Binary Search Trees (10 points)

1. [5 points] Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined?

   a. $2, 252, 401, 398, 330, 344, 397, 363.$

   b. $924, 220, 911, 244, 898, 258, 362, 363.$

   c. $925, 202, 911, 240, 912, 245, 363.$

   d. $2, 399, 387, 219, 266, 382, 381, 278, 363.$

   e. $935, 278, 347, 621, 299, 392, 358, 363.$

2. [2 points] An alternative method of performing an inorder tree walk of an n -node binary search tree finds the minimum element in the tree by calling TREE-MINIMUM and then making $n - 1$ calls to TREE-SUCCESSOR. Prove that this algorithm runs in $\Theta(n)$ time.

3. [3 points] We can sort a given set of n numbers by first building a binary search tree containing these numbers (using TREE-INSERT repeatedly to insert the numbers one by one) and then printing the numbers by an inorder tree walk. What are the worst-case and best-case running times for this sorting algorithm?

Red-Black Trees (5 points)

4. [4 points] Suppose that the black-height of each of the subtrees $\alpha, \beta, \gamma, \delta, \varepsilon$ in Figures 13.5 and 13.6 (both below) is k. Label each node in each figure with its black-height to verify that property 5 is preserved by the indicated transformation.