

Nivel 2do Año

Problem Set

24 de Octubre de 2025

Problema A: Al Azar

- SPOJ code: **TAP2014A**
- Tiempo límite: **1 segundo**
- Autor: **Fidel Schaposnik**

Existen muchos y muy diferentes juegos de cartas, cuyo origen se remonta a tiempos ancestrales. A veces puede sorprender que sigan entreteniendo después de tantos siglos de jugarse con las mismas reglas, pero debemos recordar que cada juego es esencialmente diferente a todos los demás jugados a lo largo de la historia de la humanidad, dada la enorme cantidad de maneras posibles de ordenar las cartas antes de empezar cada partida. De hecho, pocos juegos son divertidos si siempre usamos las cartas en el mismo orden, o si existe una correlación entre las cartas sucesivas que nos permita adivinar el orden en que aparecerán. Por esta razón, es costumbre mezclar las cartas antes de empezar cada partida, y por esta misma razón te pedimos que crees un programa para comprobar que una secuencia de cartas se ha mezclado correctamente.

Para simplificar el problema, nos centraremos únicamente en las barajas españolas, que constan de 48 cartas diferentes. Cada carta se identifica por un valor, que es un número del 1 al 12, y un palo, que puede ser “bastos”, “copas”, “oros” o “espadas”. Sin embargo, para no simplificar demasiado la tarea, tendremos en cuenta que no todos los juegos utilizan las 48 cartas de la baraja. Dada una secuencia de **N** cartas, decimos que está correctamente mezclada si no hay dos cartas sucesivas del mismo valor o palo. En caso contrario, decimos que la baraja no está correctamente mezclada. ¿Pueden ayudarnos a determinar si una secuencia de cartas dada está correctamente mezclada?

Entrada

La primera línea contiene un número entero **T**, el número de casos de prueba ($1 \le \mathbf{T} \le 200$). Siguen **T** casos de prueba.

La primera línea de cada caso de prueba contiene un entero **N**, que representa el número de cartas utilizadas en el juego en cuestión ($2 \le \mathbf{N} \le 48$). Cada una de las siguientes **N** líneas contiene la descripción de una carta de la secuencia que queremos analizar, dada por un entero **V** que representa su valor ($1 \le \mathbf{V} \le 12$) y un carácter **P** que representa su palo: “b” para bastos, “c” para copas, “o” para oros y “e” para espadas. Todas las cartas de la entrada son diferentes y se indican en el mismo orden en que aparecen en la secuencia.

Salida

Para cada caso de prueba, imprima una única línea con un caracter que indique si la secuencia de cartas dada en la entrada se ha mezclado correctamente. El caracter debe ser una “B” si se ha mezclado correctamente y una “M” en caso contrario.

Ejemplos

Entrada	Salida
4 4 1 b 2 c 3 e 4 o 3 1 b 2 b 3 c 3 1 b 1 c 2 e 32 5 c 2 b 4 e 3 o 12 b 1 c 7 e 6 c 12 e 4 o 1 b 6 o 3 e 12 o 11 e 12 c 5 o 10 b 9 o 3 c 4 b 11 c 8 e 9 c 1 e 4 c 8 b 2 o 6 b 9 e 7 b 5 e	B M M B

Problema B: Hermógenes y su Espiral Perfecta

- SPOJ code: **TM25HEP**
- Tiempo límite: **1 segundo**
- Autor: **David Campo**

Hermógenes, estudiante de Ingeniería en Sistemas de la UTN, está obsesionado con los gestores de ventanas tipo mosaico (como DWM). En estos gestores, por defecto, las ventanas no flotan -como en MS Windows- sino que se organizan como mosaicos para ocupar al máximo la pantalla.

