



Juego de cartas

La entrega de premios se ha retrasado y María y tú os aburrís, así que decidís jugar a un juego. Convenientemente, María ha traído su baraja de cartas *sevillana*, que consiste en n cartas, una carta con cada número del 1 al $n - 2$ y dos cartas con el número $n - 1$.

María pondrá las cartas sobre la mesa boca abajo en cierto orden y tú tendrás que adivinar el orden haciendo preguntas. En cada pregunta le puedes decir la posición de dos cartas distintas, y María te dirá cuál es el menor valor entre los valores de esas dos cartas.

Debes adivinar cuál es el orden de las cartas sin hacer demasiadas preguntas.

Interacción

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 entero n . **Debes leer este número** antes de hacer ninguna pregunta.

Para hacer una pregunta debes escribir una línea con el formato $? \ a \ b$, donde $1 \leq a, b \leq n$ son los dos números distintos que corresponden a las posiciones de las dos cartas por las que preguntas.

A continuación el juez te responderá con una línea que contiene un único entero, la respuesta a tu pregunta.

En caso de que hayas superado el límite máximo de preguntas, recibirás como respuesta el número -1 . Si lees -1 , **tu programa debe terminar inmediatamente** para obtener el veredicto “Respuesta incorrecta” en CMS. De otra forma, tu programa podría obtener veredictos arbitrarios.

Cuando hayas descubierto el orden de las cartas, deberás imprimir una sola línea, con el formato $! \ c_1 \ c_2 \ \dots \ c_n$, donde c_i es el valor de la carta en la posición i .

Ejemplo

A continuación hay un ejemplo de cómo podría ser una interacción entre el juez y tu programa en un caso en el que $n = 4$, $(c_1, c_2, c_3, c_4) = (3, 3, 1, 2)$. La primera columna indica lo que imprime el juez, que tu programa debe leer por entrada estándar; la segunda, lo que imprime tu programa en salida estándar, y la tercera es una explicación del significado de lo que imprime el juez.

Juez	Programa	Comentario
4		$n = 4$
	? 1 2	
3		$\min(c_1, c_2) = 3$
	? 1 3	
1		$\min(c_1, c_3) = 1$
	? 2 4	
2		$\min(c_2, c_4) = 2$
	! 3 3 1 2	



Restricciones

$$3 \leq n \leq 2000$$

$c_1, \dots, c_n$ corresponde a una baraja de cartas *sevillana*: es decir, hay es una permutación de $(1, 2, \dots, n-2, n-1, n-1)$.

Puedes hacer como máximo 10^5 preguntas.

El interactor **no** es adaptativo, es decir, el orden de las cartas está fijado al comienzo de la interacción en cada caso de prueba.

Subtareas

1. (8 puntos) $n \leq 50$.
2. (92 puntos) Sin restricciones adicionales.

Adicionalmente, la puntuación que obtienes en una subtarea depende del número de preguntas que hagas: para obtener una puntuación completa debes realizar como máximo 2030 preguntas y para obtener una puntuación positiva debes realizar como máximo 10^5 preguntas. La puntuación de cada subtarea es multiplicada por un multiplicador $M(q)$, donde q es el máximo número de preguntas que has hecho en los casos de esa subtarea. El valor de $M(q)$ viene dado por:

$$M(q) = \begin{cases} 0 & q > 10^5 \\ 0,25 + \frac{100000-q}{900000} & 10000 < q \leq 10^5 \\ 0,1 + \frac{2500}{q} & 4000 < q \leq 10000 \\ 0,725 + \frac{11 \cdot (4000-q)}{78800} & 2030 < q \leq 4000 \\ 1 & q \leq 2030 \end{cases}$$



Igualando

Se nos da un array a de n enteros y los enteros k, m, r (donde $m > r \cdot (n - 1)$). Una *operación* consiste en realizar las siguientes modificaciones para un índice $1 \leq i \leq n$:

- Sumar m a a_i
- Restar r a todos los elementos de a excepto a_i , es decir, a $a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n$.

Podemos realizar la operación anterior varias veces, para cualquier índice $1 \leq i \leq n$. No hay problema en que algunos elementos se vuelvan números negativos después de una operación.

Encuentra el mínimo número de operaciones que se necesitan realizar para que todos los elementos de a sean mayores o iguales que k .

Entrada y salida

La primera línea contiene T , el número de casos.

Cada caso empieza con una línea con cuatro enteros, n, k, m y r .

La siguiente línea de cada caso contiene n enteros $a_1, \dots, a_n$.

Para cada caso se debe imprimir una línea con el número mínimo de operaciones de forma que se pueda conseguir que todos los elementos de a sean mayores o iguales que k .

Ejemplo

Entrada:

```
2
3 2 3 1
1 1 4
3 2 3 1
2 2 2
```

Salida:

```
2
0
```

En el primer caso aplicamos la operación una vez a a_1 , obteniendo el vector $a = [a_1 + m, a_2 - r, a_3 - r] = [1 + 3, 1 - 1, 4 - 1] = [4, 0, 3]$, aplicamos también la operación sobre a_2 , obteniendo $[3, 3, 2]$, que cumple la condición de que todos los elementos son mayores o iguales que $k = 2$.

