CSC236 Fall 2016
Assignment #2
due November 4th, 10 p.m.

These problems are to give you some practice proving facts about recurrences, and the time complexity of algorithms that are expressed as recurrences.

Your assignment must be typed to produce a PDF document **a2.pdf** (hand-written submissions are not acceptable). You may work on the assignment in groups of **1 or 2** and submit a single assignment for the entire group on MarkUs. Your teammates must be **different** than all teammamtes you had in Assignment 1.

1. Consider the Fibonacci function f:

$$f(n) = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ f(n-2) + f(n-1) & \text{if } n > 1 \end{cases}$$

   Prove $f(n-1) * f(m-1) + f(n+1) * f(m+1) + f(n)f(m) > f(n+m)$ for $n, m \geq 1$.

2. (a) Find a recurrence relation, $T(n)$, for number of distinct full binary trees with n nodes. Show how you find the relation.

   (b) Without using a closed form, prove $T(n) \geq 2^{(n-1)/2}$ for most odd numbers.

3. (a) Find a recurrence relation, $T(n)$, for number of microorganisms in a microbial culture in which every 2 hours the number of microorganisms is quadrupled and also three times as many of the microorganisms die 4 hours after creation. There are initially 4 microorganisms in the culture.

   (b) Without using a closed form, prove $T(n)$ is strictly increasing.

   (c) Compute the closed form of $T(n)$ using unwinding (repeated substitution).

   (d) Prove the closed form, computed in part (c), is correct.

4. (a) Find a recurrence relation, $T(n)$, for number of distinct ways that a postage of n cents can be made by 4-cent, 6-cent, and 10-cent stamps for most even numbers.

   (b) Without using a closed form, prove $T(n)$ is non-decreasing.

5. (a) Devise a brute-force algorithm in Python[1] notation, **max-sum**, to find the largest sum of consecutive terms of a sequence of n positive and negative numbers.

   (b) Find the worst-case time complexity of **max-sum**.

   (c) Devise a divide-and-conquer algorithm to find **max-sum**, by splitting the sequence to halves, and finding **max-sum** in each. Note that the maximum sum of consecutive terms can include terms in both halves.

[1] or any other common programming language

(d) Find a recurrence relation for the worst-case time complexity of the divide-and-conquer **max-sum**.

(e) How does the time complexity of the the divide-and-conquer algorithm compare with that of the brute-force one?