Recall that a **precondition** is a condition that is assumed to be true **before** a set of instructions are executed, a **postcondition** is a condition that is assumed to be true **after** a set of instructions have been executed, and a **loop invariant** is a condition between variables that is always true at the start and at the end of a loop iteration. Another way to say this is that the **loop invariant** is a condition that must be true every time the program evaluates the loop condition.

Now consider the following algorithm (written in pseudo-code, where "=" represents assignment).

```
# 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 shows 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. Assuming that the loop invariant is correct, write a detailed argument that shows that the postcondition will be satisfied once the loop terminates.

3. Write a detailed argument that shows that the loop invariant is correct. That is, show that the loop invariant is true each time the program evaluates the loop condition.

4. 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 then write a detailed argument that it is true.