

Nivel “Secundaria”
Problemas

24 de Octubre de 2025

Problema A: Aventuras Submarinas en Burbuja Azul

Autor: Alejandro Olivera Torres - Timelimit: 1

Santiago, un instructor de buceo recién regresado de una expedición por el Mar Argentino, ha decidido compartir sus conocimientos con nuevos aprendices en la escuela Burbuja Azul, un centro de buceo recreativo que combina la pasión por el mar con la curiosidad científica.

Durante las prácticas, los alumnos deben aprender a calcular su autonomía de aire bajo el agua, es decir, cuánto tiempo pueden permanecer en el fondo sin superar los límites seguros.

Cada buzo cuenta con un tanque de 12 litros, cargado a 200 bares de presión. Por seguridad, deben reservar 50 bares para el ascenso. El aire se consume de acuerdo a dos factores:
- El consumo superficial, que indica cuántos litros por minuto se usan a nivel del mar.
- La presión ambiente, que aumenta con la profundidad. y se calcula según la siguiente fórmula:

P = d / 10 + 1 donde d es la profundidad en metros, y P la presión en bares.

El consumo real del buzo a esa profundidad será:

C_real = C_superficie × P

Finalmente, conociendo que:

Aire_total = 12 × 200
Aire_seguridad = 12 × 50
Aire_utilizable = Aire_total - Aire_seguridad

El tiempo máximo que el buzo puede permanecer bajo el agua se calcula como:

Tiempo_maximo = Aire_utilizable / C_real

Entrada
Se ingresan dos valores:
- La profundidad de la inmersión *d* (en metros, $5 \leq d \leq 15$).
- El consumo superficial *C_superficie* (en litros por minuto, $5 \leq C_superficie \leq 50$).

Salida
Mostrar una línea con el siguiente formato: Tiempo maximo: *XX* minutos

donde *XX* representa el número entero de minutos máximos que el buzo puede permanecer en el fondo antes de iniciar el ascenso.

Ejemplo

Entrada	Salida
10 25	Tiempo maximo: 36 minutos
15 20	Tiempo maximo: 30 minutos

Problema B: Emergencia en la Planta Eléctrica II

Autor: Matías Trossero - Timelimit: 1

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, dada una cantidad de bits Q, seguida de Q símbolos binarios (0 o 1) devuelva el número decimal N correspondiente al alternador que debe reiniciarse.

Entrada:
En la primer línea se ingresa un número entero *Q* (0 < *Q* <= 7), la cantidad de bits del código a descifrar, seguido del código separando los símbolos (0 ó 1) con un espacio entre ellos.

Salida:
El número del alternador a reiniciar (en 2*Q - 1) seguido por 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, 111; y estos valores corresponden a los alternadores 0, 1, 2, 3, 4, y 5. El Código Johnson de 4 bits es similar, pero con 4 bits!!!. Ejemplos NO válidos: … 0 1 0 …, … 1 0 1 …, etc.

Ejemplos

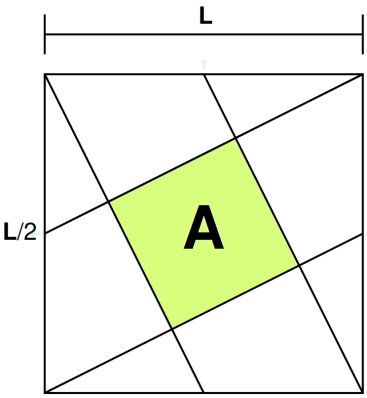
Entrada	Salida
4 1 1 1 1	4
4 0 0 1 1	2
4 1 1 1 0	5
7 1 1 1 1 0 0 0	10

Problema C: El Vitral del artesano

Autor: David Campo - Timelimit: 1

Un artesano en vidrio de Santa Fe fue contratado para realizar la decoración de los vitrales de una catedral. Pensó en un patrón sobrio, basado en figuras geométricas simples.

Una de sus ideas fue la siguiente: para facilitar el ensamblado y encastre de los mismos, cada vitral será cuadrado, con un marco exterior y una figura geométrica interior. Para una partida en particular, decidió utilizar también un cuadrado en el interior; pero rotado, ya que es una de las formas más fáciles de cortar en vidrio. Dado que anticipa mucho trabajo, decidió crear una receta detallada de construcción para que sus ayudantes puedan adelantar el trabajo. En sus instrucciones dejó explicitado: *para construir el vitral, se parte desde el punto medio de cada lado del cuadrado exterior y se traza una línea recta hasta el vértice opuesto. De este modo obtiene cuatro líneas que se cruzan formando el cuadrado sombreado que se muestra en la figura*.



