



Flechas fuera

Izan ha decidido que ya no le gustan las mates. Tras años de estudio, ya conoce a fondo todos los temas que le interesan y ya no le ve sentido a ser matemático. Pero claro, su sueño de ser rico no ha desaparecido. Habla con Joana a ver si a ella se le ocurre una nueva profesión para él. Ella, tras analizar sus habilidades, le recomienda que publique un juego para móviles.

A Izan le encanta la idea y se pasa una semana entera encerrado programando su primer videojuego: *Flechas fuera*. Consiste en, dado un tablero 2D de casillas con una ficha por casilla, ir eliminando estas fichas una a una hasta que el tablero quede vacío. Cada ficha tendrá una flecha en una de 4 direcciones (arriba, abajo, izquierda o derecha) y el jugador podrá eliminar una ficha si puede moverla en la dirección de la flecha y sacarla fuera del tablero **sin chocarse con otra**. Si no, **no podrá mover la ficha**. Pero claro, Izan se pone a generar muchos niveles y se da cuenta de que hay algunos que no tienen solución. ¿Serías capaz de, viendo un tablero del juego, decirle si se puede resolver ese nivel?

(Nota: el juego de este problema es similar —pero con diferencias fundamentales para la resolución del ejercicio— a *Tap away*, un videojuego conocido)

Entrada y salida

Recibirás una línea en la entrada con un entero T , el número de casos que has de procesar. Cada caso consistirá en:

- Una línea con un entero n : el alto y el ancho del tablero (es cuadrado)
- n líneas, cada una con una cadena de n caracteres: las flechas de cada fila del tablero: U para arriba, D para abajo, L para izquierda y R para derecha

Has de imprimir una línea por cada caso de entrada, con `POSIBLE` o `IMPOSIBLE` según corresponda.

Ejemplo

Entrada:

```
2
2
DR
UR
3
LUR
LRR
RRR
```

Salida:

```
IMPOSIBLE
POSIBLE
```



Explicación: El tablero del primer caso es imposible de resolver, ya que la flecha hacia abajo choca con la flecha hacia arriba y no hay forma de sacarlas a las dos.

El tablero del segundo caso se puede resolver. Basta con ir quitando una a una las flechas que se mueven hacia la derecha (empezando a quitarlas de derecha a izquierda, claro). Una vez hecho esto, nos quedarán dos hacia la izquierda con el camino despejado y una hacia arriba también ya al borde.

Restricciones

$$1 \leq T \leq 5000.$$

$$1 \leq n \leq 2000.$$

La suma de n^2 para todos los casos no superará 10^7 .

Subtareas

1. (14 puntos) Todos los tableros tendrán únicamente flechas de tipo L y R.
2. (16 puntos) Todos los tableros tendrán únicamente flechas de tipo L, U y R.
3. (5 puntos) $n = 2$.
4. (29 puntos) $n \leq 100$, la suma de n^3 para todos los casos no superará 10^9 .
5. (36 puntos) Sin restricciones adicionales.



Light Cycles

Light Cycles es un videojuego arcade de dos jugadores, donde cada jugador controla un coche. El Jugador 1 controla el coche amarillo, y actúa de cazador. El Jugador 2 controla el coche azul, y debe intentar escapar de su rival. A su paso, el coche amarillo va dejando una estela. Si el coche azul toca al coche amarillo o a la estela que va dejando en algún momento del juego, pierde. Si no, gana.

El mundo en el que se pueden mover es una cuadrícula, de la que los coches no pueden salir. Cada coche se mueve una casilla por turno. Primero se mueve el coche amarillo y luego el azul. Para el caso del coche amarillo, la estela que va dejando a su paso persiste k turnos. El juego dura c turnos en total.

Sabiendo el camino que han recorrido los dos coches, y la duración en turnos de la estela, ¿Podrías decirnos qué jugador gana?

Ten en cuenta que el juego no comprueba si el coche azul pierde hasta que no se haya movido al final del turno. Formalmente, si el coche amarillo visita la posición (x, y) en el turno t , entonces el coche azul pierde si visita la posición (x, y) en alguno de los turnos $t, t + 1, \dots, t + k$.

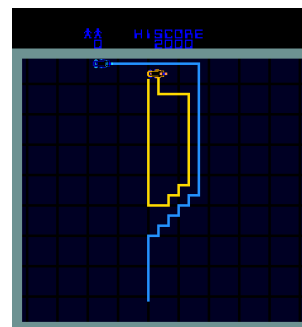


Figura 1: Imagen del videojuego real. Nótese que la estela azul no juega ningún papel en este problema.

Entrada y salida

La primera línea de la entrada contiene el número de casos T .

Por cada caso, la primera línea contiene dos enteros n y m , que representan el tamaño de la cuadrícula en la que habitan ambos coches. La segunda línea contiene otros dos números enteros, c y k .