Una posible configuracion es que se organicen en forma de **espiral decreciente**, como en la siguiente figura conceptual:

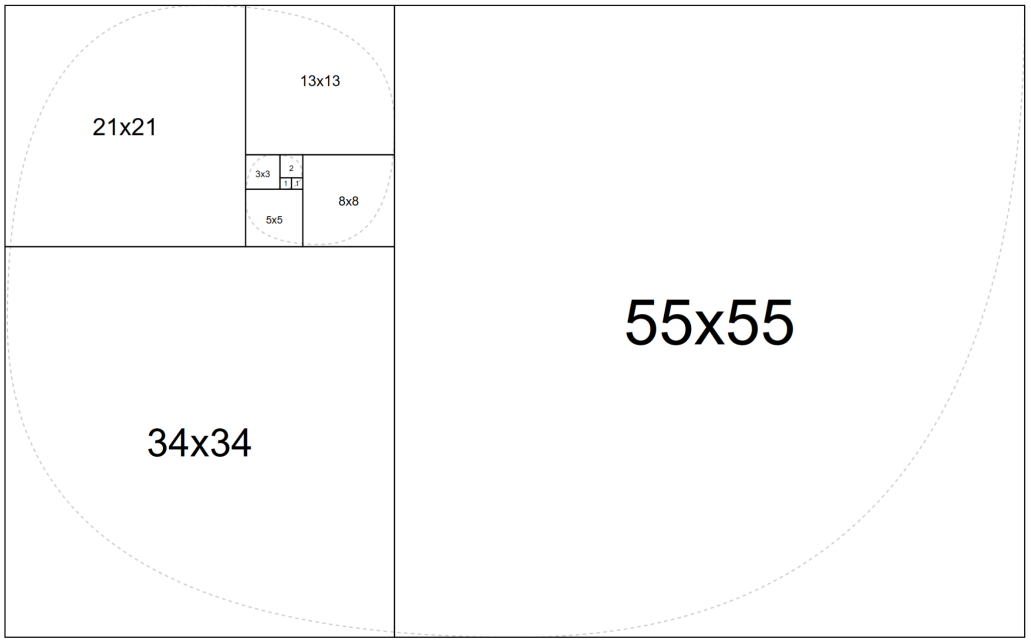


Figura 1: Ventanas con disposición en espiral decreciente, con resolucion total de una hipotética mini-pantalla de 89x55 píxeles. Se observa que la ventana más grande tiene un tamaño de 55 píxeles por 55 píxeles. Luego, la segunda más grande es de 34x34 pixeles y así sucesivamente.

Hermógenes quiere organizar sus ventanas siguiendo una sucesión que considera natural y elegante, inspirada en ciertos patrones que ha visto repetirse en la naturaleza. Hermógenes decidió llevar su obsesión a otro nivel: compró un monitor especial que soporta solamente resoluciones con dicha proporción, es decir cuyo Ancho y Alto son números particulares donde, según él, su razón (es decir el valor más grande de la resolución sobre el más pequeño) es estéticamente agradable.

Para su nuevo gestor de ventanas tipo espiral, impone la restriccion de que todas las ventanas sean de forma cuadrada y cumplan con la siguiente condicion: la primera ventana ocupa el cuadrado más grande posible, y cada nueva ventana ocupa el mayor área restante posible (cumpliendo la condición que sea cuadrada), colocándose a continuación para mantener la forma espiral.

Hermógenes continúa creando ventanas hasta que:

- Cubra todos los píxeles de la pantalla (sin solaparse).
- La ventana mas pequena posible se ha colocado y no pueden crearse ya otras ventanas.

Tu tarea es ayudarlo a descubrir cuántas ventanas puede tener abiertas al mismo tiempo.

Entrada

Varias líneas con un par de enteros cada una. Cada línea representa un caso de entrada con dos enteros positivos que indican el *Ancho* y el *Alto* de la resolución total de la pantalla en pixeles, donde $1 \leq Ancho, Alto \leq 10^9$.

Salida

Un único entero que indica la cantidad de ventanas cuadradas que se pueden crear siguiendo el enunciado.

Ejemplos

Entrada	Salida
`1597 987`	`16`
Con una resolución de 1597x987 píxeles se crean las ventanas: 987x987, 610x610, 377x377, 233x233, 144x144, 89x89, 55x55, 34x34, 21x21, 13x13, 8x8, 5x5, 3x3, 2x2, 1x1, 1x1. Es decir, 16 ventanas en total.	
`55 89`	`10`

Problema C: Emergencia en la Planta Eléctrica

- SPOJ code: **TM25EPE**
- Tiempo límite: **1 segundo**
- Autor: **Matías Trossero**

En plena madrugada, suena el teléfono. Es la planta eléctrica de la ciudad: el controlador de alternadores ha fallado y el suministro de energía está en peligro.

