1.1-3
Select a data structure that you have seen previously, and discuss its strengths and limitations.

2.1-1
Using Figure 2.2 as a model, illustrate the operation of INSERTION-SORT on the array \( A = \{31, 41, 59, 26, 41, 58\} \).

2.1-2
Rewrite the INSERTION-SORT procedure to sort into nonincreasing instead of non-decreasing order.

2.1-3
Consider the searching problem:

**Input:** A sequence of \( n \) numbers \( A = \{a_1, a_2, \ldots, a_n\} \) and a value \( v \).

**Output:** An index \( i \) such that \( v = A[i] \) or the special value NIL if \( v \) does not appear in \( A \).

Write pseudocode for *linear search*, which scans through the sequence, looking for \( v \). Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

2.2-3
Consider linear search again (see Exercise 2.1-3). How many elements of the input sequence need to be checked on the average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst case? What are the average-case and worst-case running times of linear search in \( \Theta \)-notation? Justify your answers.

**BUBBLESORT**

\[
\begin{align*}
1 & \textbf{for } i = 1 \textbf{ to } A.\text{length} - 1 \\
2 & \quad \textbf{for } j = A.\text{length} \textbf{ downto } i + 1 \\
3 & \quad \textbf{if } A[j] < A[j - 1] \\
4 & \quad \quad \text{exchange } A[j] \text{ with } A[j - 1]
\end{align*}
\]

2.2
Use analyzing technique similar to that used by the textbook for analysis of Insertion sort, to give the best and the worst case analysis of above implementation of Bubble sort.