A continuación, siguen c líneas, cada una indicando la posición de los coches en cada turno del juego, con 4 números. Los dos primeros, a_x y a_y , la casilla del coche amarillo; y los dos últimos, b_x y b_y , la casilla del coche azul.

Por cada caso debes imprimir una línea, con la palabra AMARILLO si gana el coche amarillo o AZUL si gana el coche azul.



Ejemplo

Entrada:

```
3
3 2
2 1
0 0 2 1
0 1 1 1
6 6
6 2
3 1 5 1
3 2 5 2
3 3 4 2
3 4 3 2
2 4 2 2
1 4 1 2
2 2
2 0
0 0 1 0
1 0 0 0
```

Salida:

```
AZUL
AMARILLO
AZUL
```

Explicación: En el primer caso, los coche azul y amarillo no colisionan. En el segundo caso, el coche azul colisiona con la estela del amarillo en el cuarto turno. En el tercer caso, no hay estela y el coche azul no está en el mismo turno en la misma posición que el coche amarillo (aunque se crucen).

Restricciones

$$1 \leq T \leq 1\,000.$$

$$2 \leq n, m \leq 1\,000.$$

$$0 \leq a_x, b_x < n, 0 \leq a_y, b_y < m.$$

En cada turno, cada coche se mueve a una casilla adyacente o se queda parado.

$$1 \leq c \leq 10^6, 0 \leq k \leq c.$$

La suma de $c + n \cdot m$ para todos los casos es como mucho $2 \cdot 10^6$.

Subtareas

1. (20 puntos) $k = 0$.
2. (21 puntos) La suma de c^2 para todos los casos es como mucho 10^7 .
3. (28 puntos) No se repiten los valores de (a_x, a_y) (el coche amarillo no pasa dos veces por el mismo sitio).
4. (31 puntos) Sin restricciones adicionales.



Múltiplo o Divisor

El juez tiene un número n oculto que debes determinar. Para ello, puedes hacer preguntas de los siguientes dos tipos:

1. Proporcionas un entero m y el juez te responde “Sí” si m es múltiplo de n o si m es divisor de n , y “No” en caso contrario. Esta pregunta tiene coste 1.
2. Proporcionas un entero m y el juez te responde un entero d que es la mínima distancia de m a alguno de los múltiplos o divisores de n , es decir, $d = \min_{x \in D_n \cup M_n} |m - x|$, donde D_n es el conjunto de divisores de n y M_n es el conjunto de múltiplos de n . La i -ésima vez que haces esta pregunta, la pregunta tiene coste i^2 .

Tu objetivo es determinar el número n sin que la suma de costes de tus preguntas sea demasiado grande.

Entrada y salida

Este es un problema interactivo. Debes refrescar la salida cada vez que imprimas datos (cout << endl o cout << flush en C++, System.out.flush() en Java, stdout.flush() en Python).

La primera línea de la entrada contiene el número de casos T . En cada caso, el juez tendrá un número n que deberás adivinar haciendo preguntas.

Para hacer una pregunta, debes imprimir una línea con el formato $? \ t \ m$, donde $1 \leq t \leq 2$ es el tipo de pregunta y $1 \leq m \leq 3 \cdot 10^{18}$ el parámetro de la pregunta. A continuación, se te responderá con una línea las palabras SI o NO en el caso de la primera pregunta, o con un entero d en el caso de la segunda pregunta. En caso de que realices una pregunta inválida o excedas el límite de coste, se te responderá con -1, y en ese caso tu programa debe terminar inmediatamente.

Para dar la respuesta, debes imprimir una línea con el formato $! \ n$, donde n es el valor que has determinado. A continuación, debes leer una respuesta con un carácter, que será:

- +, si se pasa al caso siguiente. En ese caso tu programa debe continuar con el nuevo caso.
- -, si no se pasa al caso siguiente, o bien porque ya se han acabado los casos o bien porque has dado una respuesta incorrecta. En ese caso, tu programa debe terminar inmediatamente.



Ejemplo

Juez	Programa	Comentario
2		$n = 2$ en el primer caso
	? 1 1	
SI		1 es divisor de 2
	? 2 3	
1		$1 = 3 - 2 = 3 - 4 $
	! 2	
+		$n = 6$ en el segundo caso
	? 2 8	
2		$2 = 8 - 6 $
	? 2 13	
1		$1 = 13 - 12 $
	? 2 4	
1		$1 = 4 - 3 $
	? 1 5	
NO		5 no es divisor ni múltiplo
	! 6	
-		

Restricciones

$$1 \leq T \leq 100.$$

$$1 \leq n \leq 10^{18}.$$

Los m de tus preguntas deben satisfacer $1 \leq m \leq 3 \cdot 10^{18}$.

