Olimpiada Chilena de Informática 2024

Regional

19 de Octubre, 2024

Información General

Esta página muestra información general que se aplica a todos los problemas.

Envío de una solución

1. Los participantes deben enviar **un solo archivo** con el código fuente de su solución.
2. El nombre del archivo debe tener la extensión `.cpp` o `.java` dependiendo de si la solución está escrita en `C++` o `Java` respectivamente. Para enviar una solución en Java hay que seguir algunos pasos adicionales. Ver detalles más abajo.

Casos de prueba, subtareas y puntaje

1. La solución enviada por los participantes será ejecutada varias veces con distintos casos de prueba.
2. A menos que se indique lo contrario, cada problema define diferentes subtareas que lo restringen. Se asignará puntaje de acuerdo a la cantidad de subtareas que se logre solucionar de manera correcta.
3. A menos que se indique lo contrario, para obtener el puntaje en una subtarea se debe tener correctos todos los casos de prueba incluídos en ella.
4. Una solución puede resolver al mismo tiempo más de una subtarea.
5. La solución es ejecutada con cada caso de prueba de manera independiente y por tanto puede fallar en algunas subtareas sin influir en la ejecución de otras.

Entrada

1. Toda lectura debe ser hecha desde la **entrada estándar** usando, por ejemplo, las funciones `scanf` o `std::cin` en C++ o la clase `BufferedReader` en Java.
2. La entrada corresponde a un solo caso de prueba, el cual está descrito en varias líneas dependiendo del problema.
3. **Se garantiza que la entrada sigue el formato descrito** en el enunciado de cada problema.

Salida

1. Toda escritura debe ser hecha hacia la **salida estándar** usando, por ejemplo, las funciones `printf`, `std::cout` en C++ o `System.out.println` en Java.
2. El formato de salida es explicado en el enunciado de cada problema.
3. **La salida del programa debe cumplir estrictamente con el formato indicado**, considerando los espacios, las mayúsculas y minúsculas.
4. Toda línea, incluyendo la última, debe terminar con un salto de línea.

Envío de una solución en Java

1. Cada problema tiene un *nombre clave* que será especificado en el enunciado. Este nombre clave será también utilizado en el sistema de evaluación para identificar al problema.
2. Para enviar correctamente una solución en Java, el archivo debe contener una clase llamada igual que el nombre clave del problema. Esta clase debe contener también el método `main`. Por ejemplo, si el nombre clave es `marraqueta`, el archivo con la solución debe llamarse `marraqueta.java` y tener la siguiente estructura:

```
public class marraqueta {
  public static void main (String[] args) {
    // tu solución va aquí
  }
}
```

3. Si el archivo no contiene la clase con el nombre correcto, el sistema de evaluación reportará un error de compilación.
4. La clase no debe estar contenida dentro de un *package*. Hay que tener cuidado pues algunos entornos de desarrollo como Eclipse incluyen las clases en un *package* por defecto.
5. Si la clase está contenida dentro de un package, el sistema reportará un error de compilación.

