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.

   Assume that the precondition holds: A is a list of integers and $\text{len}(A) > 0$ and $A[0] \leqslant \ldots \leqslant A[\text{len}(A) - 1]$ and $\exists i \in \{0, \ldots, \text{len}(A) - 1\}, A[i] = x$.

   Then just before the loop condition is evaluated for the first time, we know $\text{first} = 0$ and $\text{last} = \text{len}(A) - 1$ (by the first two statements in the algorithm).
   Then $0 \leqslant 0 = \text{first}$.
   Then $\text{first} = 0 \leqslant \text{len}(A) - 1 = \text{last}$ (because the precondition implies $\text{len}(A) > 0$, which implies $\text{len}(A) \geqslant 1$, which implies $\text{len}(A) - 1 \geqslant 0$).
   Then $\text{last} = \text{len}(A) - 1 < \text{len}(A)$.
   Then x occurs in $A[\text{first} \ldots \text{last}] = A[0 \ldots \text{len}(A) - 1]$ (by the precondition).

   Then, if the precondition holds, the loop invariant also holds just before the loop condition is evaluated for the first time.

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

   Assume that the loop invariant holds: $0 \leqslant \text{first} \leqslant \text{last} < \text{len}(A)$ and $\exists i \in \{\text{first}, \ldots, \text{last}\}, A[i] = x$.

   Assume that the loop terminates, *i.e.*, the loop condition evaluates to False: $\text{first} \not< \text{last}$.
   Then, $\text{first} = \text{last}$ (because $\text{first} \leqslant \text{last}$ by the loop invariant but $\text{first} \not< \text{last}$ by the negation of the loop condition).
   Then $\text{index} = \text{first} = \text{last}$ (after the last line of code is executed).
   Then $0 < \text{index}$ (because $0 \leqslant \text{first}$ by the loop invariant).
   Then $\text{index} < \text{len}(A)$ (because $\text{last} < \text{len}(A)$ by the loop invariant).
   Then $A[\text{index}] = x$ (because $\exists i \in \{\text{first}, \ldots, \text{last}\}, A[i] = x$ and $\text{first} = \text{last}$ implies $A[\text{first}] = x$).

Then, if the loop terminates, the postcondition holds.

Then, if the loop invariant holds and the loop terminates, the postcondition holds.

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.

   Notation: To simplify from the notation used in the course notes, we will use a "prime symbol" to represent the values of variables at the end of one iteration of the loop (when those values are different from the ones at the beginning of the iteration). For example, "first′" represents the value of first at the end of the iteration, while "first" represents the value at the beginning.

   Assume that the precondition holds.

   Assume the loop invariant is true (*i.e.*, $0 \leqslant \text{first} \leqslant \text{last} < \text{len}(A)$ and $\exists i \in \{\text{first}, \ldots, \text{last}\}, A[i] = x$) and the loop carries out at least one more iteration.

   Then first $<$ last. # by the loop condition

   Then midpoint $= \lfloor(\text{first} + \text{last})/2\rfloor > (\text{first}+\text{last})/2-1 > (\text{first}+\text{first})/2-1 = \text{first}-1$ and midpoint $= \lfloor(\text{first} + \text{last})/2\rfloor < \lfloor(\text{last} + \text{last})/2\rfloor = \text{last}$. So first $\leqslant$ midpoint $<$ last.

   Either $A[\text{midpoint}] < x$ or $A[\text{midpoint}] \geqslant x$.

   Case 1: Assume $A[\text{midpoint}] < x$.

   Then $A[\text{first}] \leqslant A[\text{first} + 1] \leqslant \ldots \leqslant A[\text{midpoint}] < x$.

   In this case, the algorithm sets first′ = midpoint + 1, so by the end of the iteration,

   - **$0 \leqslant \text{first}'$ # because $0 \leqslant \text{first} \leqslant \text{midpoint} < \text{midpoint} + 1$;**
   - **$\text{first}' \leqslant \text{last}$ # because midpoint $<$ last $\Rightarrow$ midpoint $+ 1 \leqslant$ last;**
   - **$\text{last} < \text{len}(A)$ # from the loop invariant, since last is unchanged;**
   - **$\exists i \in \{\text{first}', \ldots, \text{last}\}, A[i] = x$ # because $\exists i \in \{\text{first}, \ldots, \text{last}\}, A[i] = x$ (by the loop invariant), and $A[\text{first}] \neq x$, ..., $A[\text{midpoint}] \neq x$.**

   Hence, the loop invariant still holds.

   Case 2: Assume $A[\text{midpoint}] \geqslant x$.

   Then $A[\text{last}] \geqslant A[\text{last} - 1] \geqslant \ldots \geqslant A[\text{midpoint} + 1] \geqslant A[\text{midpoint}] \geqslant x$.

   In this case, the algorithm sets last′ = midpoint, so by the end of the iteration,

   - **$0 \leqslant \text{first}$ # from the loop invariant, since first is unchanged;**
   - **$\text{first} \leqslant \text{last}'$ # because first $\leqslant$ midpoint;**
   - **$\text{last}' < \text{len}(A)$ # because midpoint $<$ last $<$ len(A);**
   - **$\exists i \in \{\text{first}, \ldots, \text{last}'\}, A[i] = x$ # because $\exists i \in \{\text{first}, \ldots, \text{last}\}, A[i] = x$ (by the loop invariant), and either $A[\text{midpoint}] = x$ (so $A[\text{last}'] = x$) or $A[\text{midpoint}] > x$ (so $A[\text{midpoint}] \neq x$, ..., $A[\text{last}] \neq x$).**

   Hence, the loop invariant still holds.

   In both cases, the loop invariant is true at the end of the iteration.

   Hence, the loop invariant is true at the end of every iteration for which it was true at the beginning of the iteration.

   Hence, the loop invariant holds if the precondition was true.

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.

   We must show that the loop eventually terminates.

   The easiest way to do this rigorously is prove that (last − first) strictly decreases during each iteration of the loop, while remaining non-negative. This means the loop must eventually terminate because there is no infinite decreasing sequence of non-negative integers.

   (This is beyond the scope of this course — and so will be omitted here.)