Las preguntas que hagas pueden tener como máximo coste total 200 en cada uno de los casos.

El valor de n está fijo al comienzo de la interacción, es decir, no cambia adaptivamente según las preguntas que hagas.

Subtareas

1. (10 puntos) $n \leq 100$.
2. (20 puntos) $n \leq 2000$.
3. (30 puntos) $n \leq 10^9$.
4. (10 puntos) n es un número *regular*: no es divisible por primos diferentes de 2, 3 o 5.
5. (30 puntos) Sin restricciones adicionales.

Adicionalmente, la puntuación que obtienes en una subtarea depende del coste de las preguntas que hagas: para obtener una puntuación completa el coste debe ser como máximo 25 y para obtener una puntuación positiva como máximo 200. La puntuación de cada subtarea es multiplicada por un multiplicador $M(q)$, donde q es el máximo coste total de preguntas que has hecho en cada caso de esa subtarea. El valor de $M(q)$ viene dado por:

$$M(q) = \begin{cases} 0 & q > 200 \\ 0,5 + \frac{200-q}{750} & 200 \geq q > 50 \\ 0,7 + \frac{150-3q}{250} & 50 \geq q > 25 \\ 1,0 & 25 \geq q \end{cases}$$



Parejas Segedinenses

Decimos que una pareja de enteros (a, b) es *segedinense* si existen enteros positivos u, v, x, y tales que:

- $a = u + v$.
- $b = x + y$.
- u y v son divisores de b .
- x e y son divisores de a .

Dados a y b , determina si forman una pareja segedinense.

Entrada y salida

La primera línea de la entrada contiene el número de casos T .

Por cada caso habrá una líneas de entrada con dos enteros a, b .

Por cada caso, debes imprimir una línea con SI si (a, b) es una pareja segedinense y NO en caso contrario.

Ejemplo

Entrada:

```
5
4 6
1 2
9 9
12 10
5 4
```

Salida:

```
SI
NO
NO
SI
NO
```

Explicación:

En el primer caso, se puede escoger $(u, v, x, y) = (2, 2, 2, 4)$.

En el cuarto caso, se puede escoger $(u, v, x, y) = (10, 2, 6, 4)$.



Restricciones

$$1 \leq T \leq 100\,000.$$

$$1 \leq a, b \leq 10^{18}.$$

Subtareas

1. (11 puntos) $a, b \leq 50$, $T \leq 2\,500$.
2. (18 puntos) $a, b \leq 10\,000$.
3. (24 puntos) $a, b \leq 10^7$.
4. (17 puntos) $a = b$.
5. (30 puntos) Sin restricciones adicionales.



Sueño

Las personas, a la hora de irse a dormir, pueden tener problemas para conciliar el sueño recordando sus problemas o acontecimientos sucedidos a lo largo del día. Manuel, en cambio, suele acostarse pensando en problemas de programación y no se duerme hasta resolverlos.

Hoy se ha planteado el siguiente problema: “Dados n y x , calcula el número de vectores de tamaño n formados de enteros en el rango $[1, x]$ de forma que tenga i máximos absolutos para todo $i = 1, 2, \dots, n$. Como este número puede ser muy grande, imprímelo módulo **202220233**”. Por ejemplo con el vector $v = [2, 5, 3, 4, 5]$ tenemos dos máximos absolutos en las posiciones 2 y 5, y este además está formado por enteros en el rango $[2, 5]$. Dos vectores u y v se consideran diferentes si existe un i tal que $v_i \neq u_i$.

Mañana es un día muy importante para Manuel y necesita descansar. ¿Puedes resolver su problema para que así pueda dormir?

Entrada y salida

La primera línea contiene un número T , el número de casos.

Las siguientes T líneas contienen dos enteros n y x cada una.

Para cada caso, debes imprimir una línea compuesta por n enteros, donde el i -ésimo ($1 \leq i \leq n$) valor representa el número de vectores de tamaño n con i máximos absolutos módulo **202220233**.

Ejemplo

Entrada:

```
2
2 2
2 4
```

Salida:

```
2 2
12 4
```

Explicación: En el primer caso, encontramos los vectores $[1, 2]$, $[2, 1]$ con un solo máximo absoluto y los vectores $[1, 1]$ y $[2, 2]$ con dos máximos absolutos.



Restricciones

$$1 \leq T \leq 1000.$$

$1 \leq n \leq 1000$, la suma de n en todos los casos es menor o igual que 1000.

$$2 \leq x \leq 10^9.$$

Subtareas

1. (6 puntos) $n, x \leq 6$.
2. (7 puntos) $n = 2$.
3. (12 puntos) $x = 2$.
4. (14 puntos) $x \leq 1000$.
5. (23 puntos) x es una potencia de dos.
6. (38 puntos) Sin restricciones adicionales.