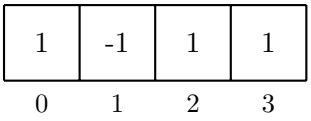
Problema A

Apuesta de alto riesgo

nombre clave: `apuesta`

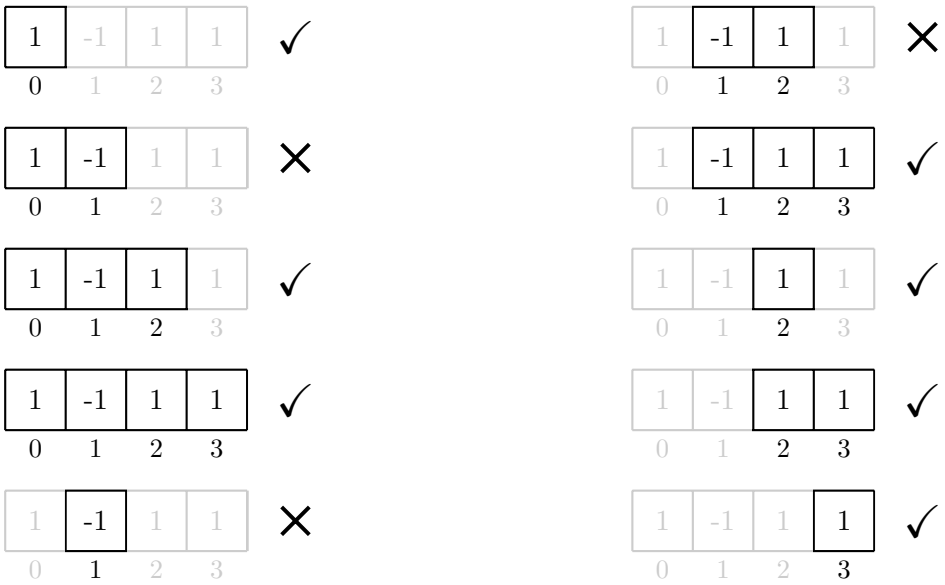
Fernanda y decidió hacer una apuesta de alto riesgo con sus amigos. Fernanda lanza una moneda. Si cae cara, pasarán a la Industria Chilena de Papas Caseras (ICPC) a comerse una porción de papas, de lo contario, se quedarán de brazos cruzados todo el día. Lamentablemente, la moneda cayó sello, por lo que el destino decidió que se perderán de esta increíble oportunidad. Pero Fernanda, no contenta con el resultado, decidió hacer una nueva apuesta en la que está segura podrá ganar.

En esta nueva apuesta, Fernanda lanza n monedas y anota los resultados en una lista. Podemos representar esta lista con un arreglo de enteros dónde 1 representa que la moneda cayó cara y -1 que cayó sello. La siguiente figura representa un escenario con 4 monedas.



Para ganar la apuesta, Fernanda debe contar la cantidad de subarreglos contiguos donde la cantidad de caras es estrictamente mayor que la cantidad de sellos. Si un subarreglo cumple con esta condición, decimos que el subarreglo es exitoso. Si el número de subarreglos exitosos es la mayoría, entonces Fernanda ganará la apuesta y cumplirá su sueño de ir a la ICPC.

La siguiente figura muestra todos los subarreglos para el ejemplo anterior. Cada caso está marcado con un ticket (✓) si el subarreglo es exitoso o con una cruz (✗) si no lo es. En este ejemplo, como la cantidad subarreglos exitosos es la mayoría, Fernanda ganaría la apuesta.



Fernanda ya lanzó las monedas y se puso a contar la cantidad de subarreglos exitosos. A pesar de sus avanzados conocimientos de combinatoria, se está demorando mucho y la ICPC cierra en menos de 4 horas. ¿Podrías ayudarla a contar la cantidad de subarreglos exitosos para que alcance a ir a la ICPC antes de que cierre?

Entrada

La primera línea contiene un único entero n ($1 \leq n \leq 10^6$), correspondiente a la cantidad de monedas.

La segunda línea contiene n enteros. El entero i-ésimo corresponde al resultado de lanzar la i-ésima moneda. El valor será 1 si la moneda cayó cara y -1 si cayó sello.

Salida

La salida debe contener un único entero correspondiente a la cantidad de subarreglos exitosos, es decir, la cantidad de subarreglos donde la cantidad de caras es estrictamente mayor que la cantidad de sellos.

Subtareas y puntaje

ieron

Subtarea 1 (10 puntos)
Se probarán varios casos de prueba donde $n \leq 4$.

Subtarea 2 (20 puntos)
Se probarán varios casos de prueba donde $n \leq 100$.

Subtarea 3 (30 puntos)
Se probarán varios casos de prueba donde $n \leq 10^4$.