Para preparar los materiales del diseño interior, el artesano te ha pedido ayuda para calcular el área A del cuadrado interno.

Entrada

La primera línea contiene un número entero *T* (0 < *T* <= 10000), que indica la cantidad de vitrales que el artesano debe fabricar.
Las siguientes *T* líneas contienen un número decimal, con hasta dos decimales de precisión, *L* (0.00 < *L* <= 10e6), correspondiente a la longitud del lado del cuadrado exterior (en unidades de longitud).

Salida

Para cada caso de prueba, se debe imprimir en una línea el área A del cuadrado interior.
Dado que la máquina de corte del taller no tiene tanta precisión, basta con mostrar el resultado con dos decimales. Utilizar variables flotantes de doble precisión para el cálculo del valor mostrado en la salida.

Ejemplo

Entrada	Salida
5 0.01 10.00 5000.25 250000.75 9999999.99	0.00 20.00 5000500.01 12500075000.11 19999999960000.00

Problema D: Hermógenes y su Espiral Perfecto

Autor: David Campo - Timelimit: 1

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:

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

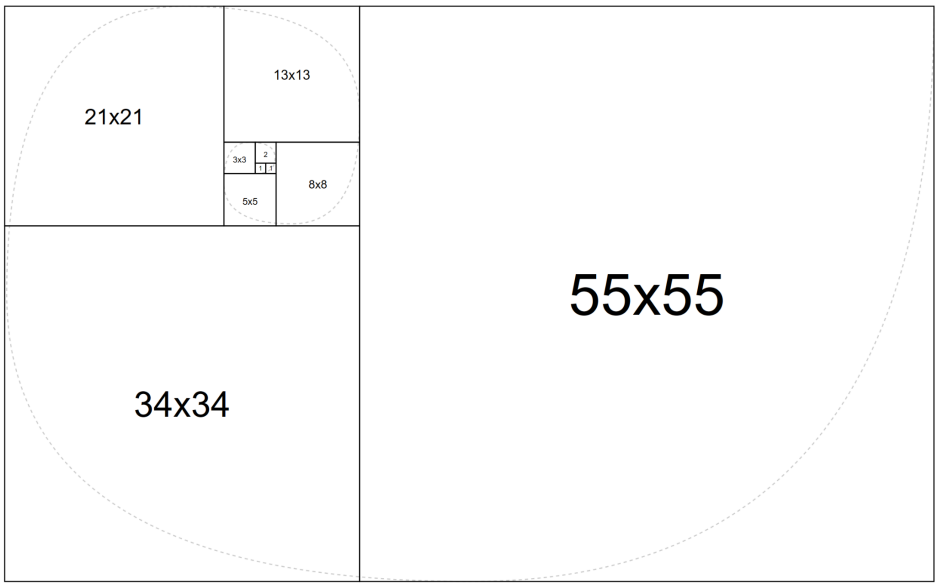


Figura 1: Ventanas con disposición en espiral decreciente, con resolucion total de una hipotética mini-pantalla de 89x55 píxeles.

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.

