Searching

Linear Search

- **Exercise 2.1-4**. This problem askas us to write the pseudocode for linear search algorithm, which scans an input array from begining to the end, looking for a given element x. This is the simplest search algorithm and does not require that the input array is sorted.

```
LINEAR-SEARCH(A, x)
// Input: a sequence of numbers A[1:n] and a value x
// Output: An index i such that x equals A[i] or NIL is x is not found
1. i = 1
2. while i <= n and A[i] != x
3.    i = i + 1
4. if i == n + 1
5.    return NIL
6. else return i
```

- **Loop Invariant**: At the start of each iteration of the while loop, x is not present in A[1:i-1]
  - **initialization**: Before the start of the while loop, $i = 1$ and $A[1 : i-1] = A[1 : 0]$, an empty subarray, which does not contain x.
  - **maintenance**: Now we show that each iteration maintains the invariant. If $i = n+1$, then the loop condition fails and it terminates and the invariant for A[1:n] holds. This is the spceial case when the value x is not found in the array. The second condition for the while loop checks if $A[i]! = x$ and only then goes into another iterations. Thus we can infer that $A[1 : i]$ does not contain x and the invariant holds true for the next iteration.
  - **termination**: The loop terminates if $i = n+1$, then the invariant says that x was not found in $A[1 : n]$. If the loop terminates with $i <= n$, then $A[i]$ must be equal to x for the loop condition to fail.
- **Runtime analyis**: In the worst-case, the while loop walks through the entire array. Each iteration of the loop takes $\Theta(1)$ time, so the wrost-case run time is $\Theta(n)$.
- Sentinel:

Binary Search

If the input array is sorted, we can check the midpoint of the subaray against x and eliminate half the subarray from further consideration. Then we can apply this idea recursively until we have either found the location of x or determined that it it doesn't exist in the oinput array.

```
BINARY-SEARCH(A, x)
// Input: A[1:n] is a sorted array, x is the element we are searching for
// Output: An index i such that x equals A[i] or NIL is x is not found
1. left = 1
2. right = n
3.
```