Homework 1

Description

The goal of this assignment is to master the following skills:

- be able to write basic algorithms using recursion
- understand and be able to use asymptotic notation (O, Θ, Ω)
  - given two functions, recognize all their possible relationships in terms of O, Θ and Ω
  - simplify functional expression with O-notation

Submission

Problems to be submitted

Problem 1

Complete the following implementation in Java.

```
/* Given a string and a character, find the number of times the character
 * appears in the string. Matches are case-sensitive.
 * Give a recursive implementation.
 */
public static int countChar(String str, char c) {
        return 0;
}
```

Problem 2

Complete the following implementation in Java.

```
/* Find the maximum value in a list of Integers, using recursion.
 *
 * Hint: to keep track of which parts of the list still need to be visited,
 * which works a lot like cleanHotel(int lo, int hi)
```

```
     * use a recursive helper function: recursiveMaxHelper(List<Integer> li, int lo, int
     *
     */
    public static int recursiveMax(List<Integer> li) {
        return 0;
    }
```

Problem 3

Complete the following implementation in Java.

```
    /* Return a list of the prime factors of a given integer n, using recursion.
     * Your function should call findAPrimeFactor() to find a single prime factor for yo
     * (We're using it as a building block).
     *
     * You may assume that n is positive.
     */
    public static List<Integer> recursivePrimeFactors(int n) {
            return null;
    }

    /* Return a prime factor of n. */
    public static int findAPrimeFactor(int n) {
            for (int f = 2; f < n; f++)
                    if (n % f == 0)
                            return f;
            return n;
    }
```

Problem 4

Show that $3^{\log_4 n} = n^{\log_4 3}$ using log rules. Give an identity that generalizes this equality. (This problem is an exercise is using log/exponent rules. The identity will also be useful for simplifying expressions.)

Problem 5

a) If $f(n)$ is $O(g(n))$, is it necessarily true that $2^{f(n)}$ is $O(2^{g(n)})$? Justify your answer.

b) Explain why the statement “The running time of algorithm A is at least $O(n^2)$” is meaningless.

Problem 6

Order by Θ-notation, and indicate those have the same asymptotic growth rate (i.e., that are Θ of each other): e^n, n, 2^n, $n\log n$, $\log n$, n^2, n^4, $n\log_{10} n$, $\sqrt{n}$, $\sqrt[36]{n}$, 2^{n^2}, $\log\sqrt{n}$, $(\log n)^2$, $\log n^2$.

Note: intuitively, “f is $\Theta(g)$” roughly means “$f = g$”, the same way that “f is $O(g)$” roughly means “$f \leq g$”.

Formally, “f is $\Theta(g)$” means “f is $O(g)$” and “g is $O(f)$”.

Problem 7

Give a big-O estimate for each of the following functions. For the function g in your estimate that $f(n)$ is $O(g(n))$ use as simple of a function g as possible, and of the smallest order.

a) $5n^3 - 7n^2 + 88$
b) $(n\log n + n^2)(n^3 + 2)$
c) $\log(n^3 + 1) + (\log n)^2$
d) $(n\log n + 1)^2 + (\log n + 1)(n^2 + 1)$
e) $n^{2n} + n^{n^2}$
f) $(n! + 2^n)(n^3 + \log(n^2 + 1))$
g) $n \cdot (5/4)^{\log_4 n + 1}$

Hint: you can simplify **g)** to $O(n^?)$ where ? is something for you to determine.

Problem 8

Give a big-O estimate for each of the following functions. For the function g in your estimate that $f(n)$ is $O(g(n))$ use as simple of a function g as possible, and of the smallest order.

a) $\sum_{i=10}^{n} i$
b) $\sum_{i=1}^{\log_2 n} ni$
c) $\sum_{i=1}^{n} 4^n$

d) $\sum_{i=1}^{n} 4^i$

e) $\sum_{i=1}^{\log_2 n} 4^i$

f) $\sum_{i=1}^{\log_2 n} n(1/3)^i$

Additional problems (not to be submitted)

need more practice with recursion?

- finish the practice problems in
- at codingbat.com, solve **Recursion-1** problems

challenging asymptotic growth rates

The following functions commonly appear in the analysis of more advanced algorithms:

- $\log \log n$,
- $n/(\log \log n)$,
- $\log^* n$ (you can also find an explanation in the textbook)

Can you order these with respect to the functions in Problem 6?