crisis del software*, usó el problema para demostrar el poder de la programación estructurada, y desde entonces es un ejemplo clásico de algoritmo *de vuelta atrás*.

Para poder colocar las reinas, el primer paso es saber cuándo un grupo de reinas sobre un tablero de ajedrez se atacan entre sí, es decir cuándo hay al menos una reina que podría *comer* a otra siguiendo las reglas del movimiento del juego.

Entrada

La entrada consta de un conjunto de casos de prueba. Cada uno comienza con una línea con dos números. El primero indica el ancho y alto del tablero de ajedrez (siempre será cuadrado de como mucho 2.000×2.000). El segundo indica el número de reinas colocadas sobre él (entre 1 y 100).

A continuación vendrá una línea con la posición de todas las reinas. Para cada una, se indicará primero la coordenada X y luego la Y, separadas por espacio. Las posiciones de cada reina también se separarán por un único espacio. Todas las posiciones serán válidas (cada coordenada estará entre 1 y el tamaño del tablero) y se garantiza que no habrá dos reinas en la misma posición.

La entrada termina con un caso de prueba con un tablero de tamaño 0×0 y sin reinas que no debe procesarse.

Salida

Para cada caso de prueba, el programa escribirá, en la salida estándar, una línea con el texto "`SI`" si hay reinas atacadas en la configuración dada, y "`NO`" en otro caso (sin las comillas).

Entrada de ejemplo

Salida de ejemplo

```
NO
SI
NO
```

Por ejemplo:

Entrada	Resultado

Entrada	Resultado
8 8 1 2 2 8 3 6 4 1 5 3 6 5 7 7 8 4 4 2 1 1 3 3 4 2 1 1 3 2 0 0	NO SI NO

Respuesta: (sistema de penalización: 0 %)

```
1
```

Comprobar

Pregunta **3**

Sin finalizar

Puntúa como 1,00

Copistas daltónicos

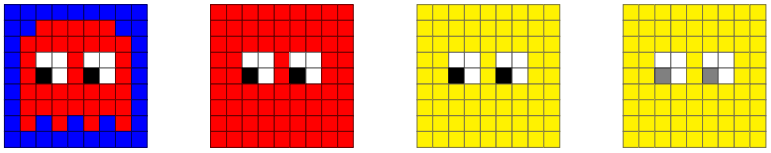
https://www.aceptaelreto.com/problem/statement.php?id=266

El daltonismo es un transtorno hereditario que ocasiona dificultad para distinguir ciertos colores. Hay distintos tipos de daltonismo que hacen que los colores que esas personas no distinguen varíen. Lo habitual, no obstante, es no distinguir algunos matices de verde y rojo.

Aunque el defecto genético no suele suponer ningún problema en la vida diaria de los afectados, la realidad es que les imposibilita para realizar algunos trabajos puntuales, como militares de carrera o pilotos.

Otro de los trabajos difíciles de realizar por los daltónicos es el de duplicador de obras de arte. El problema al hacer la copia de la obra es que esos dos colores que cualquier otro vería distintos terminan siendo el mismo en la copia. Si esa copia cae en las manos de un segundo duplicador daltónico que hace una nueva copia, y luego otro, y luego otro, el resultado final puede no parecerse en nada al original, sobre todo si el tipo de daltonismo de cada uno difiere1.

Como ejemplo, en la figura aparece la transformación que puede sufrir un cuadro de *pixel art* con uno de los personajes del Pacman tras el paso por varios copistas daltónicos. Los colores originales son azul, rojo, blanco y negro. En la primera reproducción el copista sufría un tipo de daltonismo que le hacía ver igual los colores azul y rojo lo que hace que todos los azules terminen siendo rojos2. En la segunda reproducción el copista convirtió todos los rojos en amarillo. El último de la serie veía todos los negros grises.



Lo que haremos será, precisamente, simular esa transformación de la obra original tras pasar por las manos de numerosos copistas daltónicos.

Entrada

La entrada estará compuesta por distintos casos de prueba. Cada caso de prueba comienza con la descripción de un cuadro que será copiado en serie por distintos daltónicos.