1 <= *Ancho*, *Alto* <= 10e9

Salida

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

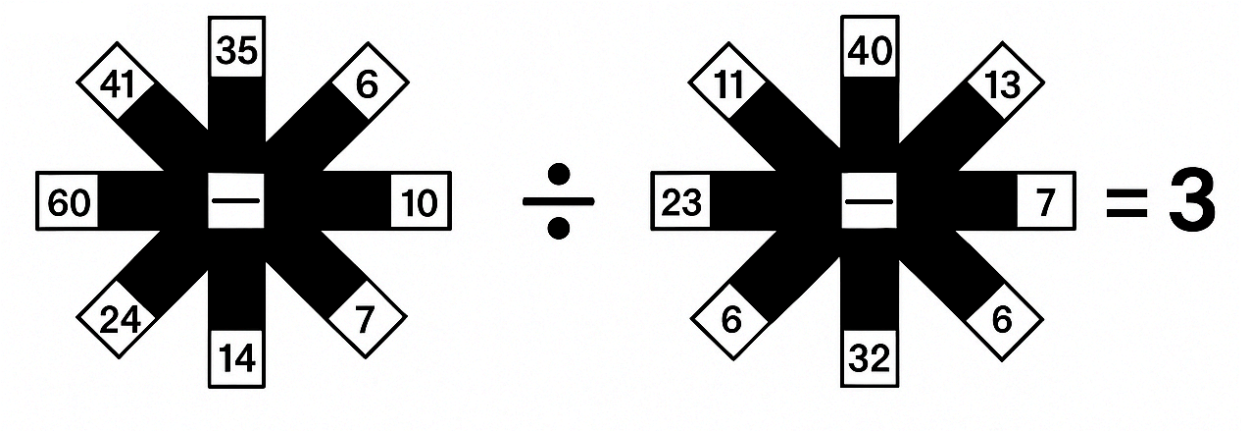
Ejemplos

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 E: Un Cálculo Enroscado

Autor: Matías Trossero - Timelimit: 1

Dos **norias numéricas** (ruedas con cuatro brazos cada una) contiene valores enteros en sus extremos.
Cada noria puede rotarse **en sentido antihorario**, de forma que los valores cambian de posición circularmente.



Queremos encontrar si existe combinación de rotaciones que cumpla la siguiente igualdad:

$$(A_{Izq} - A_{Der}) / (B_{Izq} - B_{Der}) = N$$

El programa debe determinar si existe una rotación de cada noria que satisfaga la igualdad exacta.

Entrada:
Los 8 enteros correspondientes a la primer noria (la de la izquierda) y los 8 enteros de la segunda noria (la de la derecha), estos enteros están sujetos a Q ($0 <= Q <= 100$).
Y por último un flotante F ($0 <= F <= 100$)

Salida:

- En caso de que exista alguna combinación que satisfaga el enunciado un texto por pantalla que diga “Norias validas!”
- Caso contrario, otro texto por pantalla diciendo “No es posible encontrar igualdad :(“
- Ambas seguidas por un salto de línea

Observaciones:

- El ingreso de datos de las norias es de a pares entre un extremo y su opuesto (ver ejemplo de entrada y cotejar con imagen de referencia)
- Entre $(A_{Izq} - A_{Der})$ y $(B_{Izq} - B_{Der})$ se debe realizar una división

Ejemplo:

Entrada	Salida
60 10 41 7 35 14 6 24 23 18 11 7 40 32 13 6 3	Norias validas!
60 12 41 7 35 14 6 24 23 18 11 7 40 32 13 7 3	No es posible encontrar igualdad :(
36 22 1 3 11 4 7 21 6 2 14 15 22 11 13 13 3.5	Norias validas!

Problema F: Exponenciación Eficiente

Autor: David Campo - Timelimit: 1

Gutta cavat lapidem, non vi sed saepe cadendo.
La gota perfora la piedra no por su fuerza, sino por su constancia al caer.
— Ovidio

En múltiples protocolos criptográficos (intercambio de claves, firma digital, verificación), es frecuente calcular valores de la forma:

$$R = B^P \bmod M$$

donde B, P y M pueden ser muy grandes. Tu tarea es, para cada caso de entrada, determinar el valor exacto de R.

Restricciones:
0 <= *B*, *P* <= 2147483647
1 <= *M* <= 46340

Entrada
La entrada contiene múltiples casos de prueba hasta fin de archivo (EOF).
Cada caso está compuesto por tres enteros, en este orden:

B P M

Entre un caso y el siguiente hay una línea en blanco

Salida
Para cada caso de prueba imprime el valor de *R* en una línea.

Ejemplo

Entrada	Salida
5 3 1000 5 3 100 5 3 10 3 18132 17 17 1765 3 2374859 3029382 36123	125 25 5 13 2 13195

Problema G: Carrera en el Velódromo

Timelimit: 1



A beneficio de la cooperadora de la escuela pública del barrio, los deportistas (estrellas locales) decidieron organizar una carrera benéfica entre ellos en el velódromo. Las estrellas del evento son 3: Alfa, Beta y Gama. Alfa es ciclista y cada minuto puede dar A vueltas. Beta es motociclista y cada minuto puede dar B vueltas y Gama es piloto de autos y cada minuto puede dar G vueltas.
Una vez iniciada la carrera, ¿Cuántos minutos pasaran hasta que los tres participantes vuelvan a coincidir en la línea de partida/largada?

Entrada

Hay una única línea de entrada con tres valores enteros separados por un espacio: A, B y G, la cantidad de vueltas que da cada uno de los tres ciclistas en un minuto.

Salida

Mostrar (ver formato en ejemplo), la cantidad de minutos transcurridos hasta que los tres participantes vuelven a coincidir en la línea de partida.

