Homework 1

CS 173: Intermediate Computer Science — Spring 2014
Instructor: Thomas Bressoud — Due: 2014-01-31 (classtime)

In this exercise, you will compare the running times of three sorting algorithms: selection sort, insertion sort, and Python's built-in sort method.
The selection sort algorithm is described in programming exercise 3 on page 37 of your text book.
Insertion sort is the algorithm that most people use to sort a deck of cards. Start with your left hand empty and the entire deck in your right hand. Your left hand will always contain a sorted pile of the cards considered so far while your right hand will contain the cards yet to be sorted. At each step, take the top card from your right hand and insert it into the sorted cards in your left hand. Repeat this process `n` times. In your function, all of the items remain in your original list; the index of the top item in your right hand will be marked with an increasing `for` loop variable and all of the items to the left of this index will be sorted. Inserting an item in the correct location in the left side of the list will require another (`while`) loop.
Both of these sorting algorithms should be done *in-place* and not create and append/delete from lists but should operate by swapping values within the list.

1. In a file called `studentsort.py`, write functions for selection sort, named `selection_sort` and insertion sort, named `insertion_sort` that each take a list of numbers as a parameter, return nothing, and sort the list in place. Also create a boolean function, `sorted` that takes a list as a parameter and returns a boolean, indicating whether or not the passed list is indeed in non-decreasing order. There may be repeated values in the lists. Be sure and write good function-level docstrings (with good pre and post-conditions) and inline comments describing your algorithms.

2. In a separate Python file called `runsort.py` in the same directory, use `from studentsort import *` to gain access to your functions and in this new file create a main function and its invocation to drive testing of your solutions. Use the time() function (see page 19) to time each of the three sorts (your two plus the built-in Python sort function) multiple times on lists of 10,000, 100,000, and 1,000,000 random integers between 1 and 1,000,000. Plot your timing results. (I find Excel the easiest to use for this, particularly if I print comma separated values from my program and then copy them into a text file that I can import into Excel.)

3. What is the θ running time of selection sort and insertion sort? Based on your timing results, what do you think the θ running time of the built-in sort method is?

I am still working on the best way to submit your work to me. I am currently thinking of using either DropBox or Google Drive to allow submission in the cloud, but need to validate the method, so I will inform you next week of the submission process.