Peor aún, el Ingeniero Johnson, encargado del sistema, se ha ido de vacaciones al Caribe y no hay forma de contactarlo para consultar sobre el misterioso mecanismo que ideó para reiniciar los alternadores defectuosos.

Mientras revisas su escritorio, encuentras unas notas dispersas que mencionan una terminal oculta cerca del área del fallo, y extraños códigos binarios acompañados de un número decimal. Este código podría ser la clave para reiniciar el alternador correcto.

Tu tarea es crear un programa que, dado un número entero K, la cantidad de casos de prueba; muestre por pantalla el código binario correspondiente al alternador que debe reiniciarse para cada caso de prueba. En cada uno de ellos se ingresa un par de enteros Q (cantidad de bits del código binario a generar) y N (número de alternador que corresponde reiniciar).

Entrada:
En la primer línea se ingresa un número entero K $(0 < K \leq 9)$, la cantidad de casos de prueba.
Luego se ingresan los pares de enteros Q $(0 < Q \leq 7)$, la cantidad de bits del código a generar y N $(0 \leq N \leq 13)$ el número de alternador a reiniciar.

Salida:
Para cada caso de prueba: El código binario del alternador seguido por un salto de línea.
En caso de que no se pueda formar un código para un número con dicha cantidad de bits mostrar '`Nro no valido`' seguido de un salto de línea.

Observaciones:
El código Johnson es un código de tipo circular y adyacente, esto quiere decir que nunca se va a observar una situación de intercalación de 0 y 1.
Así, el código Johnson de 3 bits corresponde a la secuencia: 000, 001, 011, 111, 110, 100; y estos valores corresponden a los alternadores 0, 1, 2, 3, 4, y 5
Ejemplos de cadena de 3 bits NO válidas: …010…, …101.

Ejemplos

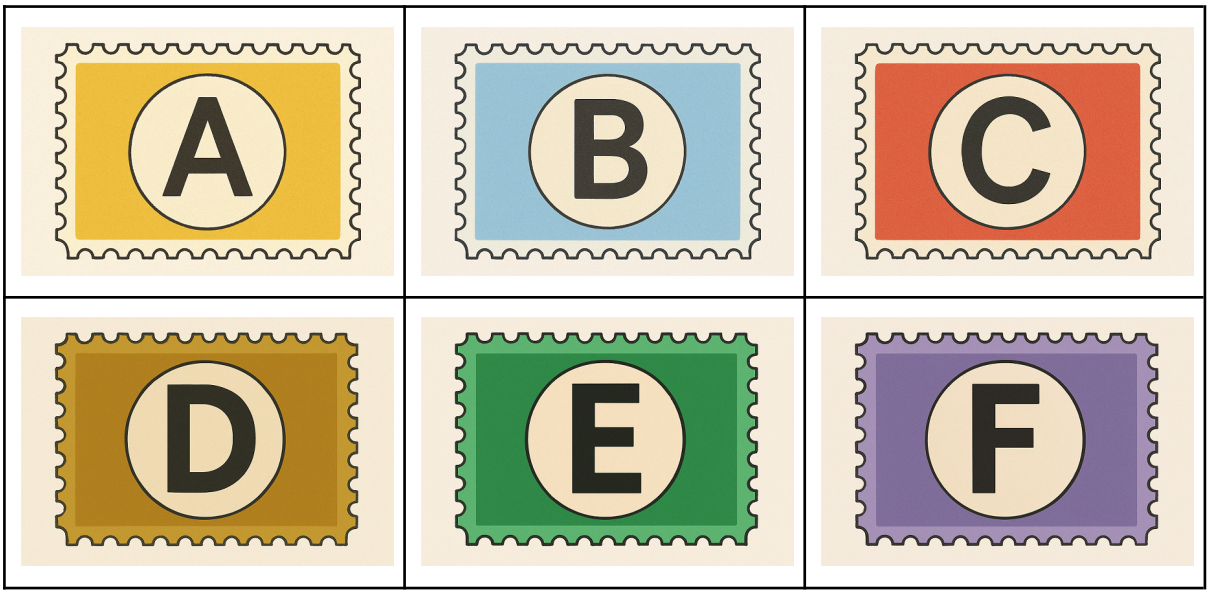
Entrada	Salida
4 7 10 5 5 5 4 7 14	1111000 11111 01111 Nro no valido
6 3 0 3 1 3 2 3 3 3 4 3 5 3 6	000 001 011 111 110 100 Nro no valido

