Olimpiada Chilena de Informática 2020

19 de Diciembre, 2020

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
Empacadora de supermercado

nombre clave: `bolsa`

Llegó el verano y Maiki ya no sabe qué hacer con tanto tiempo libre ahora que no tiene clases. Para no aburrirse, consiguió un trabajo como empacadora en un supermercado. La paga no es muy buena pero es algo, y es mejor que quedarse en la casa sin hacer nada.

El trabajo es sencillo. Maiki solo debe esperar en la caja y guardar en bolsas los productos que los clientes compran. Los productos son de tamaños variados y a veces es difícil distribuirlos en distintas bolsas. Como la única instrucción que le han dado es tratar de no malgastar las bolsas, Maiki determinó una simple estrategia. Dada una bolsa de capacidad x, se asegurará de que la suma de los tamaños de todos los productos en la bolsa sea al menos $x/2$. Naturalmente, la suma debe ser también menor o igual que x pues esta es la capacidad de la bolsa.

Dado un tamaño de bolsa x y una lista de productos, Maiki ahora se pregunta si es siquiera posible escoger un subconjunto de los productos y meterlos en la bolsa de forma que se cumplan las restricciones anteriores.

Entrada

La entrada está descrita en dos líneas. La primera línea contiene dos enteros N y x ($1 \leq N \leq 10^5$, $0 \leq x \leq 10^9$), correspondientes respectivamente a la cantidad de productos y la capacidad de la bolsa. La segunda línea contiene n enteros $a_1, \ldots, a_N$ ($1 \leq a_i \leq 10^4$), correspondientes a los tamaños de los productos.

Salida

La salida debe contener una sola línea con un 1 en caso de existir un subconjunto de los productos cuyos tamaños sumados den como resultado un valor mayor o igual a $x/2$ y menor o igual a x. La salida debe contener un 0 en caso contrario.

Subtareas y puntaje

Subtarea 1 (10 puntos)
Se probará varios casos en los que $N = 2$.

Subtarea 2 (40 puntos)
Se probará varios casos en los que $1 \leq N, x, a_i \leq 10^3$.

Subtarea 3 (50 puntos)
Se probará varios casos sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
2 5 1 2	1

Entrada de ejemplo	Salida de ejemplo
3 7 1 2 8	0

Problema B

Llaves

nombre clave: `llave`

El equipo de hackers de la Organización de Cyberseguridad Informática (OCI) ha decidido hackear al gobierno. Después de largas horas planeando su ataque, han determinado que la única forma es infiltrarse al edificio gubernamental para plantar un gusano informático en el servidor principal.

El servidor principal se encuentra ubicado en la sala de informática del edificio gubernamental. La sala no tiene ningún tipo de resguardo, salvo que cierran con llave la puerta por las noches. Los hackers de la OCI han logrado obtener una copia de los planos técnicos del edificio, entre los cuales se encuentra detallada la forma de la cerradura. Uno de los hackers de la OCI tiene estudios en cerrajería y fue capaz de crear una réplica de la llave a partir de la información en los planos.

Llegado el día del ataque, los hackers insertan confiados la llave en la cerradura, pero para su sorpresa, esta no encaja. El hacker con estudios en cerrajería asegura que aún puede arreglar la llave y salvar el ataque.

La llave tiene n *dientes* de distintas alturas que deben alinearse con el sistema interno de la cerradura. Lamentablemente, por errores en su manufactura, las alturas de los dientes no se alinean perfectamente. El hacker con estudios en cerrajería cree que puede limar algunos dientes para arreglar la llave.

Cada diente puede ser limado de forma independiente, reduciendo su altura; sin embargo, para preservar la integridad estructural de la llave, solo pueden limarse a lo más m dientes. Dado el valor máximo m, las alturas de los dientes de la llave y las alturas que estos deberían tener para poder abrir la cerradura, tu tarea es determinar si es posible arreglar la llave limándola.

Entrada

La primera línea de la entrada contiene dos enteros n y m ($0 \leq m \leq n, 0 < n \leq 100$), correspondientes respectivamente a la cantidad de dientes en la llave y al valor máximo de dientes que es posible limar. La segunda línea contiene n enteros positivos y menores o iguales que 10, correspondientes a las alturas de cada uno de los dientes de la llave. Finalmente, la tercera línea contiene n enteros positivos, también menores o iguales que 10, correspondientes a las alturas que los dientes deberían tener para poder abrir la cerradura.

