CSCI211: Problem Set 2

Due Friday, January 27
Points Possible: 25

1. **4 pts. (2.4) Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.**

$$f_1(n) = 2^{\sqrt{logn}} \quad f_2(n) = 2^n \quad f_3(n) = n^{4/3} \quad f_4(n) = n(logn)^3$$
$$f_5(n) = n^{logn} \quad f_6(n) = 2^{2^n} \quad f_7(n) = 2^{n^2}$$

2. **8 pts. Do chapter 2, exercise 6.** You may want to implement your algorithm to verify its correctness. Some Python code is available on the course web site to get you started.

3. **9 pts. (Levitan, 6.4.1)**

   (a) **Construct a heap for the keys 1, 8, 6, 5, 3, 7, 4 by successive key insertions (top-down algorithm, i.e., Heapify-Up). You can draw your heap as an array or a tree. Tree will probably be easier. Show your work to ensure the possibility of partial credit.**

   (b) **Construct a heap for the same keys (stored in an array) using the following bottom-up algorithm:**

Algorithm 1 HeapBottomUp(H[1...n])

```
for i = ⌊n/2⌋ downto 1 do
    k = i
    v = H[k]
    heap = false
    while not heap and 2 * k ≤ n do
        j = 2 * k
        if j < n then
            if H[j] > H[j + 1] then
                j = j + 1
            end if
        end if
        if v ≤ H[j] then
            heap = true
        else
            H[k] = H[j]
            k = j
        end if
    end while
    H[k] = v
end for
```

(c) **Compare the algorithms in terms of runtime and conceptual complexity.**

(d) **Will the successive-key insertions and bottom-up algorithms always yield the same heap for the same insertions?**

4. **4 pts. (Levitan, 6.4.2) Design an algorithm for checking whether an array $H[1...n]$ is a heap and determine its runtime efficiency.**