Notes on proving binary search is correct.

Assume $A[1..n]$ is a constant array of elements from some totally ordered domain D.

Here is one version of binary search.

```
1  BINSEARCH(k, lo, hi)
2      pre-conditions: 1 ≤ lo ≤ hi ≤ n and
3          A[i] ≤ A[i + 1] for 1 ≤ i < n
4      post-conditions: returns an i with lo ≤ i ≤ hi and A[i] = k if such an i exists;
5          otherwise returns "not found"
6      if lo = hi then
7          if A[lo] = k then return lo
8          else return "not found"
9      else
10         mid ← (lo+hi)/2
11         if A[mid] > k then return BINSEARCH(k, lo, mid − 1)
12         else return BINSEARCH(k, mid, hi)
13     end if
14 end BINSEARCH
```

We define the size of an input to this algorithm to be $hi - lo$.

Theorem 1. *For all $s \geq 0$, if* BINSEARCH *is called on an input of size s that satisfies the preconditions, then it satisfies the postconditions.*

Proof. (by induction on s).

Base case ($s = 0$): If the input has $hi - lo = 0$, then $hi = lo$, so the algorithm executes line 7–8.

If there is an i with $lo \leq i \leq hi$ and $A[i] = k$, then $i = lo$ and $A[i] = k$, so the algorithm will output lo, which satisfies the postconditions.

Otherwise, $A[i] \neq k$ and the algorithm outputs "not found", which also satisfies the postconditions.

Inductive step: Let $s \geq 1$. Assume the claim holds for inputs when $0 \leq hi - lo < s$. Goal: prove the claim holds for inputs with $hi - lo = s$.

Consider any input that has $hi - lo = s$ (*) and satisfies the preconditions (**).

Then, $hi - lo = s \geq 1$, so $lo < hi$, and the algorithm executes line 10–12. We want to apply the induction hypothesis to the recursive calls made on line 11 and 12. If we can show the size of these recursive calls are between 0 and $s - 1$ AND that they satisfy the preconditions, then the induction hypothesis will tell us that those recursive calls satisfy their postconditions.

We have:

$$mid = \frac{lo + hi}{2} \geq \frac{lo + lo + 1}{2} = lo + 1 \tag{1}$$

and

$$mid = \frac{lo + hi}{2} \leq \frac{hi + hi}{2} = hi. \tag{2}$$

For the recursive call on line 11, we have $lo' = lo$, $hi' = mid - 1$. So,

$$\begin{aligned} hi' - lo' &= mid - 1 - lo \\ &\geq lo + 1 - 1 - lo \qquad \text{(by (1))} \end{aligned}$$

$$
\begin{aligned}
&= 0, \\
\text{and } hi' - lo' &= mid - 1 - lo \\
&\le hi - lo - 1 \qquad \text{(by (2))} \\
&= s - 1.
\end{aligned}
$$

By (1), (2) and (**), we have

$$1 \le lo \le mid - 1 \le hi - 1 \le n$$

$$\Rightarrow 1 \le lo' \le hi' \le n.$$

Hence, the recursive call on line 11 satisfies its preconditions and its size is between 0 and $s - 1$. By the induction hypothesis, it satisfies its postconditions.

For the recursive call on line 12, we have $lo'' = mid$ and $hi'' = hi$. So,

$$
\begin{aligned}
hi'' - lo'' &= hi - mid \\
&\ge 0 \qquad \text{(by (2))}, \\
\text{and } hi'' - lo'' &= hi - mid \\
&\le hi - (lo + 1) \qquad \text{(by (1))} \\
&= s - 1
\end{aligned}
$$

By (1), (2) and (**), we have

$$1 \le lo \le mid \le hi \le n$$

$$\Rightarrow 1 \le lo'' \le hi'' \le n.$$

Hence, the recursive call on line 12 satisfies its preconditions and its size is between 0 and $s - 1$. By the induction hypothesis, it satisfies its postconditions.

Now that we know the recursive call on line 11 or 12 satisfies its postconditions, we can prove that the main call satisfies its postconditions.

Case 1 (k does not appear in $A[lo..hi]$): We must prove the algorithm returns "not found". Since $lo \le lo'$ and $hi' \le hi$ and $lo \le lo''$ and $hi \le hi''$, we know that k appears in neither $A[lo'..hi']$ nor $A[lo''..hi'']$, so whichever recursive call is made will output "not found", and so will the main call.

Case 2 (k appears in $A[lo..hi]$ and $A[mid] > k$): Since A is sorted, k cannot appear in $A[mid..hi]$, so it must appear in $A[lo..mid - 1]$. The algorithm executes the recursive call on line 11. Since this call satisfies its postconditions, it (and the main call) will output an i between lo' and hi' (and hence between lo and hi) such that $A[i] = k$.

Case 3 (k appears in $A[lo..hi]$ and $A[mid] \le k$): Since A is sorted, k must appear in $A[mid..hi]$. The algorithm executes the recursive call on line 12. Since this call satisfies its postconditions, it (and the main call) will output an i between lo'' and hi'' (and hence between lo and hi) such that $A[i] = k$. □