Salida

La salida debe contener una sola línea con un `si` en caso de ser posible limar la llave de forma que se logre abrir la cerradura, y `no` en caso contrario.

Subtareas y puntaje

Subtarea 1 (100 puntos)

Este problema contiene una sola subtarea con las restricciones descritas en el enunciado.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
3 3 3 4 5 2 3 4	si

Entrada de ejemplo	Salida de ejemplo
3 2 3 4 5 2 3 4	no

Problema C
El lokómon legendario

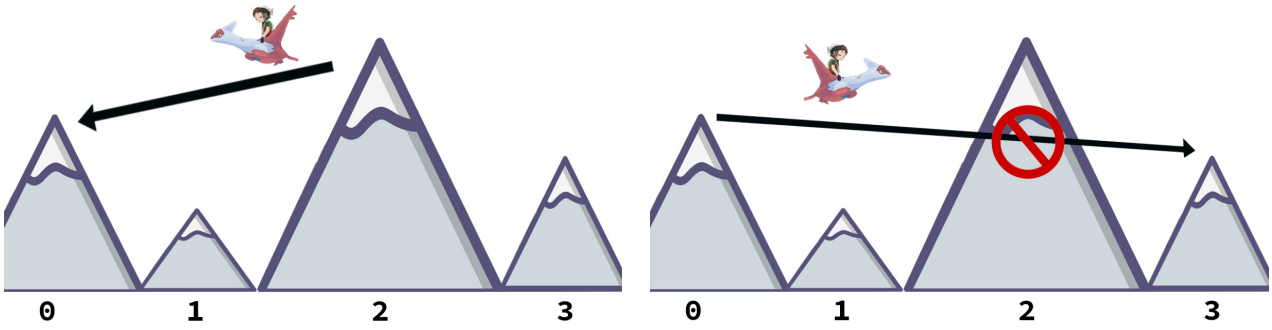
nombre clave: `lokomon`

Tu amigo Ashvier, un osado maestro lokómon, logró escalar hasta la cima de la montaña más alta de la Cordillera Norte. Tras una larga batalla, su misión ha sido todo un éxito y consiguió capturar al lokómon legendario Blatiaz.

Ashvier ahora debe descender y regresar al centro lokómon para así poder curar a sus lokómones. Lamentablemente, su kit de alpinismo se cayó por el acantilado durante la batalla y debe buscar una forma alternativa para descender. Afortunadamente, su nuevo lokómon legendario Blatiaz tiene la habilidad de planear.

La Cordillera Norte está formada por n montañas de diferentes alturas, distribuidas una al lado de la otra a lo largo de una línea. Las montañas son enumeradas de izquierda a derecha entre 0 y $n-1$. La habilidad de Blatiaz le permite planear desde la cima de una montaña hasta la cima de otra montaña más baja, siempre y cuando una montaña más alta no bloquee el desplazamiento. Dadas dos montañas en posiciones i y j, con alturas h_i y h_j $(h_j < h_i)$ respectivamente, decimos que una montaña en la posición k $(min(i,j) < k < max(i,j))$ *bloquea* el desplazamiento si su altura h_k es mayor o igual que h_i.

A continuación se muestra un ejemplo para $n = 4$ montañas. En la figura de la izquierda se puede apreciar que Blatiaz puede descender desde la montaña 2 hasta la montaña 0, pues esta es de menor altura. Por el contrario, como se muestra en la figura de la derecha, no es posible descender de la montaña 0 hacia la montaña 3, pues, a pesar de que la montaña 3 es más baja, la montaña 2 bloquea el desplazamiento.



Desde la montaña 2, Blatiaz también podría descender a las montañas 1 y 3, pero como es su primera vez planeado sobre Blatiaz, Ashvier tomará algunas precauciones. Partiendo de la cima de la montaña más alta, Ashvier siempre descenderá a la cima de **la montaña más alta que no esté bloqueada**. Siguiendo este proceso, llegará a una montaña desde la cual no es posible seguir descendiendo. Llamaremos a esta montaña la *montaña final*.

Dada la descripción de las montañas, a Ashvier le gustaría saber cuál será la montaña final. De esta forma puede avisarle de antemano a su amigo el profesor Perezoak, quien lo estará esperando para llevarlos al centro lokómon. ¿Podrías ayudarlos?