Los cuadros se representarán mediante letras mayúsculas, cada uno representando un color. Para eso una primera línea contendrá el tamaño del cuadro: dos números entre 1 y 500 indicando el número de filas y el número de columnas respectivamente, a lo que seguirá el cuadro. Acto seguido aparecerá una línea con el número de daltónicos que copiarán el cuadro. Por último por cada uno de los copistas aparecerá una línea con dos caracteres; el primero indica el código del color que no es capaz de distinguir y que es sustituido por el código del color marcado por el segundo carácter.

La salida terminará con un cuadro de tamaño 0×0 que no debe procesarse.

Salida

Para cada caso de prueba se escribirá el cuadro tal y como lo dibuja el último copista utilizando la misma representación usada en la entrada.

Por ejemplo:

Entrada	Resultado
1 4 ABCD 1 C D 9 9 AAAAAAAAA AARRRRRAA ARRRRRRRA ARBBRBBRA ARNBRNBRA ARRRRRRRA ARRRRRRRA ARARARARA AAAAAAAAA 3 A R R Y N G 0 0	ABDD YYYYYYYYY YYYYYYYYY YYYYYYYYY YYBBYBBYY YYGBYGBYY YYYYYYYYY YYYYYYYYY YYYYYYYYY YYYYYYYYY

Respuesta: (sistema de penalización: 0 %)

```
1
```

Comprobar

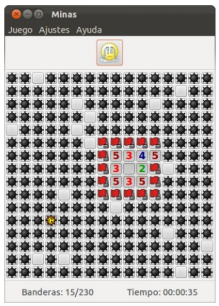
Pregunta **4**

Sin finalizar

Puntúa como 1,00

Campo de minas

https://www.aceptaelreto.com/problem/statement.php?id=176



Aunque existieron antecedentes previos, el juego *Buscaminas* comenzó a ser famoso cuando se incluyó en la versión 3.1 de Windows, en el lejano 1992. Corre el rumor de que fue incluído para que los usuarios se entretuvieran mientras, sin saberlo, cogían práctica utilizando un dispositivo por aquel tiempo apenas conocido: el ratón.

Hoy el juego es archiconocido (y el dispositivo también). El número de variantes e implementaciones es inmenso, y sigue estando incluído en la gran mayoría de los sistemas de escritorio.

El juego consiste en un tablero rectangular con celdas, algunas de las cuales ocultan *minas*. El usuario debe ir destapando las celdas con cuidado para no seleccionar ninguna mina. Como ayuda, cada vez que destapa una celda libre, se le muestra cuántas minas tiene alrededor.

En este problema os daremos la configuración de un tablero de *Buscaminas* totalmente descubierto, y os pedimos que nos digáis cuántas celdas vacías tienen al menos 6 minas a su alrededor.

Entrada

La entrada estará compuesta de múltiples casos de prueba. Cada uno comienza con una línea conteniendo dos números enteros positivos, menores que 1.000, que indican, respectivamente, el ancho y el alto del tablero. A continuación vendrá una línea por cada fila del tablero. Cada celda se representa con un * indicando que en esa celda hay una mina, o con un - indicando que está libre.

La entrada termina con un tablero de ancho o alto 0.

Salida

Para cada caso de prueba se debe indicar, en una única línea, el número de celdas vacías con al menos 6 minas alrededor.

Por ejemplo:

Entrada	Resultado
5 4 *--** ---*- ---** ***-- 4 5 **-* **** *--* **-* -*** 0 0	0 3

Respuesta: (sistema de penalización: 0 %)

```
1
```

Comprobar

Pregunta **5**

Sin finalizar

Puntúa como 1,00

Sombras en el camping

https://www.aceptaelreto.com/problem/statement.php?id=207

Tiempo máximo: 2,000-4,000 s

Se acerca el verano y los aficionados a la naturaleza pasarán buena parte de él en campings, disfrutando del aire libre.

