University of Nevada, Las Vegas Computer Science 477/677 Spring 2015

Assignment 8: Due Tuesday, May 5, 2015

Name:_____

You are permitted to work in groups, get help from others, read books, and use the internet. But the handwriting on this document must be your own. You may attach extra sheets, using a stapler.

"Designing" an algorithm does not mean writing code. In fact, explicit hand-written pseudo-code can be hard to grade (and you don't want to upset the grader, do you?) You should describe your algorithm in English, although you can use **small** bits of hand-written pseudo-code to clarify, as needed.

Of course, you may actually wish to encode and then execute your algorithm. This does not relieve you of the obligation to completely explain your algorithm in your own handwriting.

Let $f(n)$ be defined for all non-negative integers n, as follows:

$$f(n) = \begin{cases} 0 \text{ if } n = 0 \\ 1 + f\left(\lfloor \frac{3}{5} n \rfloor\right) + f\left(\lfloor \frac{4}{5} n \rfloor\right) \text{ otherwise} \end{cases}$$

1. Give a recursive algorithm to compute $f(n)$. Show that it has time complexity $O(n^2)$

2. Give a dynamic programming algorithm to compute $f(n)$ Show that it has time complexity $O(n)$

3. Give a memoization algorithm to compute $f(n)$. Show that it has time complexity $O(log^2 n)$.

   This paragraph added `Sun Apr 26 15:00:00 PDT 2015`.

   I got this problem from some old stuff, but, to my surprise, I could not prove that the time complexity is $O(log^2 n)$. In fact, I no longer believe it, even though it seems to be really close, and I've tested out to $n = 100,000,000$. I do not want to simply guess what it is, so I'll just leave it open. Please present the memoization algorithm, and if you can prove some sort of asymptotic bound that is better than $O(n)$, please do so.