CS 325 Winter 17
HW 1 – 25 points

1) (2 pts) Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in $6n^2$ steps, while merge sort runs in 42nlgn steps. For which values of n does insertion sort beat merge sort? Explain. (**Note:** lg n is log "base 2" of n or $\log_2$ n.)

2) (4 pts) Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence

$$T(n) = \begin{cases} 2, & if\ n = 2 \\ 2T\left(\frac{n}{2}\right) + n, & if\ n = 2^k, \text{for } k > 1 \end{cases}$$

is $T(n) = n \lg n$.

3) (5 pts) For each of the following pairs of functions, either f(n) is O(g(n)), f(n) is Ω(g(n)), or f(n) = Θ(g(n)). Determine which relationship is correct and explain.
   a. $f(n) = n^{0.25}$; $g(n) = n^{0.5}$
   b. $f(n) = n$; $g(n) = \log^2 n$
   c. $f(n) = \log n$; $g(n) = \ln n$
   d. $f(n) = 1000n^2$; $g(n) = 0.0002n^2 - 1000n$
   e. $f(n) = n\log n$; $g(n) = n\sqrt{n}$
   f. $f(n) = e^n$; $g(n) = 3^n$
   g. $f(n) = 2^n$; $g(n) = 2^{n+1}$
   h. $f(n) = 2^n$; $g(n) = 2^{2^n}$
   i. $f(n) = 2^n$; $g(n) = n!$
   j. $f(n) = \lg n$; $g(n) = \sqrt{n}$

4) (4 pts) Describe and give pseudocode for an efficient algorithm that determines the maximum and minimum values in a list of n numbers. Show that your algorithm performs at most 1.5n comparisons. Demonstrate the execution of the algorithm with the input L = [9, 3, 5, 10, 1, 7, 12].

5) (4 pts) Let f_1 and f_2 be asymptotically positive non-decreasing functions. Prove or disprove each of the following conjectures. To disprove give a counter example.

   a. If $f_1(n) = O(g(n))$ and $f_2(n) = O(g(n))$ then $f_1(n) = \Theta(f_2(n))$.

   b. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$

6) (6 pts) **Fibonacci Numbers**:

The Fibonacci sequence is given by : 0, 1, 1, 2, 3, 5, 8, 13, 21, ….. By definition the Fibonacci sequence starts at 0 and 1 and each subsequent number is the sum of the previous two. In mathematical terms, the sequence F_n of Fibonacci number is defined by the recurrence relation

$$F_n = F_{n-1} + F_{n-2} \quad \text{with} \quad F_0=0 \text{ and } F_1=1$$

An algorithm for calculating the n^{th} Fibonacci number can be implemented either recursively or iteratively.

```
Recursive_fib (n) {
    if (n = 0) {
        return 0;
    } else if (n = 1) {
        return 1;
    } else {
        return fib(n-1) + fib(n-2);
    }
}

Iterative_fib (n) {
    fib = 0;
    a = 1;
    t = 0;
    for(k = 1 to n) {
       t = fib + a;
       a = fib;
       fib = t;
    }
    return fib;
}
```

a) Implement both the recursive and iterative algorithms to calculate Fibonacci Numbers in the programming language of your choice. Provide a copy of your code with your HW pdf. We will not be executing the code for this assignment. You are not required to use the flip server for this assignment.

b) Use the system clock to record the running times of each algorithm for n = 5, 10, 15, 20, 30, 50, 100, 1000, 2000, 5000, 10,000, …. You may need to modify the values of n if an algorithm runs too fast or too slow to collect the running time data. If you program in C your algorithm will run faster than if you use python. The goal of this exercise is to collect run time data. You will have to adjust the values of n so that you get times greater than 0.

c) Plot the running time data you collected on graphs with n on the x-axis and time on the y-axis. You may use Excel, Matlab, R or any other software.

d) What type of curve best fits each data set? Again you can use Excel, Matlab, any software or a graphing calculator to calculate the regression function. Give the equation of the curve that best “fits” the data and draw that curve on the graphs of created in part c).