Task3:

The Invariant: x * y = k * n + res

Step 1: checked if the formula works before the loop starts.

At the beginning, k = x, n = y, and res = 0.

If we plug these in:

x * y = x * y + 0
This is obviously true.

Step 2: We assume the formula is true at the start of the loop. We need to show it stays true after we change the variables.

- **When k is even:**
  We just cut k in half and double s.
  Mathematically, (k/2) * (2n) is the same as k * n.
  Since res doesn't change here, the total stays the same.
- **When k is odd:**
  Since k is odd, when we divide by integer 2, we lose a value. So we handle that first by adding n to res.
  After that, we divide k by 2 and multiply n by 2.
  Because we moved that extra value into res, the equation (k*n) + res still equals x*y.

Step 3:

The loop only stops when k becomes 0.

Using the invariant formula:

x * y = 0 * n + res
This simplifies to x * y = res.

This means res holds the correct answer, so the code is correct.

Task4:

Precondition: The array A exists.

Post-condition: The function returns the largest index i where A[i] == x, or -1 if x is not in the array.

The Loop Invariant:

For every index k such that i < k < n, A[k] != x.

//scanned the right side of i and verified x isn't there.

A. Initialization

At the start, i = n - 1.

The range of "indices greater than i" is empty (there are no indices greater than n-1).

Therefore, the statement "all values to the right of i are not x" is true.

B. Maintenance (Inside the loop)

In each step, we check A[i].

- **If A[i] == x:**
  We return i. Since the invariant told us that no index *greater* than i holds x, finding x at i guarantees that i is the **last** occurrence. This satisfies the post-condition immediately.
- **If A[i] != x:**
  The invariant previously said x is not at indices i+1, i+2, . . .
  Now we checked index i and it is also not x.
  So, x is not at indices i, i+1, i+2, . . .
  We decrement i to i-1. The statement "x is not to the right of the new i" holds true.

C. Termination

The loop stops when i = -1.

Using our invariant one last time: "For all k > -1, A[k] != x."

Since indices 0, 1, . . . , n-1 are all > -1, this means x is nowhere in the array.

The function returns -1, which is the correct answer.