Subtarea 4 (40 puntos)
Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
4 1 -1 1 1	7

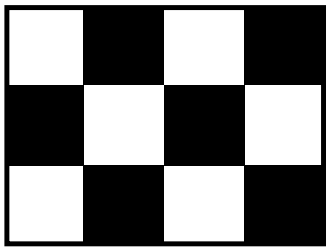
Entrada de ejemplo	Salida de ejemplo
5 1 1 1 1 1	15

Problema B
Baldosas

nombre clave: `baldosas`

Después de mucho meditarlo, Ana ha decidido postular a la Organización de Cocinas Impecables (OCI). La OCI es muy estricta y pone ciertos requisitos que las cocinas de todos sus miembros deben cumplir. Uno de ellos exige que el piso de la cocina tenga un novedoso patrón de baldosas con estilo de ajedrez. Este patrón consiste en formar una grilla donde se alternan baldosas de color blanco y color negro. Adicionalmente, de acuerdo a la última moda, el cuadrado superior izquierdo siempre debe ser de color blanco.

La cocina de Ana no cumple con el requisito y por lo tanto debe remodelarla. Su cocina es un rectángulo de n metros de alto y m metros de ancho. Para poder formar el patrón, Ana ha decidido comprar baldosas de 1×1 metros y por lo tanto necesitará un total de $n \times m$ baldosas para cubrir todo su piso.



Una cocina de 3×4 metros.

Ana no quiere desperdiciar material, y por lo tanto desea comprar la cantidad exacta de baldosas que requiere de cada color. La tienda cierra en menos de 4 horas, y no hay tiempo que perder. ¿Podrías ayudarla escribiendo un programa que determine la cantidad exacta de baldosas necesarias de cada color?

Entrada

La entrada consiste en una sola línea con dos enteros n y m ($1 \leq n \cdot m \leq 10^{18}$), representando respectivamente el alto y ancho de la cocina.

Salida

La salida debe contener una única línea con dos enteros x e y, representando respectivamente la cantidad de baldosas blancas y la cantidad de baldosas negras que Ana debe comprar para poder completar el patrón.

Subtareas y puntaje

Subtarea 1 (25 puntos)
Se probarán varios casos de prueba donde $1 \leq n \leq 10^3$ y $1 \leq m \leq 10^3$.

Subtarea 2 (25 puntos)
Se probarán varios casos de prueba donde $n = 1$ y $1 \leq m \leq 10^{18}$.

Subtarea 3 (25 puntos)
Se probarán varios casos de prueba donde $1 \leq n \cdot m \leq 10^{18}$ y además ambos son pares.

Subtarea 4 (25 puntos)
Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
2 2	2 2

Entrada de ejemplo	Salida de ejemplo
3 3	5 4

Problema C
Ricardo y su nóctulo

nombre clave: `noctulo`

Ricardo no tiene ni un perro ni un gato. Ricardo tiene una mascota muy exótica: un nóctulo. Para los que no sepan, un nóctulo es un murciélago chico que vive en el desván. Ricardo y su nóctulo casi nunca se ven. El problema es que ambos tienen horarios de trabajo muy rígidos y por lo tanto es difícil que sus horarios libres coincidan.

Dependiendo del día de la semana, Ricardo trabaja de día o de noche. Su horario se describe como una cadena de 7 caracteres en dónde cada carácter representa su horario de trabajo ese día de la semana: `'D'` significa que trabaja de día y `'N'` que trabaja de noche. Por ejemplo, la cadena `"DDDDNNN"` representa que Ricardo trabaja de día los lunes, martes, miércoles y jueves, y que trabaja de noche los viernes, sábados y domingos.

Los nóctulos no funcionan de la misma manera, pues ellos tienen una semana de 5 días: noctulunes, noctumartes, noctumiércoles, noctujueves y noctuviernes, y después del noctuviernes vuelve a ser noctulunes. Por lo tanto, para representar el horario del nóctulo se usa una cadena de 5 caracteres.