Entrada

La entrada consiste de dos líneas. La primera contiene un solo entero n ($2 \leq n \leq 1\,000\,000$) correspondiente al número de montañas que conforman la Cordillera Norte. La segunda línea contiene n enteros que describen las montañas. El primer entero corresponde a la posición de la montaña más alta, el segundo a la posición de la segunda montaña más alta y así hasta el último entero que corresponde a la posición de la montaña más baja.

Salida

La salida debe contener un único entero, correspondiente a la posición de la *montaña final*.

Subtareas y puntaje

Subtarea 1 (30 puntos)

Se probará varios casos en los que $n < 1000$.

Subtarea 2 (70 puntos)

Se probará varios casos sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
4 2 0 3 1	1

Entrada de ejemplo	Salida de ejemplo
10 0 1 2 3 4 5 6 7 8 9	9

Entrada de ejemplo	Salida de ejemplo
10 0 9 1 8 2 7 3 6 4 5	5

Problema D

Jaque mate

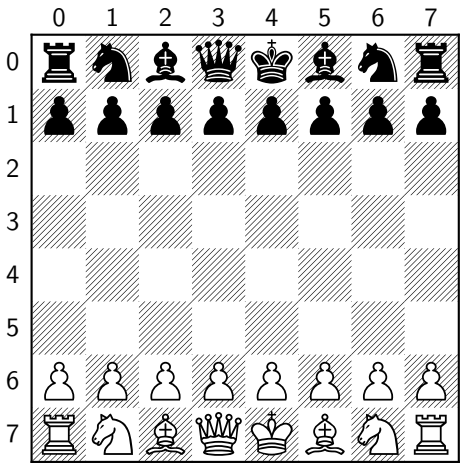
nombre clave: `jaque`

Elizabeth es una maestra mundial de ajedrez. Tras muchas victorias, Elizabeth ha notado que a algunos jugadores les toma demasiado tiempo darse cuenta de que perdieron, por lo que ella debe explicarles el jaque mate.

Tu tarea en este problema es simple. Debes ayudar a Elizabeth creando un programa que, dada la descripción de un tablero, determine si ella tiene a su oponente en jaque mate. A continuación se describen las reglas del ajedrez y cuándo un jugador está en jaque mate.

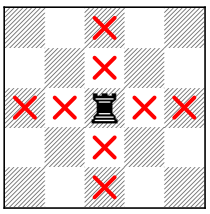
Reglas del juego

El ajedrez se juega entre dos jugadores sobre un tablero de 8×8 casillas. En el contexto de este problema, enumeraremos las columnas del tablero de izquierda a derecha entre 0 y 7, y las filas de arriba hacia abajo entre 0 y 7. Uno de los jugadores controla 16 piezas blancas y el otro controla 16 piezas negras. **Elizabeth siempre controlará las piezas negras**. Existen 6 tipos de piezas: la torre (♖), el alfil (♗), la reina (♕), el rey (♔), el caballo (♘) y el peón (♙). Cada jugador comienza el juego con 8 peones, 2 torres, 2 caballos, 2 alfiles, 1 reina y 1 rey. Al inicio del juego las piezas negras son ubicadas en la parte superior del tablero y las blancas en la parte inferior como se muestra en la siguiente figura.



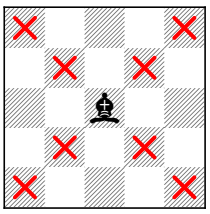
Los jugadores toman turnos y en cada turno deben mover una de sus piezas desde una casilla a otra. Cada tipo de pieza tiene distintas reglas para poder moverse. A continuación se describen las reglas para mover las piezas, excluyendo al peón que será descrito más adelante. Adicionalmente, para cada pieza se muestra una figura marcando con una cruz las posibles casillas en un tablero de 5×5 a las que podría moverse cada pieza en un solo movimiento.

Torre



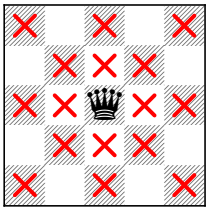
La torre puede moverse en línea recta de forma horizontal o vertical cuantas casillas quiera en un movimiento, pero solo en una dirección.