Ejemplo

Entrada	Salida
2 3 4	Minutos Transcurridos: … // reemplazar los "..." por el número de minutos transcurridos con 2 decimales de precisión. (existe un espacio en blanco entre ':' y el número).

Problema H: Robot Trufero

Autor: Exequiel Benavidez - Timelimit: 1

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 de mapa que debe utilizar el robot para crear un mapa con la ubicación de las trufas. El campo se organiza en cuadrículas de un metro por un metro y en el mapa se debe indicar la posición de las trufas con el símbolo asterisco (*) y en caso de que no haya trufa en la celda se debe colocar un número que indique cuántas trufas están en contacto con esa cuadrícula (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 cuadrícula si hay una trufa (*) o si la cuadrícula 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*<=20 y 1<*M*<=20)

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

Salida
N líneas con caracteres separados por espacio, éstos deben ser * (si hay una trufa en esa cuadrícula) o del 0 al 8 (indicando cuantas trufas toca esa cuadricula).

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 I: Maratón Desordenada

Timelimit: 1

Durante la maratón global 2025 llevada a cabo la semana pasada, con atletas de todo el mundo corriendo al mismo tiempo en su ciudad de residencia (en diferentes países y con toma de tiempos desde la app que cada uno tenía instalada en su smartwatch), se registraron los datos de los atletas en un archivo de texto. El archivo estaba ordenado según las posiciones obtenidas en la maratón, pero por algún comportamiento extraño de la nube donde estaba almacenado el mismo, las líneas de ese archivo se mezclaron aleatoriamente…!!!

Se necesita encontrar una solución para visualizar los resultados en el sitio web de la maratón, por lo tanto, deberás construir un programa que lea los datos de entrada y muestre como salida la información del podio (los tres primeros atletas), y la mención especial al atleta más veterano participante de la prueba.

Entrada
En la primer línea se ingresa un número entero *N* (5 <= *N* <= 1000), la cantidad de participantes de la maratón.
Luego se ingresan *N* líneas y en cada una de ellas viene la información registrada para cada atleta:

- Apodo (una cadena de caracteres sin espacios, de hasta 20 caracteres de largo)
- Edad (un entero en 18..99)
- Posición (un entero en 1..*N*)

Observación: se sabe que no hay atletas empatados en su posición.

Salida
- Los atletas premiados y que suben al podio. Para cada uno de los tres primeros, puesto obtenido y apodo. Luego hay una línea en blanco en la salida.
- Apodo del atleta más veterano participante (ver Ejemplos).

Ejemplo

Entrada	Salida
5 PedroCorrecaminos 27 3 MariaVelociraptor 19 1 LuisLiebrePatagonica 58 2 SpeedayClarisa 19 5 BrianSharpenHawk 65 4	1 MariaVelociraptor 2 LuisLiebrePatagonica 3 PedroCorrecaminos Mencion Veterano: BrianSharpenHawk

Problema J: A las vueltas en la Setubal

Autor: Danilo Figueroa - Timelimit: 1

La Unión de Nadadores Tecnológicos de Santa Fe organizará este año una carrera de ultramaratón acuática en la laguna Setúbal. Para favorecer la observación del público, se construirá un circuito circular de 200 metros de perímetro, permitiendo que los competidores realicen varias vueltas. Según la estimación de un meteorólogo, habrá como mínimo una hora de buen tiempo y como máximo cinco horas para el desarrollo del evento.

La competencia contará con tres categorías: Omega, Pi y Epsilon. Para cada una de ellas se conoce el tiempo promedio que sus nadadores tardan en completar una vuelta al circuito. En la largada, los nadadores utilizan andariveles distintos en la línea de partida. Durante la competencia, normalmente no se requieren andariveles, pero cada vez que las tres categorías vuelven a coincidir en la línea de partida, se vuelven a habilitar andariveles para evitar colisiones y molestias entre brazada y brazada…

Entrada:
En la primer línea de entrada vienen tres valores enteros positivos separados por un espacio: *O*, *P* y *E*: el tiempo en minutos que demoran los nadadores de cada categoría en dar una vuelta al circuito ($O < P < E$, y O,P,E son valores enteros positivos).
En la segunda viene *T*, el tiempo en horas(entero positivo) que puede durar la carrera.

Salida:
Mostrar (ver formato en ejemplo), en una única línea, separados por un espacio, el tiempo en minutos en que los nadadores de las tres categorías coinciden y por lo tanto se requiere el uso de los andariveles.

