2.1-3 Linear Search

```
procedure LINEAR-SEARCH(A, v)
    for i ← 1 to length[A] do
        if A[i] == v then
            return i
    return NILL
```

Proof of correctness To prove linear search is correct, I formulate a *loop invariant* that need to hold true at *initialization*, during *maintainance* and at *termination*.

Loop invariant At the start of loop iteration i, the value v is not in the sub-array $A[1..i-1]$. In formal terms, $v \notin \{A[j] | j \in \{1, ..., i-1\}\}$

Initialization At initialization $i = 0$, so the sub-array $A[1..i-1]$ is empty. So v is trivially not a member of that array, and the loop invariant holds.

Maintenance Assuming that v is not in the sub-array $A[1..i-1]$ at the start of iteration i, then two outcomes are possible during the iteration. Either, $A[i] == v$ and the loop terminates, or $A[i] \neq v$ and v is not an element of the array $A[1..i]$ which is the loop invariance condition for iteration $i + 1$.

Termination The algoritm terminates in two different ways. The first happens when $A[i] == v$, assuming $v \notin A[1..i-1]$ before termination, then the loop invariant still holds. Again, assuming $v \notin A[1..i-1]$ before termination. In the second case we have that $i = n+1$, in that case $v \notin A[1..i-1] = A[1..n]$, which is the entire array, and the procedure returns NILL.

2.2-3 Running time of Linear search

Assuming the index of correct element is uniformlly distributed from 1 to n, the expected index is $E[i] = n/2$. So the average case running time must be $c_1 \frac{n}{2} + c_2$, where c_1 is the running time of each comparison $A[i] == v$ and the loop increment, and c_2 is the running time of the return statement. In the worst case the algorithm needs to run through all n elements as well as returning NILL, so the running time is $c_1 n + c_2$. Both the average and worst case running times of linear search are $\Theta(n)$.

2.3-5 Insertion srort with Binary search

```
procedure BINARY-SEARCH(A, p, r, v)
    if p ≤ r then
        m = p + ⌊(r − p)/2⌋
        if A[m] == v then
            return m
```

```
        else if A[m] > v then
            return BINARY-SEARCH(A, p, m - 1, v)
        else
            return BINARY-SEARCH(A, m +1, r, v)
    return p
```