Fundamentals of Computer Science I (CS151.01 2006F)

Homework 15: Selection Sort

Assigned: Tuesday, 14 November 2006
Due: Friday, 17 November 2006
No extensions!

Summary: In this assignment, you will explore another sorting algorithm, selection sort.

Purpose: To help you think more about sorting.

Expected Time: One to two hours.

Collaboration: You may work in a group of any size between one and four, inclusive. You may consult others outside your group, provided you cite those others. You need only submit one assignment per group.

Submitting: Email me your work, using a subject of *CSC151 Homework 15*.

Warning: So that this exercise is a learning assignment for everyone, I may spend class time publicly critiquing your work.

Background: Selection Sort

As you may recall from our in-class discussion of sorting techniques, one popular strategy for sorting vectors involves repeatedly finding the smallest remaining value and then swapping it into the next place in the vector. We call this procedure *selection sort* because the primary operation is selecting the smallest value.

The process is fairly straightforward. For example, in a vector of 10 elements, selection sort

- Find the index of the smallest element in positions 0 .. 9 and swap it into position 0.
- Find the index of the smallest element in positions 1 .. 9 and swap it into position 1.
- Find the index of the smallest element in positions 2 .. 9 and swap it into position 2.
- ...
- Find the index of the smallest element in positions 8 .. 9 and swap it into position 8.

As that example suggests, a key helper for selection sort is finding the index of the smallest value in a subvector. We might document that procedure as follows:

```
;;; Procedure:
;;;   index-of-smallest
;;; Parameters:
;;;   vec, a vector
;;;   pos, an integer
;;;   may-precede?, a binary predicate
;;; Purpose:
```

```
;;;   Find the index of the smallest value in positions [pos..len) of vec.
;;; Produces:
;;;   ios, an integer
;;; Preconditions:
;;;   may-precede? is transitive and reflexive.
;;;   0 <= pos < (vector-length vec).
;;;   vec contains no duplicate values.
;;; Postconditions:
;;;   For all i, pos <= i < (vector-length vec), i not equal to ios,
;;;     (may-precede? (vector-ref vec ios) (vector-ref vec i)).
```

Assignment

a. Implement `index-of-smallest`.

b. Implement `(swap! vec i j)`, a procedure that swaps the values at positions `i` and `j` of `vec`.

c. Using `index-of-smallest` and `swap!`, implement `selection-sort!`. Note that you will probably want to iterate over the position in the vector.
