1. Correctness of Binary Search con’t

Recall the algorithm for Binary Search:

```python
def BINSEARCH(A, x):
    #Precondition: A is a sorted array of length at least 1,
    #indexed from 0 to length(A)-1
    #Postcondition: Return an integer t such that 0 <= t <= length(A)-1,
    #and A[t]=x if such a t exists, and -1 otherwise.
    f = 0 #f is the first index of the subarray that you are looking for x in
    l = length(A) - 1 #l is the last index of the subarray
    while f != l:
        m = (f+l)/2 #middle element (this is integer division, i.e. rounding down)
        if A[m] >= x: #x is in the first half
            l = m
        else:
            f = m+1
    if A[f] = x:
        return f
    else:
        return -1
```

Before we get into the proof, here’s a useful fact:

**Lemma 1.**
\[ \forall f, l \in \mathbb{N}, f < l \rightarrow f \leq \left\lfloor \frac{f+l}{2} \right\rfloor < l \]

**Proof.**
\[
\left\lfloor \frac{f+l}{2} \right\rfloor \geq \left\lfloor \frac{f+f}{2} \right\rfloor = \left\lfloor \frac{2f}{2} \right\rfloor = [f] = f
\]
\[
\left\lfloor \frac{f+l}{2} \right\rfloor \leq \frac{f+l}{2} < \frac{l+l}{2} = \frac{2l}{2} = l
\]

A little bit of notation: In programming, variables can change their values. But in math, things are simpler: a variable never changes. To refer to variables in the code, we will index them. So \( f_0 \) is the initial value of \( f \) and \( f_i \) is the value of \( f \) after \( i \) iterations (i.e. after going through the loop \( i \) times).

Here, \( A[a \ldots b] \) is the subarray of \( A \) from index \( a \) to index \( b \), inclusive.

2. Partial Correctness

The bulk of the difficulty in proving that a program is correct is proving the loop invariant. Recall that a loop invariant is something that is true in every iteration.

**Lemma 2.** Suppose the precondition of BINSEARCH holds before the program starts. For each \( i \in \mathbb{N} \), if the loop is executed \( i \) times, then:

(i) \( 0 \leq f_i \leq l_i \leq length(A) - 1 \)
(ii) if \( x \) is in \( A \), then \( x \) is in \( A[f_i \ldots l_i] \)

The proof will be by Induction. Our predicate is:

\( P(i) \): if the loop is executed \( i \) times, then (i) and (ii) hold.,

and we are trying to prove:

\( \text{Precondition } \rightarrow \forall i \in \mathbb{N}, P(i) \) (equivalently \( \forall i \in \mathbb{N}, \text{Precondition } \rightarrow P(i) \)).
Proof. By Induction on $i$.

**Base case:** $i = 0$. 
$f_0 = 0$ and $l_0 = \text{length}(A) - 1$.

(i) is true since $\text{length}(A) \geq 1$ from the precondition.

(ii) is just a tautology.

**Induction step:** Let $i \in \mathbb{N}$. Assume that if the loop is executed $i$ times, then both (i) and (ii) hold. [IH] Assume the loop is executed $i + 1$ times.

Note that by the loop condition, $f \neq l$ (more specifically, $f < l$), so we can use Lemma 1.

**Case 1:** $A[m_{i+1}] \geq x$

$f_{i+1} = f_i$ and $l_{i+1} = m_{i+1} = \left\lfloor \frac{f_i + l_i}{2} \right\rfloor$.

Let’s prove (i) first:

$f_{i+1} = f_i \geq 0$ by IH.

$l_{i+1} = \left\lfloor \frac{f_i + l_i}{2} \right\rfloor = m_{i+1} = l_{i+1}$ by Lemma 1.

Thus, $0 \leq f_{i+1} \leq l_{i+1} \leq \text{length}(A) - 1$.

Now, to prove (ii), suppose $x \in A$. Then $x$ is in $A[f_i \ldots l_i]$ by IH. Since $A$ is sorted, $A[t] > x$ for all $t \in \mathbb{N}$ such that $m_{i+1} < t \leq l_i$. Since $x$ is in $A[f_i \ldots l_i]$ but not in $A[f_i \ldots m_{i+1}]$, $x$ must be in $A[f_i \ldots m_{i+1}]$. Recall that $f_{i+1} = f_i$ and $m_{i+1} = l_{i+1}$, and thus $x$ is in $A[f_i \ldots l_i]$. The second case is quite similar:

**Case 2:** $A[m_{i+1}] < x$

$f_{i+1} = m_{i+1} + 1 = \left\lfloor \frac{f_i + l_i}{2} \right\rfloor + 1$ and $l_{i+1} = l_i$.