Ricardo y su nóctulo tienen muchas ganas de verse, pero no saben cuándo ambos van a estar libres al mismo tiempo. Si ambos trabajan de día, se pueden ver en la noche, si ambos trabajan de noche se pueden ver en el día, pero si no trabajan al mismo tiempo, no se pueden ver ese día. Hoy justo es lunes y noctulunes al mismo tiempo, ¡ayúdalos determinando cuántos días deben esperar para poderse ver! Si nunca se van a ver, tienes que darles las malas noticias imprimiendo `"No nos vemos nunca"`.

Entrada

La entrada consiste de dos líneas.

La primera línea contiene la cadena que representa el horario de Ricardo. La cadena será de largo 7 y estará formada solo de los caracteres `'D'` y `'N'`.

La segunda línea contiene la cadena que representa el horario del nóctulo de Ricardo. La cadena será de largo 5 y estará formada solo de los caracteres `'D'` y `'N'`.

Salida

La salida debe contener un único entero que represente en cuántos días más Ricardo y su nóctulo podrán verse. Es decir, cuántos días deben esperar para que ambos trabajen al mismo tiempo considerando que hoy es lunes y noctulunes al mismo tiempo. Si sus horarios nunca coinciden y por lo tanto no pueden verse nunca, debes imprimir `"No nos vemos nunca"`.

Subtareas y puntaje

Subtarea 1 (30 puntos)

Se probarán varios casos de prueba en los que se garantiza que van a poder verse en los primeros 5 días.

Subtarea 2 (70 puntos)

Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
DNNDDNN DDNNN	0

Explicación: En este caso coinciden el primer día y por lo tanto deben esperar 0 días.

Entrada de ejemplo	Salida de ejemplo
NNNDDNN DDDNN	8

Explicación: Las siguientes tablas muestran los calendarios de Ricardo y su nóctulo a partir de los cuales podemos concluir que deben esperar 8 días para que ambos trabajen al mismo tiempo.

Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
00 - Noche	01 - Noche	02 - Noche	03 - Día	04 - Día	05 - Noche	06 - Noche
07 - Noche	**08 - Noche**	09 - Noche	10 - Día	11 - Día	12 - Noche	13 - Noche

Noctulunes	Noctumartes	Noctumiércoles	Noctujueves	Noctuviernes
00 - Día	01 - Día	02 - Día	03 - Noche	04 - Noche
05 - Día	06 - Día	07 - Día	**08 - Noche**	09 - Noche

Entrada de ejemplo	Salida de ejemplo
NNNNNNN DDDDD	No nos vemos nunca

Explicación: Dado que Ricardo siempre trabaja en la noche y su nóctulo siempre trabaja en el día, sus horarios libres nunca coinciden.

Problema D
Tranvía en Chuchunco

nombre clave: `tranvia`

Conscientes de la crisis climática, la ciudad de Chuchunco han decidido construir un nuevo sistema de transporte público sustentable. Tras una larga evaluación, el consejo ciudadano de transporte ha decidido que la mejor opción es construir un *tranvía*.

En su primera fase, el proyecto contempla la construcción de una línea separada del tráfico, con intersecciones a desnivel en puntos estratégicos. En esta primera fase también se espera que la línea opere en solo una dirección siguiendo el flujo del tráfico a la hora punta.

La línea tendrá un largo L y contará con N estaciones numeradas de 1 a N. La estación i estará a una distancia d_i ($0 \leq d_i \leq L$) desde el inicio de la línea. Para su apertura, también está considerada la compra de M trenes articulados de piso bajo de última generación. Estos trenes serán numerados de 1 a M.

