**CSIS385 Algorithms**

**Lab #7 Mini Version: Divide & Conquer Algorithms**

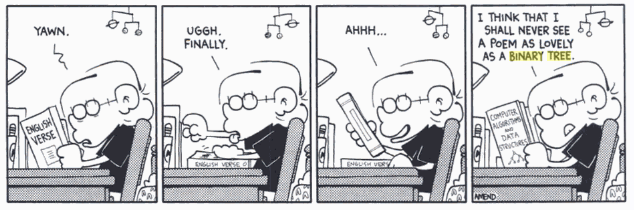
**Due Monday, March 20, 2017**

Names: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You may work in pairs on this lab if you wish, or you may work alone. If you work in pairs, turn in only one copy of the lab with both names on it.

Learning goals:

* To introduce the algorithm design technique of pre-sorting
* To review and extend understanding of binary search tree algorithms



1. (10 Points) Suppose you want to determine if all the elements in an array are unique. For example, the elements in array A = {6, 9, 2, 1, 0, 3, 10, 9, 5, 4} are not all unique because the number 9 appears more than once. You could easily solve this problem with brute force in (n2) time. Come up with an asymptotically more efficient algorithm that solves this problem. Write pseudocode for your solution below.

// A[0..n-1] is an array of n integer values. Return true if the values are all unique, and false if there

// are duplicates.

ALGORITHM UniqueElements ( A[0…n-1] )

Give the worst case running time of your algorithm and briefly explain how you got it.

Download the binary search tree implementation from Blackboard and complete the implementations of the following tree methods in the class. To debug your code, use the constructor BinarySearchTree( “small” ) to create a 4 node binary search tree and then invoke your methods on it. A picture of the “small” tree created can be found in the constructor code. To thoroughly test your code, use the constructor BinarySearchTree( “large” ) to create a 1000 node tree. Answers for the 1000 node tree are provided with each problem.

1. (10 Points) Complete the method recurisveNumDistK( int k, Node v ) that returns a count of the number of nodes in v’s subtree that are distance k (edges) away from v. For the 1000 node tree and k = 10, the answer is 103. Give the worst case running time of your algorithm and briefly explain how you got it.
2. (10 Points) Complete the method recursiveNumWithInRange( int s, int e, Node v ) that returns a count of the number of data values in v’s subtree that are in the range [s…e] (inclusive). Make your algorithm as efficient as possible by using of the binary search tree ordering property to your advantage. For the 1000 node tree and s=1000 and e=6000, the answer is 61.

Printout your code, screen shots showing the results of the runs on the large tree, and attach both to this lab.