CS41 Homework 3

This homework is due at 11:59PM on Sunday, September 23. Write your solution using LaTeX. Submit this homework using **github** as a file called **hw3.tex**. This is an individual homework. It's ok to discuss approaches at a high level. In fact, we encourage you to discuss general strategies. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab partner *while in lab*. In this case, note who you've worked with and what parts were solved during lab.

1. **Asymptotic analysis, specifics.** Arrange the following functions in ascending order of growth rate. That is, if g follows f in your list, then it should be the case that $f = O(g)$.

   - $f_1(n) = n^{2.8}$
   - $f_2(n) = 6n + 10$
   - $f_3(n) = n^2 \log(n)$
   - $f_4(n) = 10^n$
   - $f_5(n) = 100^n$
   - $f_6(n) = 2^{2^n}$
   - $f_7(n) = \sqrt{4n}$

   No proofs are necessary.

2. **Asymptotic analysis, generalized.** For these problems, your example functions should have domain and range $\mathbb{N}$.

   (a) Show that if $f(n)$ is $\Omega(g(n))$ and $g(n)$ is $\Omega(h(n))$, then $f(n)$ is $\Omega(h(n))$.

   (b) Give a proof or counterexample: If $f(n)$ is not $O(g(n))$, then $g(n)$ is $O(f(n))$.

   (c) Let k be a fixed constant and suppose that $f_1, \ldots, f_k$ and h are functions such that $f_i = O(h)$ for all i.

      i. Let $g_1(n) := f_1(n) + ... + f_k(n)$. Is $g_1 = O(h)$? Prove or give a counterexample. If you give a counterexample, give an alternate statement that best captures the relationship between the two functions.

      ii. Let $g_2(n) := f_1(n) \cdot \ldots \cdot f_k(n)$. Is $g_2 = O(h)$? Prove or give a counterexample. If you give a counterexample, give an alternate statement that best captures the relationship between the two functions.

3. **Close to sorted.** Say that a list of numbers is "k-close-to-sorted" if each number in the list is less than k positions from its actual place in the sorted order. (So a 1-close-to-sorted list is *actually* sorted.) Give an $O(n \log k)$ algorithm for sorting a list of numbers that is k-close-to-sorted.

   In your algorithm, you may use any data structure or algorithm from CS35 by name, without describing how it works.

4. **(extra challenge problem)** For these problems, your *distinct* example functions should have domain and range the positive integers $\mathbb{N}$.

   - Find (with proof) a function f_1 such that $f_1(2n)$ is $O(f_1(n))$.
   - Find (with proof) a function f_2 such that $f_2(2n)$ is not $O(f_2(n))$.
   - Find (with proof) a function f_3 such that $f_3(2^n)$ is $O(f_3(n))$.

5. **(extra extra challenge problem)** Define a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ such that $f = O(g)$ for all exponential functions g, but f is *not* $O(h)$ for any polynomial function h.