Problema D: Estampillas en una Pila

- SPOJ code: **TM25ESTP**
- Tiempo límite: **1 segundo**

Paolo es un entusiasta coleccionista de estampillas, pero en lugar de guardar las mismas en los clásicos álbumes con organizadores y separadores, decidió plegar las "planchas" de estampillas que adquirió durante los últimos años. En este año 2025 identificó a las planchas que compró siempre de la misma forma.

Cada plancha de estampillas tiene 2 filas y 3 columnas (6 estampillas por plancha). Paolo tiene un sistema para el nombrado de las mismas, en base a las primeras letras del alfabeto, como se observa en la siguiente figura:



En años anteriores, en lugar de etiquetar las estampillas en el orden que usó en este 2025, utilizó otros órdenes, por ejemplo ACEBDF), y ahí empiezan los problemas…
Debes construir un programa para ayudar a Paolo para que pueda ingresar la secuencia correspondiente a un plegado de alguna plancha (sólo pliegues no cortes) en una pila, y determinar si la misma corresponde a alguna plancha adquirida en el 2025 (según las etiquetas mostradas en la figura).

Entrada
En la primer línea se ingresa un número entero N (1 ≤ N ≤ 24), la cantidad de casos de entrada a ser ingresados
Luego se ingresan N líneas y en cada una de ellas vienen separadas por un espacio, las letras -en mayúscula- que identifican a las 6 estampillas según quedaron de arriba a abajo en la pila obtenida cuando se plegó la plancha.

Salida

Para cada caso de entrada:

- Mostrar “VALIDA” si la secuencia corresponde a un plegado válido de una plancha de estampillas adquirida en el 2025.
- Mostrar “NO VALIDA” en otro caso.

Ejemplo

Entrada	Salida
2 A D E B C F B A E F D C	VALIDA NO VALIDA

Problema E: Robot Trufero

- SPOJ code: **TM25ROBT**
- Tiempo límite: **1 segundo**
- Autor: **Exequiel Benavidez**

Debido a un conflicto entre los productores de trufas negras y agrupaciones defensoras de animales se acordó un plan para reemplazar a los perros truferos con robots.

Tu tarea es ayudar a desarrollar el software generador del mapa que debe utilizar el robot para encontrar la ubicación de las trufas. El campo se organiza en una cuadrícula con celdas de un metro por un metro. En el mapa se debe indicar la posición de las trufas con el símbolo asterisco (*) y, en caso que no haya una trufa en la celda, se debe colocar un número que indique cuántas trufas están en contacto con esa celda (considere las 8 direcciones posibles: adelante, atrás, a la izquierda, a la derecha y en las diagonales).



Para probar tu algoritmo se te dará una simulación de un campo con trufas donde se indica por celda de la cuadrícula si hay una trufa (*) o si la celda esta vacía (`.`).

Entrada

En la primer línea vienen 2 enteros *N* y *M* (cantidad de filas y columnas de la cuadrícula, separadas por espacio). ($1 < N \leq 20$ y $1 < M \leq 20$)

En las *N* líneas siguientes vienen cadenas formada por * (si hay una trufa) y `.` (si la celda está vacía).

Salida

N líneas con caracteres separados por espacio, éstos deben ser * (si hay una trufa en esa celda) o dígitos del 0 al 8 (indicando cuántas trufas limitan con la celda respectiva).

Ejemplo

Entrada	Salida
4 4 *... .... .*..	* 1 0 0 2 2 1 0 1 * 1 0 1 1 1 0
3 5 **... ..... .*...	* * 1 0 0 3 3 2 0 0 1 * 1 0 0

Problema F: El Juego de los Primos

- SPOJ code: **TM25JPRIM**
- Tiempo límite: **1 segundo**
- Autor: **Danilo Figueroa**

En esta edición especial del Juego del Calamar, los jugadores no compiten físicamente, sino mentalmente. Cada participante recibe un número consecutivo del 1 al N. Sin embargo, solo los números primos son dignos de continuar a la siguiente ronda.

