4. **a.** Write pseudocode for a selection sort algorithm that selects the largest, instead of the smallest, entry in the array and sorts the array into descending order.
   b. Using your algorithm, repeat Exercise 1.
   c. Revise the iterative method `selectionSort`, as given in Segment 8.6, so that it implements your algorithm.

5. Repeat Exercise 4, but this time sort the array into ascending order.

6. Revise the iterative method `selectionSort`, as given in Segment 8.6, so that it has `first` and `last` as parameters instead of `n`.

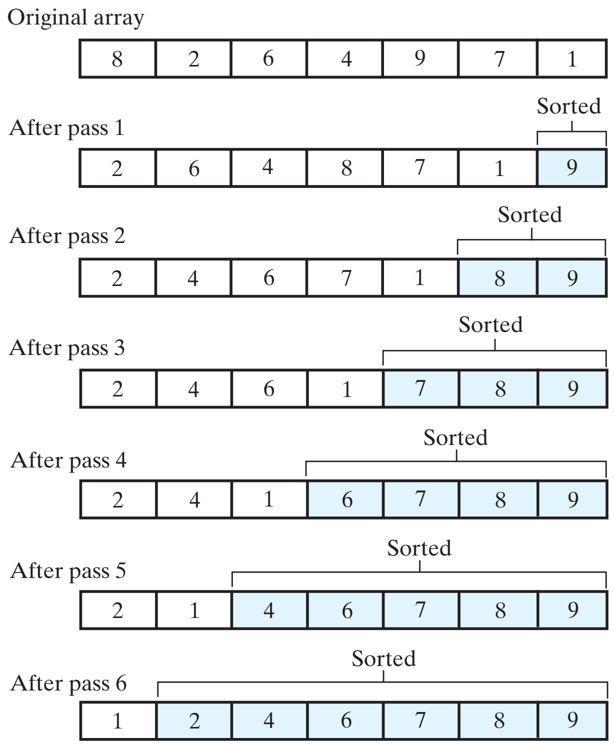
7. Consider a revised selection sort algorithm so that on each pass it finds both the largest and smallest values in the unsorted portion of the array. The sort then moves each of these values into its correct location by swapping array entries.

   a. How many comparisons are necessary to sort *n* values?
   b. Is the answer to Part *a* greater than, less than, or equal to the number of comparisons required by the original version of selection sort?

8. A **bubble sort** can sort an array of *n* entries into ascending order by making *n* – 1 passes through the array. On each pass, it compares adjacent entries and swaps them if they are out of order. For example, on the first pass, it compares the first and second entries, then the second and third entries, and so on. At the end of the first pass, the largest entry is in its proper position at the end of the array. We say that it has bubbled to its correct spot. Each subsequent pass ignores the entries at the end of the array, since they are sorted and are larger than any of the remaining entries. Thus, each pass makes one fewer comparison than the previous pass. Figure 8-17 gives an example of a bubble sort.
   Implement the bubble sort

   a. Iteratively
   b. Recursively

FIGURE 8-17 A bubble sort of an array (see Exercise 8)


Original array
8 2 6 4 9 7 1
After pass 1
Sorted
2 6 4 8 7 1 9
After pass 2
Sorted
2 4 6 7 1 8 9
After pass 3
Sorted
2 4 6 1 7 8 9
After pass 4
Sorted
2 4 1 6 7 8 9
After pass 5
Sorted
2 1 4 6 7 8 9
After pass 6
Sorted
1 2 4 6 7 8 9