Un requisito imprescindible en los meses de calor es colocar la tienda de campaña bajo la sombra de un buen árbol para poder pasar frescos las horas de siesta. Pero, dependiendo de la zona, eso no siempre es fácil. En los campings nuevos, el número de árboles es escaso, y también lo es por tanto el número de parcelas aptas para tiendas.

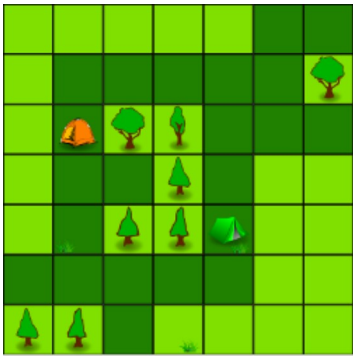


Figura 1: primer ejemplo de entrada destacando las zonas de sombra

Sabiendo que cada árbol proporciona sombra a las ocho parcelas adyacentes, ¿cuántas tiendas de campaña disfrutarán de sombra en un *camping*?

Entrada

El programa deberá procesar múltiples casos de prueba recibidos por la entrada estándar. Cada uno representa un *camping* formado por una cuadrícula de parcelas de igual tamaño en los que puede haber hueco para una tienda, o un árbol.

Cada caso de prueba comienza con dos números $1 \leq c, f \leq 50$, indicando el número de columnas y de filas de la cuadrícula de parcelas. A continuación se indica el número a de árboles del camping.

Si hay árboles, en la siguiente línea aparece la posición de todos ellos, indicando para cada uno la columna (1...*c*) y la fila (1...*f*) que ocupan. En total, aparecerán $2 \times a$ números.

La entrada termina con una línea con tres ceros (camping con dimensiones nulas y sin árboles), que no debe procesarse.

Salida

Por cada caso de prueba el programa indicará, en una línea, el número de parcelas que disfrutarán de sombra.

Por ejemplo:

Entrada	Resultado
7 7 8 7 2 3 3 4 3 4 4 3 5 4 5 1 7 2 7 5 3 1 3 2 0 0 0	22 8

Respuesta: (sistema de penalización: 0 %)

```
1
```

Comprobar

Pregunta **6**

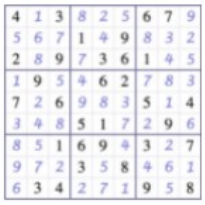
Sin finalizar

Puntúa como 1,00

Sudokus correctos

https://www.aceptaelreto.com/problem/statement.php?id=345

El sudoku es un pasatiempo lógico que consiste en rellenar una cuadrícula de 9×9 casillas dividida en nueve regiones 3×3 (las separadas con líneas más gruesas en la imagen) con los números del 1 al 9 de tal forma que no se repitan números en ninguna fila, columna o región. El sudoku inicialmente se presenta con algunas casillas ya rellenas, a modo de *pistas*, y el jugador debe deducir los valores de las casillas vacías. Si el sudoku está bien planteado, la solución es única.



Dado un sudoku completamente relleno, ¿sabrías construir un programa que comprobara si es correcto (es decir, cada fila, columna o región contiene los números del 1 al 9 exactamente una vez)?

Entrada

La entrada comienza con un número que representa el número de casos de prueba que vendrán a continuación.

Cada caso de prueba está formado por 9 líneas, cada una con 9 números entre el 1 y el 9 separados por espacios, que representan un sudoku completamente relleno.

Salida

Para cada caso, se escribirá una línea con la palabra `SI` si el sudoku ha sido resuelto correctamente, y `NO` en caso contrario.

Por ejemplo:

Entrada	Resultado
1 4 1 3 8 2 5 6 7 9 5 6 7 1 4 9 8 3 2 2 8 9 7 3 6 1 4 5 1 9 5 4 6 2 7 8 3 7 2 6 9 8 3 5 1 4 3 4 8 5 1 7 2 9 6 8 5 1 6 9 4 3 2 7 9 7 2 3 5 8 4 6 1 6 3 4 2 7 1 9 5 8	SI

Respuesta: (sistema de penalización: 0 %)

```
1
```

Comprobar

◄ 04C. Problemas con submatrices

Ir a...

Soluciones unidad 4 ►