CS 430 – SPRING 2023
INTRODUCTION TO ALGORITHMS
HOMEWORK #3
DUE Mar 2, 2023
(20pts)

1. (5 pts)What is the largest possible number of internal (key) nodes in a red-black tree with black-height *k* (measured from root)? What is the smallest possible number?

2. (5 pts) A team of biologists keeps information about DNA structures in a balanced binary search tree (i.e. AVL, red/black, etc) using as key the specific weight (an integer) of the structure. The biologists routinely ask questions of the type: "Are there any structures in the tree with specific weight between a and b (inclusive)?" and they hope to get an answer as soon as possible. Design an efficient algorithm that, given integers a and b, returns true if there exists a key x in the tree such that a<=x<=b, and false if no such key exists in the tree. Describe your algorithm in pseudocode. What is the time complexity of your algorithm?

3. (5 pts) Suppose that we can augment a red-black tree of n nodes with an additional attribute at each node, the size, which is the number of (internal) nodes in the subtree rooted at x. That is, size(x) and it is the number of internal nodes in the subtree rooted at x. Also, the size at a node x can be computed using only the information stored in x, left(x), and right(x), and that we can maintain "size" at all nodes during insertion and deletion in O(log n) time. In particular, we can maintain size so that Insert(x) and Delete(x) are supported in O(log n) time.
Write a new function on a red-black tree, Count(D, x), which returns the number of elements larger than x in the red-black tree D. Your function should work even if x is not in the red-black tree. Show your function runs in O(log n) time. (pseudo code required)

4. (5pts) x is an integer. All nodes in a BST *T* are primes. Design an algorithm to check x's primality. If x is a prime and its key exists in the BTS already, return 1; if x is a prime but its key does not exist in the BTS , insert it into the BST; if x is not a prime, return 0. (pseudo code required)