Alfil



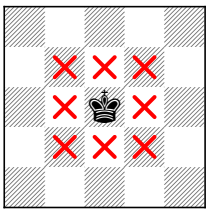
El alfil es similar a la torre, pero en vez de moverse de forma horizontal o vertical, puede moverse a lo largo de una diagonal cuantas casillas quiera en un movimiento.

Reina



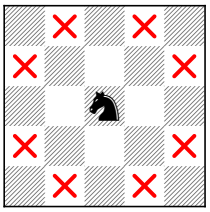
La reina es una combinación de la torre y el alfil ya que puede moverse en diagonal, horizontal o verticalmente cuantas casillas quiera en un movimiento.

Rey



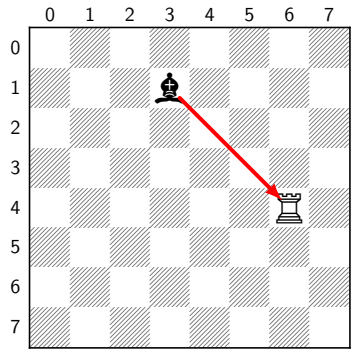
El rey, al igual que la reina, puede moverse en cualquier dirección, pero solo una casilla por movimiento.

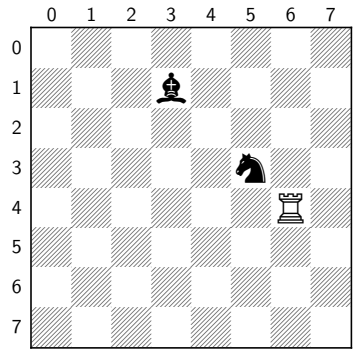
Caballo

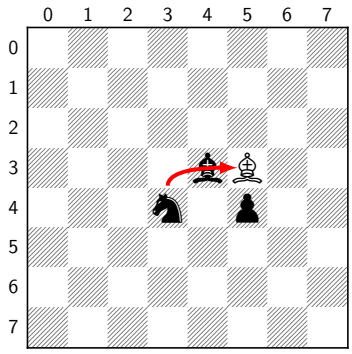


El caballo tiene un movimiento en forma de "L", es decir, siempre se mueve 2 casillas en una dirección (horizontal o vertical), y luego una casilla en la otra dirección.

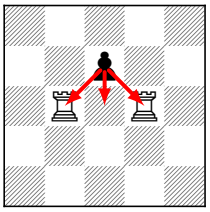
Siguiendo las reglas anteriores, las piezas pueden moverse libremente a cualquier casilla vacía pero sin pasar por encima de otra pieza, a excepción del caballo que puede saltar sobre otras piezas en su movimiento de "L". Adicionalmente, una pieza puede moverse a una casilla que contenga una pieza adversaria, en cuyo caso la pieza adversaria será *capturada* y removida del tablero. Si una pieza p puede capturar en un movimiento a la pieza q, diremos que la pieza p está *atacando* a la pieza q. A continuación mostramos algunos ejemplos de esto. En el primer ejemplo, el alfil en la casilla (3, 1) está atacando a la torre en la casilla (6, 4). En el segundo, el alfil no está atacando a la torre, pues el caballo en la casilla (5, 3) bloquea el movimiento. Finalmente, en el tercer ejemplo el caballo en la casilla (3, 4) está atacando al alfil en la casilla (5, 3), pues este puede saltar sobre otras piezas.



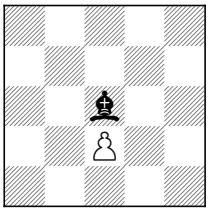




El peón tiene las reglas de movimientos más complejas del ajedrez. En primer lugar, a diferencia de las otras piezas que pueden avanzar o retroceder, el peón solo puede moverse hacia adelante; es decir, los peones negros solo pueden moverse hacia una casilla cuya fila tenga un índice mayor, mientras que los blancos solo pueden moverse a una casilla cuya fila tenga un índice menor. En segundo lugar, los peones se mueven de forma distinta dependiendo de si se mueven hacia una casilla vacía o si están capturando. Específicamente, un peón puede moverse una casilla hacia adelante si dicha casilla está vacía o puede capturar una pieza adversaria si la pieza está en cualquiera de las casillas diagonales en frente del peón.

