Consider the following algorithm (written in pseudo-code, where “←” represents assignment).

```plaintext
# Precondition: A is a non-empty list of integers (i.e., len(A) > 0) sorted in non-decreasing order
# (i.e., A[0] ≤ A[1] ≤ . . . ≤ A[len(A) − 1]) and x is an integer that occurs in A (i.e.,
# ∃i ∈ {0, 1, . . . , len(A) − 1}, A[i] = x).
first ← 0
last ← len(A) − 1
# Loop Invariant: 0 ≤ first ≤ last < len(A) and x occurs in A[first . . . last]
# (i.e., ∃i ∈ {first, . . . , last}, A[i] = x).
while first < last:
    midpoint ← ⌊(first + last)/2⌋
    if A[midpoint] < x:
        first ← midpoint + 1
    else:
        last ← midpoint
    index ← first
# Postcondition: 0 ≤ index < len(A) and A[index] = x.
```

1. Write a detailed argument that the loop invariant holds just before the loop condition is evaluated for the first time, under the assumption that the precondition is true.

2. State clearly the property that must be satisfied by the loop invariant (in order to actually be a loop invariant). Do NOT argue that this property is satisfied! For this question, we only want you to say what the property is.

3. Assuming that the loop invariant is correct, i.e., that it satisfies the property you stated in the previous question, give a detailed argument that the postcondition will be satisfied once the loop terminates. (Again, do NOT try to prove that the loop invariant is correct. Make sure you understand clearly what this means, and how it is different from what this question is asking.)

Bonus: In order to prove that the algorithm is correct, there is one important property that must be shown (in addition to proving that the loop invariant is correct and that the postcondition holds at the end of the loop). State this property clearly, and give a detailed argument that it is true.