La línea contará con un moderno sistema de Control de Trenes Basado en Comunicaciones (CBTC por sus siglas en inglés). El sistema CBTC permitirá conocer la posición exacta de los trenes de forma de tener una gestión de tráfico eficiente y segura. Aprovechando esta tecnología, el consejo de transporte ha decidido instalar pantallas en las estaciones que muestren el tiempo estimado de llegada del próximo tren.

Las pantallas obtendrán la información de un servidor central que mantendrá el estado de cada uno de los trenes. En condiciones normales, el estado de un tren se representa con un entero p indicando su posición desde el inicio de la línea. El estado de un tren también puede ser desconocido en caso de no estar en servicio o haber fallos de conexión.

En un inicio, el estado de todos los trenes es desconocido. A medida que estos entran en funcionamiento, el servidor va recibiendo eventos indicando el último estado conocido de los trenes. Basándose en el último estado conocido, el servidor debe responder el tiempo estimado de llegada del próximo tren asumiendo que los trenes se seguirán moviendo a una velocidad constante unitaria. Específicamente, para una estación i a distancia d_i, decimos que el próximo tren corresponde al tren más cercano cuya posición actual es **menor o igual** que d_i. Dada la posición p del próximo tren, el tiempo estimado de llegada se calcula como $(d_i - p)$.

Concretamente, el servidor deber responder a dos tipos de eventos:

- **Tipo 1**: Dada dos enteros j y p, este evento indica que la última posición conocida del tren j es p. Si p es igual a -1 entonces la posición del tren es desconocida y este ya no debe ser considerado al responder los eventos de tipo 2.
- **Tipo 2**: Dada una estación i, el servidor debe responder el tiempo estimado de llegada del próximo tren o determinar que hay ningún tren que se aproxima.

Notar que debido a fluctuaciones en el sistema, no hay garantías en como evolucionan las posiciones de los trenes. Estos pueden retroceder o desaparecer del sistema en cualquier momento. El servidor siempre responderá a los eventos de tipo 2 usando la posición actual de los trenes.

El consejo de transporte tiene poca experiencia en programación. ¿Podrías ayudarlos implementando el servidor?

Entrada

La primera línea de la entrada contiene 4 enteros L, N, M y E:

- L es el largo de la línea ($0 < L \leq 10^9$)
- N es la cantidad de estaciones ($0 < N \leq 10^5$)
- M es la cantidad de trenes ($0 < M \leq 10^5$)
- E es el número de eventos que debe procesar el servidor ($0 < E \leq 4 \cdot 10^5$)

La segunda línea contiene N enteros en orden creciente, todos entre 0 y L, correspondientes a las distancias donde se ubican cada una de las estaciones. Se garantiza que todas las estaciones estarán en una posición distinta.

Luego siguen E líneas que describen cada uno de los eventos. Cada línea comienza con un entero t ($t = 1$ o $t = 2$) indicando el tipo de evento:

- Si $t = 1$ siguen dos enteros j ($1 \leq j \leq M$) y p ($-1 \leq p \leq L$), indicando que la última posición conocida del tren j es p. Si p es igual a -1 la posición del tren es desconocida. En caso contrario, p indica la nueva posición del tren desde el inicio de la línea.
- Si $t = 2$ sigue un entero i indicando que el servidor debe responder el tiempo estimado de llegada del próximo tren para la estación i.

Salida

La salida debe contener la respuesta a cada evento de tipo 2. Cada respuesta debe aparecer en una nueva línea en el orden en que los eventos aparecen en la entrada. La respuesta debe ser un entero correspondiente al tiempo estimado de llegada del próximo tren o -1 en caso de no haber un próximo tren.

Subtareas y puntaje

Subtarea 1 (30 puntos)

Se probarán varios casos de prueba donde $M \leq 10^3$ y $E \leq 10^3$.

Subtarea 2 (70 puntos)

Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
100 3 2 10 20 50 80 2 1 1 1 10 2 2 2 2 1 1 20 1 2 40 2 1 2 2 1 2 -1 2 3	-1 40 40 0 10 60