Primera prueba: “Luz verde, luz roja”. Se eliminan los jugadores cuyos números no son primos. Solo los números primos avanzan a la siguiente etapa.

Segunda prueba: “El puente de cristal”. Entre los jugadores restantes, se eliminan los primos gemelos, es decir, los números primos separados exactamente por 2 unidades (por ejemplo, 11 y 13, 17 y 19). Solo los primos aislados continúan en competencia.

Tercera prueba: “El premio final”. Se anuncia un Premio Total, y solo ganan aquellos jugadores cuyo número divide exactamente el valor del premio.

Entrada

Un entero N ($10 \le N \le 10\,000$): cantidad de jugadores.
Un entero X: valor total del premio.

Salida

Dependiendo del resultado, el programa mostrará uno de los siguientes mensajes:

No hay ganadores para el premio de *X*
El unico ganador de X es el numero: *Y*
El premio de X se dividira entre los competidores: A, B, C, ...

Ejemplo

Entrada	Salida
50 2809	No hay ganadores para el premio de 2809
50 2491	El unico ganador de 2491 es el numero: 47
50 39997	El premio de 39997 se dividira entre los competidores: 23, 37, 47

Problema G: Cartones Credecrecientes

- SPOJ code: **CREDECRE**
- Tiempo límite: **0.1 segundos**
- Autor: **Daniel Ambort**

Para una variante de la tómbola, se quiere confeccionar e imprimir cartones del siguiente tipo:

```
   T1            T2            T3
1  2  3       1  2  4       1  2  5
6  5  4       7  6  5       8  7  6
7  8  9       8  9 10       9 10 11
```

En la matriz, se deben ubicar los números en 1..N, los cuales pueden aparecer una vez como máximo -en cada cartón-. Por ejemplo, T1, es un cartón válido cuando N=9.

T2 es un ejemplo de un cartón válido cuando N=10, y T3 es un ejemplo de un cartón válido cuando N=11.

Los tableros válidos cumplen con las siguientes condiciones:

- Los elemenos de la fila 1 tienen que estar ordenados en forma creciente.
- Los elementos de la fila 2 tienen que estar ordenados en forma decreciente.
- Los elementos de la fila 3 tienen que estar ordenados en forma creciente.
- El menor de los elementos de la fila 2 tiene que ser mayor que el mayor de la fila 1.
- El menor de los elementos de la fila 3 tiene que ser mayor que el mayor de la fila 2.

Los cartones siempre son de 3×3, pero el número N puede variar. Se está organizando un evento multitudinario y se quiere conocer la cantidad de cartones válidos para diferentes valores de N. Ud. debe codificar un programa que lea valores para N, ingresados por teclado, e informe la cantidad de cartones válidos diferentes que se pueden confeccionar para dicho valor de N.

La entrada de datos termina cuando ingresa N=0, los valores de N están en el rango 0..24.

Ejemplo

Entrada	Salida
4 9 11 10 24 0	0 1 55 10 1307504

Problema H: Base-3 Balanceada

- SPOJ code: **TAP2014B**
- Tiempo límite: **1 segundo**
- Autor: **Fidel Schaposnik**

A lo largo de la historia se han desarrollado numerosos sistemas de numeración. Algunos, como los números romanos, se han abandonado casi por completo debido a su incomodidad. Otros sistemas de numeración aún más exóticos solo se utilizan en ciertas aplicaciones, como los números factorádicos en el contexto de la numeración por permutaciones. En este problema, consideraremos un sistema de numeración conocido como *base-3 balanceada*, que aparece de forma natural en el análisis de diversos problemas matemáticos relacionados con balanzas.

