CPSC 320: Intermediate Algorithm Design and Analysis
Assignment #3, due Friday, May 27th, 2016 at 2:15pm in Room x235, Box 32

[8] 1. Write recurrence relations for the running time of the following algorithms as a function of n.

[3] a. Assume that n is the number of elements of $A[\texttt{first}, \ldots, \texttt{last}]$, that is, $n = \texttt{last} - \texttt{first} + 1$.

```
Algorithm Quantal(A, first, last)
  if (n $\leq$ 5) then
    return A[first] + A[last]

  quint $\leftarrow \lfloor (4 \cdot \text{first} + \text{last})/5 \rfloor$
  return Quantal(A, first + 1, last − 2) $*$ Quantal(A, quint, last)
```

[5] b. Assume that the call `Ringtail(n, x)` runs in $O(\log n)$ time.

```
Algorithm Raring(A, first, n)
  h $\leftarrow$ 1
  sum $\leftarrow$ 1
  if (n < 8) then
    return n + 1

  while (h < n/2) do
    sum $\leftarrow$ sum + Ringtail(n, Raring(A, first + 2*h, n/4))
    h $\leftarrow$ h * 2
  return sum
```

[9] 2. Using the Akra-Bazzi method, give a tight upper bound on the solution to the following recurrence relation:

$$T(n) = \begin{cases} T(n/2) + 2 \cdot T(n/3) + T(n/6) + n^2 & \text{if } n \geq 6 \\ 1 & \text{if } n \leq 5 \end{cases}$$

Hint: Determining p exactly may be difficult, but you should be able to evaluate the integral in terms of the unknown parameter p, then plug this into the Akra-Bazzi formula, which also involves p. Next, one can find two integers p_1 and p_2 such that $p_1 < p < p_2$, and see what happens in the Akra-Bazzi formula.

[12] 3. Solve the following recurrence relations by applying the Master theorem. Assume that $T(n) \in \Theta(1)$ when n is sufficiently small.

[4] a. $T(n) = 8T(n/3) + n^2$

[4] b. $T(n) = 3T(\lfloor n/9 \rfloor) + \sqrt{n}$

[4] c. $T(n) = 5T(\lfloor n/2 \rfloor) + n^2$

[12] 4. You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values — so there are $2n$ values in total — and you may assume that no two values are the same. You would like to determine the median of this set of $2n$ values (this is the n^{th} smallest value).

However the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

[4] a. Suppose that you compare the median ($n/2^{th}$ smallest) element of the first database to the median element of the second database. These elements divide each database in two "halves". What does the result of the comparison tell you about the location of the i^{th} smallest value overall, with respect to the four "halves" of the two databases?

[8] b. Describe a divide-and-conquer algorithm that finds the median value using at most $O(\log n)$ queries.

[1] 5. (Bonus) How long did it take you to complete this assignment (not including any time you spent revising your notes before starting)?