Once again, let’s prove (i) first:

$f_{i+1} = \left\lfloor \frac{f_i + l_i}{2} \right\rfloor + 1 \geq f_i + 1 > f_i \geq 0$ by Lemma 1 and IH.

$l_{i+1} = \left\lfloor \frac{f_i + l_i}{2} \right\rfloor + 1 \leq l_i = l_{i+1}$ by Lemma 1.

Thus, $0 \leq f_{i+1} \leq l_{i+1} \leq \text{length}(A) - 1$.

Now, to prove (ii), suppose $x \in A$. Then $x$ is in $A[f_i \ldots l_i]$ by IH. Since $A$ is sorted, $A[t] < x$ for all $t \in \mathbb{N}$ such that $f_{i+1} \leq t \leq m_{i+1}$. Since $x$ is in $A[f_i \ldots l_i]$ but not in $A[f_i \ldots m_{i+1}]$, $x$ must be in $A[m_{i+1} + 1 \ldots l_i]$. Recall that $l_i = l_{i+1}$ and $m_{i+1} + 1 = f_{i+1}$, and thus $x$ is in $A[f_i \ldots l_i]$.

**Theorem (Partial Correctness).** If the precondition of BINSEARCH holds and it terminates, then the postcondition holds after execution.

Proof. Say the loop exits after $i$ iterations. By Lemma 2, we know that $P(i)$ is true, i.e., (i) and (ii) hold. By the while loop condition, $f_i = l_i$. Furthermore, from (i), $0 \leq f_i \leq \text{length}(A) - 1$.

**Case 1:** $x$ is in $A$. i.e., $\exists t \in \mathbb{N}, 1 \leq t \leq \text{length}(A) - 1$ and $A[t] = x$.

By part (ii), $x$ is in $A[f_i \ldots l_i]$. Since $f_i = l_i$, this is just a one element array, and thus $x = A[f_i]$, and the program correctly returns $f_i$.

**Case 2:** $x$ is not in $A$. i.e., $\forall t \in \mathbb{N}, 1 \leq t \leq \text{length}(A) \to A[t] \neq x$. So $A[f_i] \neq x$, and the program returns $-1$.

3. **Termination**

In order to complete the proof that the Binary Search algorithm is correct, we need to prove termination. We’ll use the following theorem:

**Theorem.** Every decreasing sequence of natural numbers is finite.

Why is this true? Well, if $a_0, a_1, a_2, \ldots$ is a decreasing sequence of natural numbers (i.e., $a_0 > a_1 > \ldots$), then $a_i \leq a_0$ for all $i \in \mathbb{N}$. Since the $a_i$ are unique and there are only a finite number of natural numbers from 0 to $a_i$, the sequence must be finite.

In this case, we’ll show that the size of the subarray $A[f \ldots l]$ is decreasing. We already know from Lemma 1 that $f \leq l$. Thus $f_0 - l_0, f_1 - l_1, f_2, l_2, \ldots$ is a sequence of natural numbers. If we show that the sequence is decreasing, the above theorem will give us that this sequence is finite, and thus the loop is finite.

**Theorem (Termination).** $\forall i \in \mathbb{N}$, if the loop is run for $i + 1$ steps, then $l_{i+1} = f_{i+1} < l_i - f_i$.

Proof. Since the loop is run for $i + 1$ iterations, $f_i \neq l_i$ by the while loop condition. Thus $f_i < l_i$, so we can apply Lemma 1: $f_i \leq \left\lfloor \frac{l_i + f_i}{2} \right\rfloor = m_{i+1} < l_i$.

**Case 1:** $A[m_{i+1}] \geq x$

$f_{i+1} = f_i$ and $l_{i+1} = m_{i+1}$.

$l_{i+1} - f_{i+1} = m_{i+1} - f_i < l_i - f_i$ by Lemma 1.
Case 2: $A[m_{i+1}] < x$
$f_{i+1} = m_{i+1} + 1$ and $l_{i+1} = l_i$.
$l_{i+1} - f_{i+1} = l_i - (m_{i+1} + 1) < l_i - m_{i+1} \leq l_i - f_i$ by Lemma 1.