El sistema de numeración de base-3 balanceada es similar al sistema decimal o de base-10 que conocemos, ya que es *posicional*. Los sistemas de numeración posicional tienen *dígitos* cuyo orden relativo determina qué potencia de la base los acompaña. Por ejemplo, en base 10 tenemos

$$\mathbf{123} = \mathbf{1}\times 10^2 + \mathbf{2}\times 10^1 + \mathbf{3}\times 10^0.$$

En los sistemas posicionales estándar, los dígitos permitidos son los números del **0** al **B-1**, donde **B** es la base del sistema correspondiente. Por lo tanto, el número **123** en base-10 se escribe en base-3 estándar como `"11120"` porque

$$\mathbf{1}\times 3^4 + \mathbf{1}\times 3^3 + \mathbf{1}\times 3^2 + \mathbf{2}\times 3^1 + \mathbf{0}\times 3^0 = 123.$$

La base-3 balanceada se diferencia de la base-3 estándar en que los dígitos permitidos son **0**, **1** y **-1**, que denotaremos respectivamente como `'0'`, `'+'` y `'-'`. Entonces, el número **123** en base-10 se escribe como `"+----0"` en base-3 balanceada, ya que tenemos

$$\mathbf{1}\times 3^5 + (\mathbf{-1})\times 3^4 + (\mathbf{-1})\times 3^3 + (\mathbf{-1})\times 3^2 + (\mathbf{-1})\times 3^1 + \mathbf{0}\times 3^0 = 123.$$

La conversión de números en base-10 a base-3 balanceada es un proceso mecánico y algo tedioso, por lo que necesitamos un programa que lo haga por nosotros. ¿Puedes ayudar?

Entrada

La primera línea contiene un número entero **T**, el número de casos de prueba (**1 ≤ T ≤ 200**). A continuación, siguen **T** casos de prueba.

Cada caso de prueba se describe mediante una sola línea que contiene un número entero positivo **N**, el número en base-10 que queremos escribir en base-3 balanceada (**1 ≤ N ≤ 1000**).

Salida

Para cada caso de prueba, imprima una única línea que contenga una cadena compuesta únicamente por los caracteres `'0'`, `'+'` y `'-'`, que no comience por `'0'`, y que represente los dígitos del número **N** en base-3 balanceada. Tenga en cuenta que la restricción de que la cadena no comience por `'0'` garantiza que esta representación sea única.

Ejemplo

Entrada	Salida
2 123 729	+----0 +000000

Problema I: SpeedForces

- SPOJ code: **TM25SFOR**
- Tiempo límite: **1 segundo**
- **Este problema no se puede resolver eficientemente con GNU Smalltalk.**

SpeedForces es la principal página de programación competitiva en la ciudad de Santa Fe, donde podés participar de competencias semanales para subir de ELO, o lo que es más común, bajar de ELO.

La más reciente novedad de *SpeedForces* es el *SpeedPredictor*, lo último en predicción de cambios de ELO en competencias. Hay N competencias este año y *SpeedPredictor* ya ha generado las predicciones para tu cambio de ELO para cada una de ellas. Una predicción puede ser de de uno de los siguientes tipos:

1. = *E*, luego de la competencia tu ELO es igual a *E*.
2. + *E*, luego de la competencia a tu ELO se le suma *E*.

¡Pero todavía hay más! Tu entrenador, Matías, esta interesado en saber cual sería tu máximo ELO si compitieras sólo en un rango [L, R] de todas estas competencias. Cabe aclarar que podés saltearte las competencias que quieras con tal de maximizar tu ELO ¿Podés darle una mano a Mati para encontrar la respuesta a cada una de sus consultas?

Entrada

Una línea con un entero **N ($1 \le N \le 2\times10^5$)**, la cantidad de competencias en el año.

Una línea por cada competencia con un símbolo "=" o "+", y un entero **E ($-600 \le E \le +600$)**:

- "= E" significa que tu ELO luego de la competencia es **E**.
- "+ E" significa que tu ELO luego de la competencia se le suma el entero **E**.

Una línea con un entero **C ($1 \le C \le 2\times10^5$)**, la cantidad de consultas de tu entrenador.

Una línea por cada consulta de tu entrenador con dos enteros **L** y **R ($1 \le L \le R \le N$)**.

Salida

C líneas, la i-esima línea debe contener un entero **M** con la respuesta a la i-ésima consulta de Mati.

Ejemplo

Entrada	Salida
5 + -2 = 3 = -2 + 2 + -1 2 1 2 1 5	3 5