Mcs-Cs 401, Summer 2016, Lowman, Quiz 1 (HW5).
due in class Monday, July 18.

1. Solve the following recurrence relation using a recurrence tree. $T(n) = T(n/3) + T(2n/3) + cn^2, T(1) = constant$

2. Use integrals to bound H_n, the nth harmonic number, on both sides and use your bounds to show that $H_n = \ln(n) + O(1)$,

3. Suppose that a function T is defined on the non-negative integers by

$$\begin{aligned} T(0) &= 0, \\ T(n) &= T(\lfloor n/3 \rfloor) + T(\lfloor n/5 \rfloor) + T(\lfloor n/7 \rfloor) + n, \quad n > 0 \end{aligned}$$

   Prove that $T(n) = O(n)$. Use strong induction.

4. Given the recursive function $f_n = f_{n-1} + f_{n-2}, f_0 = 0, f_1 = 1$
   (a) solve the recurrence by assuming $f(n) = a^n$
   (b) find a Big-Oh and Big-Omega for f_n. Your bounds should be tight.

5. Show Build-Max-Heap can always be done in $O(n)$ by summing the number of array element comparisons needed to fix the sub-heaps for every node that has children.

6. Look up what it means for a sorting algorithm to be a stable sort.
   (a) Is ***HeapSort*** a stable sorting algorithm? Work through a few convincing examples to show that it is or is not.
   (b) Is ***Mergesort*** a stable sorting algorithm? Work through a few convincing examples to show that it is or is not.
   (c) Is ***qsort*** a stable sorting algorithm? Use the same algorithm used in class (i.e. one sided partition function). Work through a few convincing examples to show that it is or is not.
   (d) Is ***isort*** a stable sorting algorithm? Work through a few convincing examples to show that it is or is not.

7. (a) Using the same qsort algorithm used in class (one sided partition function), what is the running time for an array with all array elements having the same key value? Do a walk-thru on an array of all equal elements to demonstrate how the bad case occurs.
   (b) Can the running time be improved by randomizing the input? Explain.
   (c) You should find that an array with all elements equal is a bad case for quicksort. How can this be fixed by modifying the partition function?
   (d) Give an algorithm for a modified partition function and walk though an array with all distinct elements to demonstrate how it works. Your new partition function should remove the bad case of all equal elements from qsort. Do a walk-thru on an array of all equal elements and demonstrate how it fixes the bad case.

8. The analysis of Randomized-Quicksort resulted in solving the following recurrence relation.

$$\boldsymbol{E[T(n)] = \frac{2}{n}\sum_{k=0}^{n-1} E[T(k)] + cn}$$

with appropriate base cases.

   (a) Use the substitution method to find a tight Big-Oh bound.

   (b) Solve the recurrence using either the repeated-substitution method or the telescoping-series method to get a tight Big-Oh bound.

9. In the selection problem (finding the k th smallest element), if we group the n elements into n/3 groups each of 3 elements and make appropriate changes to the algorithm, derive the speed of the resulting algorithm. Repeat it when each group consists of 7 elements.

10. When working with a nearly complete binary tree it was shown that

   (a) $\boldsymbol{\lceil \lg(n+1) \rceil = \lfloor \lg(n) \rfloor + 1}$

   (b) $\boldsymbol{Height = \lceil \lg(n+1) \rceil - 1}$

   (c) $\boldsymbol{Height = \lfloor \lg(n) \rfloor}$

Derive the three corresponding expressions for a nearly complete ternary tree.