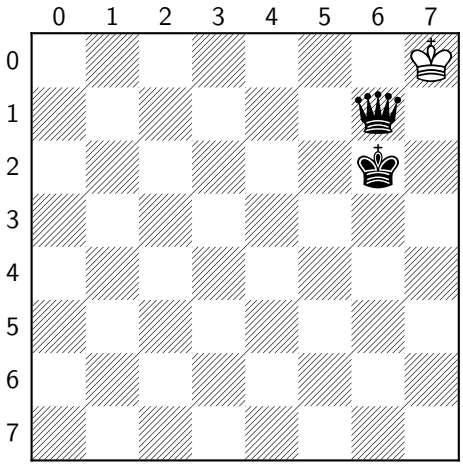


El peón negro puede moverse a la casilla en frente, pues está vacía. También puede moverse a alguna de las diagonales y capturar una de las torres.



No hay movimientos posibles para el peón, pues la casilla en frente está ocupada por el alfil negro y no hay piezas para capturar en las diagonales.

Finalmente, diremos que el rey está en *jaque* si hay una pieza atacándolo. Cuando un jugador tiene su rey en jaque, está obligado a sacarlo del jaque en su siguiente movimiento. Si no existe ningún movimiento con el cual es posible evadir el jaque, se dirá que la *posición* es un *jaque mate* y el jugador cuyo rey está en jaque perderá el juego. La siguiente figura muestra un tablero con una posición que es jaque mate. En primer lugar, el rey blanco está en jaque, pues la reina lo está atacando. Si el rey negro se mueve a las casillas (6,0) o (7, 1), seguirá siendo atacado por la reina y podrá ser capturado en el siguiente turno. Si se mueve a la casilla (6, 1) y captura a la reina, seguirá estando en jaque, pues el rey negro puede capturarlo moviéndose a esta casilla en el siguiente turno. Como no existen más movimientos posibles para el rey blanco, esta posición es jaque mate.



Dada la descripción del tablero, tu tarea es escribir un programa que determine si Elizabeth (piezas negras) tiene a su oponente en jaque mate (piezas blancas).

Nota En ajedrez, los peones también tienen reglas distintas cuando se mueven por primera vez. Además existe el *enroque*, la *coronación* y capturar *en passant*. Ninguna de estas reglas debe ser considerada en el contexto de este problema.

Entrada

La entrada comienza con una línea que contiene un entero n ($3 \leq n \leq 32$) que representa la cantidad de piezas en el tablero. Las siguientes n líneas representan cada una de las piezas en el tablero. Cada una de estas líneas contiene 4 enteros c, p, y y x.

El entero c representa el color de la pieza y será **0 si la pieza es blanca** o 1 **si la pieza es negra**.

El segundo entero p es el tipo de la pieza. Este será 0 **si es una torre,** 1 **si es alfil,** 2 **si es reina,** 3 **si es rey,** 4 **si es caballo o** 5 **si es peón**.

El valor y ($0 \leq y \leq 7$) corresponde al índice de la fila en la cual se encuentra ubicada la pieza. Finalmente, el valor x ($0 \leq x \leq 7$) corresponde al índice de la columna en que la pieza se encuentra ubicada.

La cantidad de piezas respetará las reglas descritas para el ajedrez. Es decir, la cantidad total de peones de cada color será menor o igual que 8, la de torres menor o igual que 2, la de caballos menor o igual que 2, la de alfiles menor o igual que 2 y la de reinas menor o igual que 1. Adicionalmente, habrá exactamente un rey negro y exactamente un rey blanco. Finalmente, el rey blanco siempre estará en jaque. Es decir, al menos una pieza de color negro estará atacando al rey blanco.

Salida

Tu programa debe imprimir una única línea con solo un 1 si Elizabeth (piezas negras) tiene a su oponente en jaque mate, o 0 en caso contrario.

Subtareas y puntaje

Subtarea 1 (20 puntos)

Se probará varios casos en los que Elizabeth (piezas negras) solo tiene caballos, peones y al rey; y el oponente (piezas blancas) tiene solo una pieza: el rey.

Subtarea 2 (30 puntos)

Se probará varios casos en que el jaque es múltiple, es decir, hay más de una pieza blanca que está atacando al rey blanco. Puede haber tanto piezas blancas como negras de cualquier tipo.

Subtarea 3 (50 puntos)

Se probará varios casos sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
3 0 3 0 7 1 2 1 6 1 3 2 6	1

Entrada de ejemplo	Salida de ejemplo
5 0 3 0 0 0 0 1 5 1 3 5 4 1 0 7 0 1 0 7 1	0