Exercise Session 08

Exercise 1.
(CLRS 12.3-1) Implement a recursive variant of the Tree-Insert procedure.

Tree-Insert(c, x)

if x.leftchild = null && x.key $\geq c.key$
    Insert(c, x.leftchild)
if x.rigthchild = null && x.key $\leq c.key$
    Insert(c, c.rigthchild)
if x.key $\geq c.key$
    Tree-Insert(c, x.leftchild)
if x.key $\leq c.key$
    Tree-Insert(c, x.rigthchild)

Exercise 2.
(CLRS 12.3-3) We can sort a sequence of n numbers by iteratively inserting each number in a binary search tree and then performing an inorder tree walk. Write the pseudocode of this algorithm. What are the worst-case and best-case running times for this sorting algorithm?

T is an empty tree

CreateandSort-Tree(A, T)

T.root = A[1]
for i = 2 to A.length
    Tree-Insert(A[i], T.root)
Inorder-Tree-Walk(T.root)

$$T(n) = c_1 + (n-1) \cdot (c_2 + c_3) + c_4$$
$$T(c) = 1 + n - 1 + (n-1) \cdot (n \cdot log(n)) + n$$
$$T(c) = n + n^2 log(n) + n = n^2 \cdot log(n)$$

Exercise 3.
Consider the binary search tree T depicted in Figure 2. Delete the node with *key = 10* from T by applying the procedure Tree-Delete(T, z) as described in CLRS.

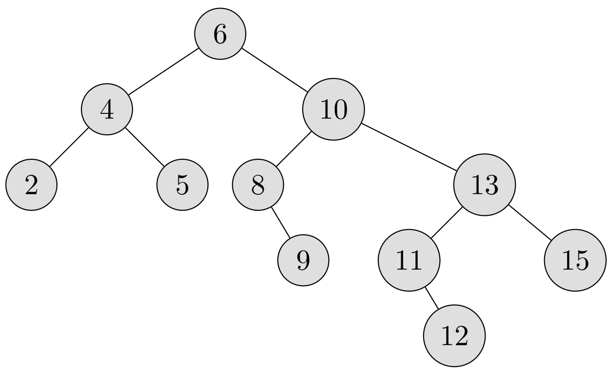

Figure 1: Binary Tree

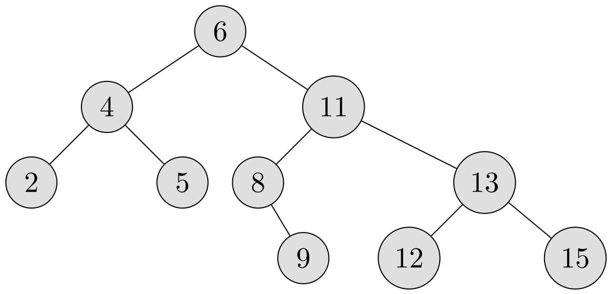

Figure 2: Binary Tree after deletion of 10

Exercise 4.
Show the red-black trees that result after successively inserting the keys 41; 38; 31; 12; 19; 8 into an initially empty red-black tree.

Exercise 5.
Consider the red-black tree T depicted in Figure 3. Insert first a node with $key = 15$ in T, then delete the node with $key = 8$. Show all the intermediate transformations of the red-black tree with particular emphasis on the rotations.

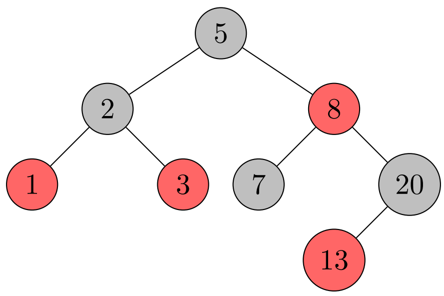

Figure 3: RB-Tree (Nil leaf nodes are omitted from the drawing)