Mínimo XOR

Hay un conjunto de n enteros no negativos $x_1, \ldots, x_n$ que tienes que determinar.

Para determinarlo, puedes hacer preguntas del siguiente tipo: proporcionas un entero no negativo y y se te devuelve el valor $\min_{i=1,\ldots,n} (x_i \oplus y)$, donde $\oplus$ indica la operación disyuntiva exclusiva bit a bit (XOR), que definimos abajo. Debes encontrar los valores de los n números usando como mucho 20000 preguntas.

La operación XOR de dos enteros no negativos se calcula de la siguiente forma: se escriben los dos números en binario y el resultado es el número que escrito en binario tiene 1 en las posiciones en las que uno y solo uno de los dos operandos tiene un 1, y 0 en las posiciones en la que los dos operandos tienen 0 o los dos operandos tienen 1 (dicho de otra manera: se pone un 0 en las posiciones que los dos operandos tienen el mismo dígito binario y 1 en las que los dígitos son diferentes).

Por ejemplo, $26 \oplus 14 = 20$, porque 26 en binario es 11010, 14 en binario es 1110, y el número resultante de poner 1 en las posiciones en las que sólo uno de los dos operandos tiene un 1 es 10100, que es 20 en binario.

$$\begin{array}{r} 11010 \\ \oplus\, 01110 \\ \hline 10100 \end{array}$$

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 un entero n, el número de elementos del conjunto. Debes leer este valor antes de hacer ninguna pregunta.

Para hacer una pregunta debes escribir una línea con el formato `? y`, donde y es un entero que debe satisfacer $0 \leq y < 2^{30}$. Después de hacer una pregunta, debes leer de la entrada un entero, el resultado. En caso de que hagas una pregunta inválida o superes el límite de preguntas el resultado que leerás será -1, si tu programa lee un -1 debería terminar inmediatamente.

Una vez hayas determinado los n números, debes escribir una línea con el formato `! x_1 x_2 ... x_n`, donde $x_1 < x_2 < \ldots < x_n$ son los n enteros del conjunto **en orden ascendente**. Después de escribir esto, tu programa debería terminar.

XXV Olimpiada Informática Española
Final día 2
`xor`

Ejemplo

Entrada:

```
3
1
1
0
1
```

Salida:

```
? 1
? 2
? 3
? 4
! 0 3 5
```

Explicación: En este caso los valores ocultos son $0, 3, 5$. La primera línea de la entrada es n, el número de valores. A continuación se hace una pregunta con $y = 1$: el interactor calcula los tres valores $0 \oplus 1 = 1, 3 \oplus 1 = 2, 5 \oplus 1 = 4$ y da como respuesta en la entrada el valor más pequeño, que es 1. En la pregunta con $y = 2$, se calcula $0 \oplus 2 = 2, 3 \oplus 2 = 1, 5 \oplus 2 = 7$ y se retorna el valor más pequeño, que es 1. Con $y = 3$, $1 \oplus 3 = 2, 3 \oplus 3 = 0, 5 \oplus 3 = 6$ y se responde 0. Con $y = 4$, $1 \oplus 4 = 5, 3 \oplus 4 = 7, 5 \oplus 4 = 1$ y se responde 1. Finalmente, se dan los tres números tal como se especifica en el formato.

Restricciones

$1 \leq n \leq 10000$.

$0 \leq x_i < 2^{30}$. Los n valores de x_i son números distintos.

Se pueden hacer como máximo 20000 preguntas (dar la respuesta no cuenta como pregunta).

Subtareas

1. (7 puntos) $x_i < 20000$.
2. (13 puntos) $x_i < 2^{15}$.
3. (8 puntos) $n = 2$.
4. (20 puntos) $n \leq 300$.
5. (12 puntos) $n \leq 600$.
6. (20 puntos) $n \leq 5000$.
7. (20 puntos) Sin restricciones adicionales.