Summer CS161 Take Home Midterm

Due: Thursday, July 26th at 9 a.m. sharp.
No late submissions will be accepted.

Problem 1: Recurrences (20 points, 5 points each)

Give asymptotically tight upper (big O) bounds for $T(n)$ in each of the following recurrences. Prove your solution by naming the particular case of the master theorem, by iterating the recurrence, or by using the substitution method. You may use the version of the master theorem stated in Exercise 4.4-2 if you wish, and you may do so without proving it. Assume that $T(n)$ is constant for sufficiently small n.

a) $T(n) = 3T(n/9) + \sqrt{n}\lg^4 n$

b) $T(n) = 2T(n/3) + n\lg n$

c) $T(n) = T(n/3) + T(n/5) + T(n/7) + T(n/11) + T(n/13) + n$

d) $T(n) = T(\sqrt{n}) + 1$

Problem 2: Quick and Dirty Algorithms (14 points, 7 points each)

a) Provide an $\Theta(n)$-time algorithm that, given an integer array A of length n, prints all those elements between order statistics $\lceil n/4 \rceil$ and $\lceil 3n/4 \rceil$, inclusive. You should assume that all of the elements of A are distinct, and you needn't print the elements in any particular order.

b) Provide an $\Theta(n + k\lg n)$ algorithm that, given an integer array A of length n, partially sorts A so that the k smallest elements are sorted and placed in positions 1 through k, and that the k largest elements are sorted and placed in positions $n-k+1$ through n. The other elements—those ending up in positions $k+1$ through $n-k$—needn't be in any particular order. You should **not** assume that the elements of A are distinct.

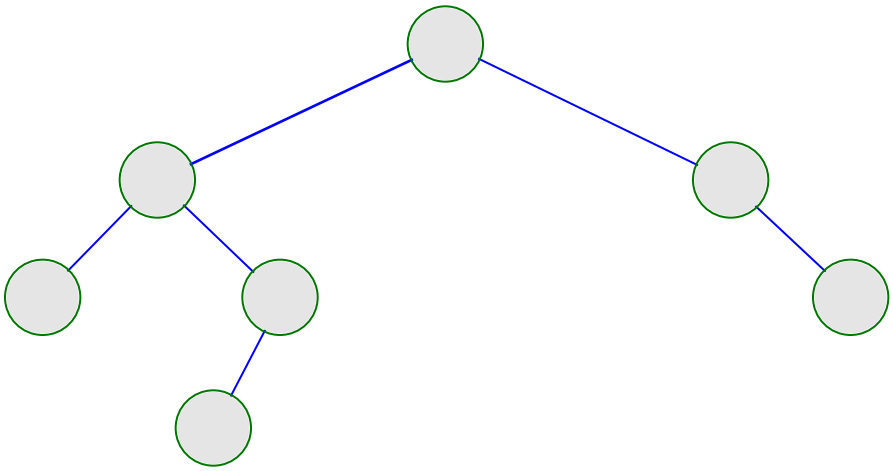
Problem 3: Sorting Hardware (16 points)

Professor Engler has developed a hardware priority queue for his computer. The priority queue device can store up to k records, each consisting of a key and a small amount of satellite data (such as a pointer). The computer to which it is attached can perform `Insert` and `Extract-Min` operations on the priority queue, each of which takes $\Theta(1)$, no matter how many records are stored on the device. The professor wishes to use the hardware priority queue to help implement a sorting algorithm on his computer. He has n records stored in the primary memory of his machine. If $n \le k$, the professor can certainly sort the keys in $\Theta(n)$ time by first

inserting all of them into the priority queue, and then repeatedly extracting the minimum. Design an efficient algorithm for sorting n items using the hardware priority queue, when $n > k$. Analyze your algorithm in terms of both k and n.

Problem 4: Transitioning from Binary Search Trees to Red-Black Trees (10 points, 3, 3, and 4)

a) Label the following binary search tree with numbers from the set $\{16,40,11,14,33,1,28\}$ so that it is a legal binary search tree.



b) Label each node in your tree above as either **Red** or **Black** so that the tree is also a legal red-black tree.

c) Make the left child of the root be the root by performing a single rotation. Draw the binary search tree that results, and label your tree with the keys from part **a)**. Is it possible to label the nodes with colors so that the new tree is a red-black tree? Justify you answer.