En el segundo caso no aplicamos la condición ya que el $a_1, a_2, a_3 \geq k$.

Restricciones

$$1 \leq T \leq 10^3.$$

$$1 \leq n \leq 2 \cdot 10^3.$$

$$1 \leq k, m, r, a_i \leq 10^6.$$

$$m > r \cdot (n - 1).$$

La suma de n para todos los casos es como mucho $2 \cdot 10^3$.



Subtareas

1. (21 puntos) La suma sobre todos los casos de $k \cdot n^2$ es menor o igual a 10^5 .
2. (23 puntos) $k \leq 5 \cdot 10^3$.
3. (56 puntos) Sin restricciones adicionales.



Rectángulo

El juez tiene un rectángulo oculto, cuyas dimensiones $a \times b$ debes determinar, donde a y b son números enteros. Para ello, puedes hacer preguntas del siguiente tipo: das dos enteros c y d y se te responde si un rectángulo $c \times d$ cabe dentro del rectángulo $a \times b$.

Por ejemplo, si $(a, b) = (18, 12)$ y preguntas por $(c, d) = (19, 3)$, el juez responderá sí, ya que un rectángulo 19×3 cabe dentro de uno 18×12 , tal como se muestra en la figura.

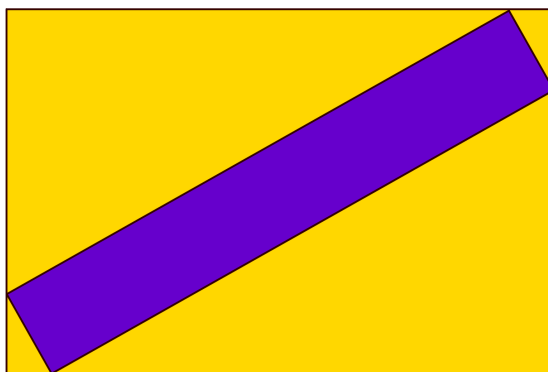


Figura 1: Un rectángulo 19×3 dentro de uno 18×12 .

Interacción

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)

Para hacer una pregunta, debes imprimir una línea con el formato `? c d`, donde $1 \leq c, d \leq 200$ son los lados del rectángulo por el que preguntas. Para recibir la respuesta, debes leer una línea, que será 1 si el rectángulo $c \times d$ cabe dentro del rectángulo $a \times b$, o 0 en caso contrario.

Para dar la respuesta, debes imprimir una línea con el formato `! a b`, donde a, b son las dimensiones del rectángulo oculto. Da igual en qué orden escribas los números a y b . Una vez hayas dado la respuesta, **tu programa debe terminar inmediatamente**, sin hacer más preguntas.

En caso de que hayas superado el límite máximo de preguntas o que hayas hecho una pregunta inválida, recibirás como respuesta el número -1. Si lees -1, **tu programa debe terminar inmediatamente** para obtener el veredicto “Respuesta incorrecta” en CMS. De otra forma, tu programa podría obtener veredictos arbitrarios.

Ejemplo

A continuación hay un ejemplo de cómo podría ser una interacción entre el juez y tu programa en un caso en el que $(a, b) = (18, 12)$. La primera columna indica lo que imprime el juez, que tu programa debe leer por entrada estándar; la segunda, lo que imprime tu programa en salida estándar, y la tercera es una explicación del significado de lo que imprime el juez.



Juez	Programa	Comentario
	? 19 3	
1		19×3 cabe dentro de 18×12 .
	? 12 18	
1		12×18 cabe dentro de 18×12 (es igual).
	? 19 4	
0		19×4 no cabe dentro de 18×12 .
	? 3 20	
0		3×20 no cabe dentro de 18×12 .
	! 12 18	Tu programa da la respuesta correctamente.

Restricciones

$1 \leq a, b \leq 100$.

Los enteros c y d de tus preguntas deben satisfacer $1 \leq c, d \leq 200$.

Puedes hacer como máximo 20 preguntas. Dar la respuesta **no** cuenta como pregunta para el límite.

El interactor **no** es adaptativo, es decir, los valores de a y b están fijados al comienzo de la interacción en cada caso de prueba.

Subtareas

1. (13 puntos) $a, b \leq 5$.
2. (12 puntos) $a, b \leq 6$.
3. (18 puntos) $a, b \leq 7$.
4. (15 puntos) $\min(a, b) = 1$.
5. (16 puntos) $a = b$.
6. (26 puntos) Sin restricciones adicionales.



Resto

Dado un array a de tamaño n , consideramos

$$f(x) = (x \% a_1) + ((x \% a_1) \% a_2) + (((x \% a_1) \% a_2) \% a_3) + \dots + (((\dots (x \% a_1) \dots) \% a_n)$$

.