Ejemplo

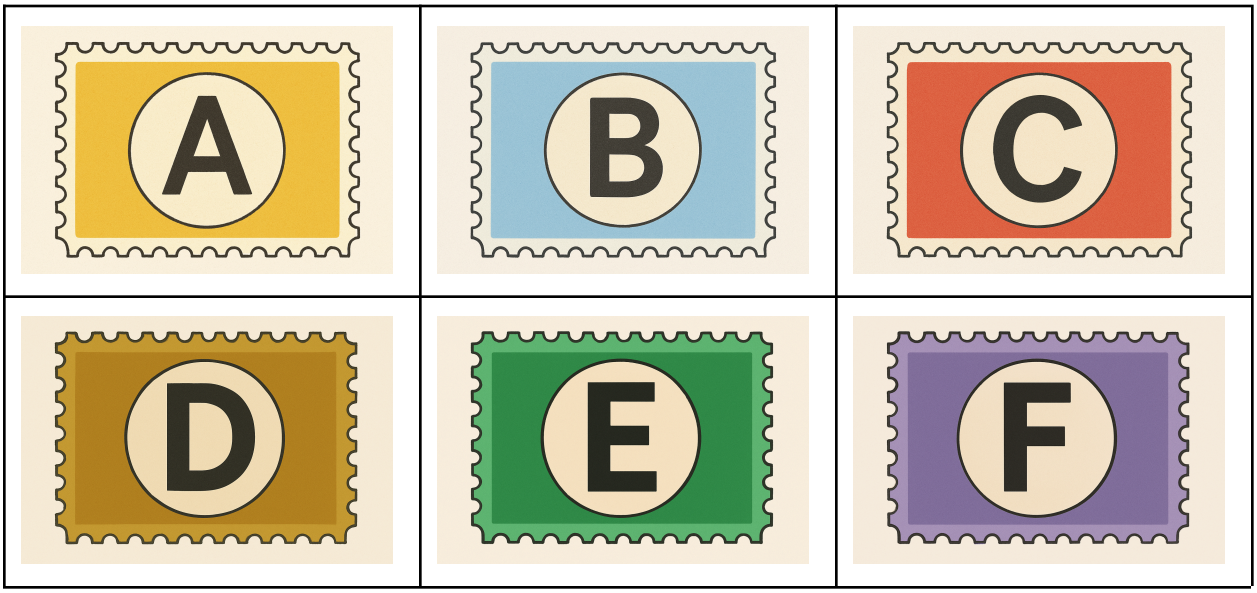
Entrada	Salida
2 4 6 1	12 24 36 48 60
2 3 6 2	6 12 18 24 30 36 42 48 54 60 66 72 78 84 90 96 102 108 114 120
5 6 9 2	90

Problema K: Estampillas en una Pila

Timelimit: 1

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 mayúsculas- 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 L: Pizzas, Programadores y Proporcionalidad

Autor: Matías Trossero - Timelimit: 1

En la Facultad de Ingeniería en Sistemas, se ha descubierto un curioso fenómeno: ¡la velocidad a la que desaparecen las pizzas es directamente proporcional a la cantidad de programadores hambrientos y al tiempo disponible!

Sabemos que:

- CantE estudiantes pueden comer CantP pizzas en CantM minutos.
- Además, se ha comprobado experimentalmente que la velocidad de consumo es constante para todos los estudiantes, sin importar la cantidad de participantes ni el tiempo considerado. (ejemplo, hoy todos estuvieron tan hambrientos entre las charlas de la mañana de TecnoMate y el comienzo de la competencia que tienen una velocidad de ingesta de pizzas máxima y constante en el límite de la genética del ser humano…)

Teniendo en cuenta esta relación proporcional, tu misión es desarrollar un programa capaz de responder preguntas como:

> "¿Cuántas pizzas pueden comer X estudiantes en Y horas, conociendo los valores iniciales de la primera afirmación?"

En otras palabras, deberás calcular la cantidad de pizzas que puede devorar un grupo de estudiantes/programadores cuando se conoce su "potencia pizzera" base.

Entrada:
En la primer línea se ingresa un número entero *K (*0 < *K* <= 4*)*, la cantidad de casos de prueba. Luego se ingresa una terna de valores flotantes *CantE*, CantP y *CantM (*0 <= *CantE, CantP, CantM* <= 5*)*.
Por último se ingresará el entero *X (*0 < *X* <= 10*),* la cantidad de estudiantes que comerán pizza y el flotante *Y (*0 < *Y* <= 6*),* la cantidad de horas que disponen para comer.

Salida:
Un flotante con 2 decimales de precisión, indicando la cantidad de pizzas que pueden comer en dicho lapso.

