CPSC 320: Intermediate Algorithm Design and Analysis
Assignment #3, due Friday, May 29th, 2015 at 2:15pm

[7] 1. Using the guess and test method, prove that the function $T(n)$ defined by

$$T(n) = \begin{cases} 2T(n-1) + 15T(n-2) + 48 & \text{if } n \geq 3 \\ 46 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \end{cases}$$

belongs to $O(5^n)$. Hint: you will need a trick we used in class.

[9] 2. Using a method of your choice, give a tight upper bound on the solution to the following recurrence relation:

$$T(n) = \begin{cases} 3T(\lfloor 4n/9 \rfloor) + 3T(\lfloor n/9 \rfloor) + n^{1.5} & \text{if } n \geq 9 \\ 1 & \text{if } n \leq 8 \end{cases}$$

[8] 3. 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 Narwhal(A, first, last)
  if (n < 8) then
    return A[first] + A[last]

  third ← (2 * first + last) / 3
  return Narwhal(A, first + 3, last - 2) * Narwhal(A, first, third)
```

[5] b. Assume that the call `Ocelot(n, x)` runs in $O(\sqrt{n})$ time.

```
Algorithm Pangolin(A, first, n)
  h ← 1
  sum ← 1
  if (n < 5) then
    return n + 1

  while (h < n/3) do
    sum ← sum + Ocelot(n, Pangolin(A, first + 2*h, n/3))
    h ← h * 3
  return sum
```

[8] 4. Determine whether or not each of the following recurrence relations can be solved by applying the Master theorem. Justify why or why not, and in the cases where the recurrence can be solved, give a Θ bound on the solution to the recurrence. Assume that in both cases, $T(n) \in \Theta(1)$ when n is sufficiently small.

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

[4] b. $T(n) = 10T(n/3) + f(n)$ where $f(n) = n$ when n is even, and $f(n) = n^2$ when n is odd.

[12] 5. 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] 6. (Bonus) How long did it take you to complete this assignment (not including any time you spent revising your notes before starting)?