Assignment 14

R-3.11 Consider the following sequence of keys:

(5, 16, 22, 45, 2, 10, 18, 30, 50, 12, 1, 7, 55)

Consider the insertion of items with this set of keys, in the order given, into:

a. an initially empty (2,4) tree *T'*.
b. an initially empty red-black tree *T''.*

Draw *T'* and *T''* after each insertion.

R-3.14 For each of the following statements about red-black trees, determine whether it is true or false. If you think if it is true, provide a justification. If you think it is false, give a counterexample.

a. a subtree of a red-black tree is itself a red-black tree.
b. the sibling of an external node is either external or it is red.
c. given a red-black tree *T*, there is an unique (2,4) tree *T'* associated with *T*.
d. given a (2,4) tree *T*, there is an unique red-black tree *T'* associated with *T*.

1. Design a pseudo-code algorithm, **isPermutation(A,B),** that takes two Sequences A and B and determines whether or not they are permutations of each other, i.e., they contain same elements but possibly occurring in a different order. **Hint**: A and B may contain duplicates. Same problem as in previous homework, but this time use a dictionary to solve the problem.
2. What is the worst case time complexity of your algorithm? Justify your answer.
3. Design and solve this problem in four ways in JavaScript:
    a. By sorting A and B
    b. Using a Priority Queue
    c. Using a Hash Table based Dictionary
    d. Using a BST based Dictionary

4. Assume the elements in A and B cannot be sorted, i.e., there is no comparator. How would this restrict the way you would have to implement a solution to isPermutation(A,B), i.e., which of the above strategies could you use and which couldn't you use?
5. Which of the above strategies leaves the inputs A and B unchanged?
6. Are any of the approaches considered in-place?

7. Calculate the height of a Binary Tree. Implement your solution in the JavaScript file RBTree-HW.js that is provided. You are to do this both as a recursive function that traverses the tree and secondly using the Euler Tour template class (i.e., implement two different functions in JavaScript).

8. Calculate the black height of each node of a Red-Black Tree. Implement your solution in the JavaScript file RBTree-HW.js that is provided. You are to do this both as a recursive function that traverses the tree and secondly using the Euler Tour template class (again two different functions). There are two methods on a Red-Black tree to determine the color of a node, i.e., T.isRed(p) and T.isBlack(p). The black height for each node corresponds to the height of that key in a 2-4 Tree. The definition of the black-height of a node p, denoted bh(p), is the number of black nodes from p to every external node in the subtree rooted at p, but not including node p. See the lecture notes for more details and examples.