Observaciones:

- Todos los valores flotantes utilizados son de doble precisión.

Ejemplo:

Entrada	Salida
1 1.5 1.5 1.5 3 0.5	60.00
2 2 2 2 3 0.5 1 1 1 3 0.5	45.00 90.00

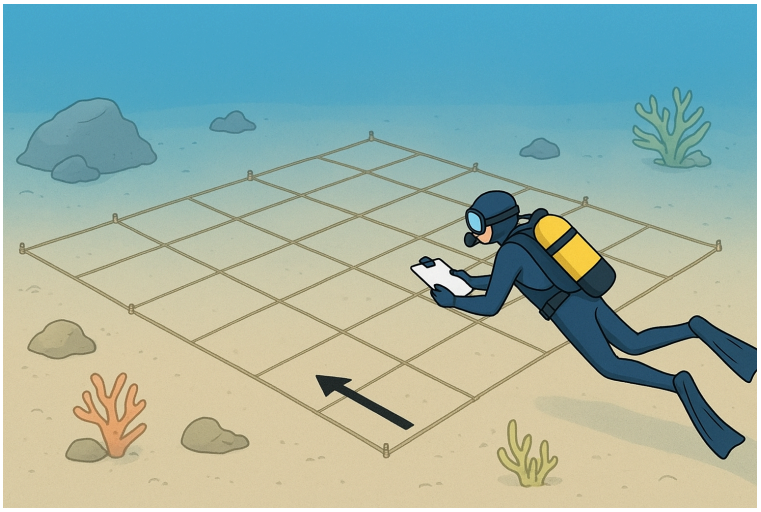
Problema M: Exploración Submarina

Autor: Alejandro Olivera Torres - Timelimit: 1

La tensión era palpable en la cubierta del Alcyon. Las olas mecían bruscamente el barco, pero la quietud reinaba entre los miembros de La Expedición Austral. Bajo el sol implacable del mar argentino, una cuadrícula de exploración, meticulosamente trazada sobre una pizarra, era la única evidencia visible del ambicioso proyecto que estaban a punto de emprender.

El área, una extensión inexplorada del vasto mar argentino, se dividía en una matriz cuadrada de N×N, cada celda representando una porción de misterio submarino. En cada casilla, un número: la cantidad de minutos necesarios para explorar a fondo ese rincón específico.

Santiago, enfundado en su traje de buceo, sentía el peso familiar del equipo sobre sus hombros. No era solo el peso físico, sino también la responsabilidad. Él era el ojo de La Expedición Austral, el explorador que se adentraría en la opacidad del abismo para traer de vuelta conocimiento. Miró la cuadrícula, estudiándola por última vez. Cada número era un desafío, una promesa de descubrimiento.



Cada fila de la cuadrícula representa un recorrido horizontal posible del equipo de buceo.

Durante la misión, los buzos deben desplazarse siempre hacia la derecha dentro de una misma fila, comenzando en una celda determinada, y pueden detenerse en cualquier punto, pero sin que el tiempo total acumulado supere su límite máximo de buceo `t`.

El objetivo de la misión es determinar en qué recorrido (fila) los buzos pueden maximizar el número de celdas consecutivas exploradas sin superar el tiempo máximo permitido `t`.

Entrada
Un número entero N ($5 \leq N \leq 1000$), que representa el tamaño de la cuadrícula.
A continuación, N filas con N valores enteros positivos V ($1<=V<=50$)) correspondientes al tiempo de exploración necesario (en minutos) para cada celda.
Finalmente, se ingresa un entero t, que representa el tiempo máximo de buceo permitido (en minutos).

Salida

La fila del recorrido máximo, numerada de 1 a *N* y la cantidad de celdas recorridas en el tiempo *t* leído. con el formato: Fila Numero: *i* - Cantidad máxima: *c*

Ejemplos

Entrada	Salida
6 25 26 45 22 31 47 29 47 2 25 25 32 34 17 6 32 2 28 37 7 39 18 11 29 6 40 18 18 16 40 13 43 49 41 10 5 68	Fila Numero: 3 - Cantidad maxima: 4
5 2 3 46 22 39 49 7 16 32 2 18 33 25 42 34 10 44 28 4 12 9 30 43 21 1 492	Fila Numero: 1 - Cantidad maxima: 5

Problema N: El Juego de los Números Primos

Autor: Danilo Figueroa - Timelimit: 1

Enunciado. 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 ($10 \leq N \leq 10\,000$): 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