Homework 3

Due Monday, June 30, at 9 AM (GMT-7)

CSc 345 – Summer 2014

Instructor: Qiyam Tung

Instructions

1. This is an individual assignment. You must do your own work.
2. If you are having difficulty and need to ask a question you can:
   (a) Ask questions in class.
   (b) Stop by my office hours (or make an appointment).
   (c) Post a question on Piazza.
   (d) Post a private question on Piazza if the question is too specific.
3. Show all work. Incomplete solutions will **not** receive full credit
4. You may write your solutions by hand, or you may type them using any appropriate program such as Microsoft Word, OpenOffice Writer, LaTeX, etc...
   However, the final copy should be in PDF form and formatted so that it is legible.
5. If the listed problem is only a number, refer to the online book for the description of the problem.

Problems (70 points possible)

1. (10) Question 7.1 (P.257)
2. (10) Question 7.4 (P.258)
3. (5) Question 7.5(a) (P.258)
4. (10) (**Stability**) Of the sorting algorithms Insertion Sort, Bubble Sort, Selection Sort, Shellsort, Mergesort, Quicksort, which of these are stable, and which are not (based on the pseudo-code showed in class)? For Selection Sort, please describe either why it is or is not stable. If it is not stable, how to modify the implementation to make it stable? Describe the change, and explain how it effects the running time.
5. (25) (**Funsort**) Consider this sorting algorithm, where low and high are indices into the array list marking the sequence of keys to be sorted:

```
FunSort(list, low, high) returns list
    if (low < high) then
        FunSort(list, low, high-1)
        if list[high-1] > list[high] then
            swap list[high-1] and list[high]
            FunSort(list, low, high-1);
        end if
    end if
end FunSort
```

   (a) Give a sample list of 5 key values that represents a best case input to FunSort.
   (b) Identify the recurrence that describes the number of key comparisons performed by FunSort in the best case, **and** solve the recurrence (that is, find an equivalent closed-form expression).
   (c) Give a sample list of 5 key values that represents a worst case input to FunSort.

(d) The worst case recurrence (again for key comparisons) appears to be K(n) = K(n-1) + 1 + K(n-1). Solve this recurrence (show your work!), and tell us the Big-O of K(n). Note that no proof of your solution is required, but if you want to be sure that it is correct . . .

(e) Code FunSort, modifying it to also report the number of key comparisons performed (remember what we did in step-counting?). Using your implementation, complete the following table. (We dont want to see your code, so be as sloppy with your coding as you wish!)

(i)	n	worst-case comparisons(w)	w_i/w_{i-1}
1	10		
2	20		
3	40		
4	80		
5	160		
6	320		
7	640		
8	1280		

6. (10) (**Sorting the k largest elements**) Suppose you have a list of n numbers. You wish to find the k numbers on this list that are largest, and return these k numbers in sorted order. Devise an algorithm to solve this problem, and analysis the running time of your algorithm.

(Hint: basic idea appeared in Counting Sort)