The power of an integer x is defined as the number of steps needed to transform x into 1 using the following steps:

* if x is even then x = x / 2
* if x is odd then x = 3 \* x + 1

For example, the power of x = 3 is 7 because 3 needs 7 steps to become 1 (3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 --> 1).

Given three integers lo, hi and k. The task is to sort all integers in the interval [lo, hi] by the power value in **ascending order**, if two or more integers have **the same** power value sort them by **ascending order**.

Return the k-th integer in the range [lo, hi] sorted by the power value.

Notice that for any integer x (lo <= x <= hi) it is **guaranteed** that x will transform into 1 using these steps and that the power of x is will **fit** in 32 bit signed integer.

**Example 1:**

**Input:** lo = 12, hi = 15, k = 2

**Output:** 13

**Explanation:** The power of 12 is 9 (12 --> 6 --> 3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 --> 1)

The power of 13 is 9

The power of 14 is 17

The power of 15 is 17

The interval sorted by the power value [12,13,14,15]. For k = 2 answer is the second element which is 13.

Notice that 12 and 13 have the same power value and we sorted them in ascending order. Same for 14 and 15.

**Example 2:**

**Input:** lo = 1, hi = 1, k = 1

**Output:** 1

**Example 3:**

**Input:** lo = 7, hi = 11, k = 4

**Output:** 7

**Explanation:** The power array corresponding to the interval [7, 8, 9, 10, 11] is [16, 3, 19, 6, 14].

The interval sorted by power is [8, 10, 11, 7, 9].

The fourth number in the sorted array is 7.

**Example 4:**

**Input:** lo = 10, hi = 20, k = 5

**Output:** 13

**Example 5:**

**Input:** lo = 1, hi = 1000, k = 777

**Output:** 570

**Constraints:**

* 1 <= lo <= hi <= 1000
* 1 <= k <= hi - lo + 1