

Lab: Sorting

- Exercises
 - Exercise 0: Preparation
 - Exercise 1: Testing Insert
 - Exercise 2: Inserting strings
 - Exercise 3: Generalizing Insertion
 - Exercise 4: Displaying Steps in Insertion Sort
 - Exercise 5: Checking Potential Problems
 - Exercise 6: Estimating Running Times
 - Exercise 7: Generalizing Insertion Sort
 - Exercise 8: Inserting into Vectors
 - Exercise 9: Testing Vector-Based Insertion Sort

Exercises

Exercise 0: Preparation

Copy the code from the accompanying reading into DrScheme.

Exercise 1: Testing Insert

a. Test both versions of the `insert-number` procedure from the reading by inserting a number

- into an empty list;
- into a list of larger numbers, arranged in ascending order;
- into a list of smaller numbers, arranged in ascending order;
- into a mixed list, arranged in ascending order;
- into a list of copies of the number to be inserted.

b. What happens if `ls` is *not* in ascending order when `insert-number` is invoked?

Exercise 2: Inserting strings

Write a new `string-insert` procedure that inserts a string into a list of strings that are in alphabetical order:

```
> (string-insert "dog" (list "ape" "bear" "cat" "emu" "frog"))  
("ape" "bear" "cat" "dog" "emu" "frog")
```

Exercise 3: Generalizing Insertion

- Show how to call the generalized `insert` procedure using lists of strings.
- Show how to call the generalized `insert` procedure using lists of numbers.
- Redefine `insert-number` in terms of `insert`
- Document `insert`.

Exercise 4: Displaying Steps in Insertion Sort

- Add calls to the `display` and `newline` procedures to the body of the helper in `insertion-sort-numbers` so that it displays the values of `unsorted` and `sorted`, appropriately labeled, at each step of the sorting process.
- Use the revised `insertion-sort-numbers` procedure to sort the values 7, 6, 12, 4, 10, 8, 5, and 1.

Exercise 5: Checking Potential Problems

Test the `insertion-sort-numbers` procedure on some potentially troublesome arguments:

- an empty list,
- a list containing only one element,
- a list containing all equal values,
- a list in which the elements are originally in *descending* numerical order.

Exercise 6: Estimating Running Times

Using `time`, determine how long it takes to insertion sort lists for 50, 100, 150, and 200 numbers. You should probably try a few different lists of each size. If you find that these examples are too quick, use larger lists.

You may want to use the following procedure to generate your lists.

```
;;; Procedure:
;;; random-list
;;; Parameters:
;;; max, the largest value to be produced
;;; len, an integer
;;; Purpose:
;;; Produces a list of "random" values.
;;; Preconditions:
;;; max > 0
;;; len >= 0
;;; Postconditions:
;;; The result list has length length.
;;; Every value in the result list is between 0 and max, inclusive.
;;; The result list is hard to predict.
```

```
(define random-list
  (lambda (max len)
    (if (= len 0) null
        (cons (random (+ max 1)) (random-list max (- len 1))))))
```

Exercise 7: Generalizing Insertion Sort

Document, write, and test a procedure, (`insertion-sort list may-precede?`), that generalizes the `insertion-sort-numbers` procedure.

Exercise 8: Inserting into Vectors

Document, write, and test the generalized `insert!` procedure to insert into a sorted vector. Your procedure should take the following parameters:

- *vec*, a vector;
- *val*, a value to insert;
- *may-precede*, a binary predicate;
- *space-pos*, the position of an “empty space” in the vector.

Your procedure should shift values as appropriate to make space for the new value. Your procedure should not affect values in the vector after *space-pos*.

Hint: Work backwards from *space-pos*.

Exercise 9: Testing Vector-Based Insertion Sort

Create and name a vector containing the strings "bear", "emu", "frog", "ape", "dog", and "cat".

Rearrange the elements of the vector into alphabetical order by means of an appropriate call to `insertion-sort!`. (Note that the sorting occurs as a side effect of this call. Hence, to confirm that the sorting procedure worked you'll have to inspect the vector again afterwards.)