Donde definimos $x \% y$ como el resto de la división entera de x entre y .

Tenemos también Q preguntas de dos tipos.

1. Si la i -ésima pregunta es de tipo 1, recibiremos el entero q_i y deberemos responder con $f(q_i)$.
2. Si la i -ésima pregunta es del tipo 2, recibiremos el entero p y el entero x y tendremos que actualizar el valor del p -ésimo elemento a x ($a_p = x$).

Entrada y salida

La primera línea contiene T , el número de casos.

Cada caso empieza con dos enteros, n y Q .

La siguiente línea de cada caso contiene n enteros $a_1, \dots, a_n$.

Las siguientes Q líneas de cada caso contienen las preguntas. Cada línea comienza con un entero que representa el tipo de la pregunta (1 o 2), seguida de un entero q_i si la pregunta es de tipo 1 o de dos enteros p, x si es de tipo 2.

Para cada pregunta del tipo 1, debemos responder con una única línea con $f(q_i)$.

Ejemplo

Entrada:

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

Salida:

```
1
0
7
5
```

El el primer caso, la secuencia de valores generada por $f(q_1) = f(3)$ es: $[3 \% 2, (3 \% 2) \% 1, ((3 \% 2) \% 1) \% 1] = [1, 0, 0]$, y su suma es 1. Para $f(q_2)$, la secuencia de valores es: $[4 \% 2, (4 \% 2) \% 1, ((4 \% 2) \% 1) \% 1] = [0, 0, 0]$ con suma 0.



En el segundo caso, la secuencia de valores generada por $f(q_1) = f(5)$ es: $[5 \% 6, (5 \% 6) \% 4, ((5 \% 6) \% 4) \% 2] = [5, 1, 1]$ con suma 7.

Después tenemos una actualización del segundo valor ($a_2 = 5$), y cuando volvemos a realizar la pregunta $f(5)$, el resultado pasa a ser 5.

Restricciones

$$1 \leq T \leq 10^5.$$

$$1 \leq n, Q \leq 10^5.$$

La suma de n para todos los casos es como mucho 10^5 .

La suma de Q para todos los casos es como mucho 10^5 .

$$1 \leq a_i, q_i \leq 10^{11}.$$

Para cada pregunta del tipo 2, $1 \leq p \leq n$, $1 \leq x \leq 10^{11}$.

Subtareas

1. (5 puntos) La suma de n y la suma de Q sobre todos los casos es menor o igual que 10^3 .
2. (33 puntos) La suma de a_i sobre todos los casos es menor o igual que 10^6 y todas las preguntas son del tipo 1.
3. (27 puntos) Todas las preguntas son del tipo 1.
4. (35 puntos) Sin restricciones adicionales.



XORdenamiento

Se da una lista de n enteros de 0 a $2^{20} - 1$.

Se puede realizar la siguiente operación cualquier cantidad de veces (cero o más).

Elegir dos posiciones **distintas** $1 \leq x, y \leq n$ y asignar:

$$a_x := a_x \text{ XOR } a_y$$

Donde **XOR** es el XOR bit a bit de los dos números.

Hay que realizar estas operaciones para que la lista resultante esté ordenada de forma no decreciente (es decir, $a_i \leq a_{i+1}$ para cada posición i de 1 a $n - 1$).

¿Puedes construir una secuencia de operaciones suficientemente corta para solucionar esta tarea?

Entrada y salida

La primera línea contiene el número n , el tamaño de la lista.

La segunda línea contiene n enteros $a_1, \dots, a_n$.

En la primera línea, imprime el número de operaciones m .

En las siguientes m líneas, imprime las posiciones x y y .

Ejemplo

Entrada:

```
5
0 1 1 1 2
```

Salida:

```
0
```

Entrada:

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

Salida:

```
5
5 1
1 5
2 3
2 4
4 3
```



Restricciones

$n = 1000$ en todos los casos.

Los ejemplos muestran casos con $n < 1000$ solamente para ayudarte. No aparecerán en los casos de prueba reales.

$0 \leq a_i \leq 2^{20} - 1$.

Puedes hacer como máximo 21000 operaciones.

Subtareas

1. (100 puntos) Sin restricciones adicionales.

Adicionalmente, la puntuación que obtienes depende del número de operaciones que hagas: para obtener una puntuación completa debes realizar como máximo 2500 operaciones y para obtener una puntuación positiva debes realizar como máximo 21000 operaciones. La puntuación de cada subtarea es multiplicada por un multiplicador $M(m)$, donde m es el máximo número de operaciones que has hecho en los casos de esa subtarea. El valor de $M(m)$ viene dado por:

$$M(m) = \begin{cases} 0 & m > 21000 \\ 0,1 + \frac{21000-m}{90000} & 21000 \geq m > 3000 \\ 0,3 + \frac{21000-7m}{5000} & 3000 \geq m > 2500 \\ 1,0 & 2500 \geq m \end{cases}$$