1. (3 points) BT&T (Big Telephone & Telegraph) has 256 million customers. Their current telephone directory consists of many heavy volumes typeset in small point size, which are expensive to print and inconvenient to use. To overcome the above problems, BT&T has decided to set up an on-line computerized directory, and their software engineers are debating what is the most efficient data structure for the purpose. Assume that the BT&T computer can compare two names in one microsecond. You do not need to simplify your calculations.

1a) One of the engineers suggests implementing the on-line directory as an unsorted linked list. With this implementation, give an estimate of the worst-case search and insertion times.

1b) Another engineer wants to implement the on-line directory as a red-black tree. With this implementation, give an estimate of the worst-case search and insertion times.

1c) A third engineer proposes the use of a sorted array. With this implementation, give an estimate of the worst-case search and insertion times.

2a) (2 points) Show each red-black tree that results after successively inserting the keys 4 7 12 15 3 5 14 18 into an initially empty red-black tree. At the steps were a red-black tree rule is violated, explain how it is corrected.

2b) (3 points) Now delete these keys in this order and show each resultant red-black tree 18 15 7 14. At the steps were a red-black tree rule is violated, explain how it is corrected.

3. (3 points) Exercise 13.1-1 In the style of Figure 13.1(a), draw the complete binary search tree of height 3 on the keys \{1, 2, ..., 15\}. Add the NIL leaves and color the nodes in three different ways such that the black-heights of the resulting red-black trees are 2, 3, and 4.

4. (2 points) 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?

5. (4 points) What is the upper bound on the height of an AVL tree containing $n$ keys. Prove your result.

6. (4 points) 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) marshaller/" 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 or English. What is the time complexity of your algorithm?

7. (4 points) In chapter 14 we saw 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 $x$. We also saw that the size at a node 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($x$), which returns the number of elements larger than $x$ in the red-black tree. Your function should work even if $x$ is not in the red-black tree. Show your function runs in $O(\log n)$ time.