**2.5.21** Multidimensional sort. Write a Vector data type for use in having the sorting methods sort multidimensional vectors of d integers, putting the vectors in order by first component, those with equal first component in order by second component, those with equal first and second components in order by third component, and so forth.

**2.5.22** Stock market trading. Investors place buy and sell orders for a particular stock on an electronic exchange, specifying a maximum buy or minimum sell price that they are willing to pay, and how many shares they wish to trade at that price. Develop a program that uses priority queues to match up buyers and sellers and test it through simulation. Maintain two priority queues, one for buyers and one for sellers, executing trades whenever a new order can be matched with an existing order or orders. **2.5.23** Sampling for selection. Investigate the idea of using sampling to improve selection. Hint: Using the median may not always be helpful. **2.5.24** Stable priority queue. Develop a stable priority-queue implementation (which returns duplicate keys in the same order in which they were inserted).

**2.5.25** Points in the plane. Write three static comparators for the Point2D data type of page 77, one that compares points by their x coordinate, one that compares them by their y coordinate, and one that compares them by their distance from the origin. Write two non-static comparators for the Point2D data type, one that compares them by their distance to a specified point and one that compares them by their polar angle with respect to a specified point.

**2.5.26** Simple polygon. Given N points in the plane, draw a simple polygon with N Sorting points as vertices. Hint : Find the point p with the smallest y coordinate, breaking ties with the smallest x coordinate. Connect the points in increasing order of the polar angle they make with p.

**2.5.27** Sorting parallel arrays. When sorting parallel arrays, it is useful to have a version of a sorting routine that returns a permutation, say index[], of the indices in sorted order. Add a method indirectSort() to Insertion that takes an array of Comparable objects a[] as argument, but instead of rearranging the entries of a[] returns an integer array index[] so that a[index[0]] through a[index[N-1]] are the items in ascending order.

**2.5.28** Sort files by name. Write a program FileSorter that takes the name of a directory as a command-line argument and prints out all of the files in the current directory, sorted by file name. Hint : Use the File data type.

**2.5.29** Sort files by size and date of last modification. Write comparators for the type File to order by increasing/decreasing order of file size, ascending/descending order of file name, and ascending/descending order of last modification date. Use these comparators in a program LS that takes a command-line argument and lists the files in the current directory according to a specified order, e.g., "-t" to sort by timestamp. Support multiple flags to break ties. Be sure to use a stable sort.