**EE 367 Spring 2018**

**Homework 3 Total Points = 24**

**Problem X-1 (1 pt)** Prove that is

**Textbook CLRS Problem 2-2** (Correctness of Bubblesort, page 40, 2 pts; there are 4 parts, each part is worth 0.5 pts)

**Textbook CLRS Problem 3.1-4** (page 52, 1 pt)

**Textbook CLRS Problem 3-1-5** (page 53, 1 pt)

**Textbook CLRS Problem 3-1-7** (page 53, 1 pt)

**Textbook CLRS Problem 3.2-1** (page 60, 1 pt)

**Textbook CLRS Problem 3.2-2** (page 60, 1 pt)

**Textbook CLRS Problem 4.1-5** (page 75, 1 pt)

**Textbook CLRS Problem 4.2-7** (page 83, 1 pt) (comment: this type of problem is related to Strassen’s algorithm)

**Textbook CLRS Problem C.1-5** (page 1188, 1 pt).

**Textbook CLRS Problem C.1-7** (page 1188, 1 pt)

**Textbook CLRS Problem C.1-15** (page 1189, 1 pt) [Hint: See (C.34) ]

**Textbook CLRS Problem C.2-1** (page 1194, 1 pt)

**Textbook CLRS Problem C.3-2** (page 1200, 1pt)

**Textbook CLRS Problem C.4-2** (page 1206, 1 pt)

The next two problems are about are about determine asymptotic bounds of functions. For example, is f(x) O(g(x))?

If f( ) and g( ) are increasing (and pretty well behaved), we can take their ratio f(x)/g(x) and determine if it increases to infinity, decreases to 0, or converges to some positive constant as x goes to infinity. If it increases to infinity then f(g) is Ω(g(x)), if it decreases to 0 then f(g) is O(g(x)), and if it converges to a positive constant then f(g) is Θ(g(x)).

If the ratio is difficult to evaluate, we can take it’s log, i.e., log(f(x)) – log(g(x)).

* If this goes to infinity then f(g)/g(x) goes to infinity
* If this goes to negative infinity then f(x)/g(x) goes to zero
* If it stays constant then f(x)/g(x) goes to a positive constant.

If log(f(x)) and log(g(x)) are difficult to evaluate then you can take an additional log, i.e., log(log(f(x))) and log(log(g(x))).

* If log(log(f(x))) goes to infinity and faster than log(log(g(x))) then f(x) is Ω(g(x)).
* If log(log(f(x))) goes to infinity but slower than log(log(g(x))) then f(x) is O(g(x)).

**Problem 3.2-4** (page 60, 1 pt) Polynomially bounded means it is O( nb) for some constant b

**Problem X-2** (3 pts, 1pt per part) Below, “ln” is the natural log, and , , and are constants.

**(a)** Is  ? Prove your answer.

**(b)** Is  *n* , or , or both, or neither? Prove your answer.

**(c)** Is  , or , or both, or neither? Prove your answer.

**Problem X-3. [2 pts]** Attached with this homework is a file “rotate.c”. It has a function ‘rotate(head, k)’ that will rotate the linked list ‘head’ to the right by the shift amount ‘k’, but it doesn’t work. Implement this function correctly and it must have O(n) time complexity. **Upload this rotate.c into laulima.**

**Problem X-4** (2 pts). Attached is a program “merge.c”. It has a function merge( ) that is a recursive implementation of merge sort. Rewrite it so that it is implemented using an iterative algorithm rather than a recursive implementation and still has O(n log n) time complexity. **Upload this merge.c into laulima.**