German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher

Data Structures and Algorithms, Winter term 2020
Practice Assignment 9

Exercise 9-1 Insertion

Draw a binary search tree of the following elements, which are inserted in order:

a) `30 40 24 58 48 26 11 13`

b) `5 3 9 4 1 7 11`

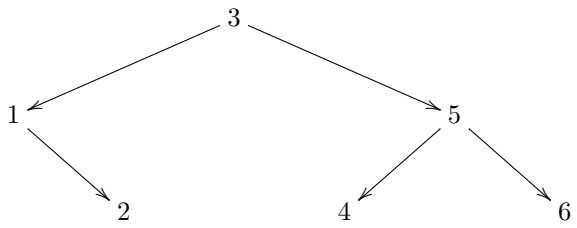
c) `9 8 7 4 1`

Exercise 9-2 Ideal Topology of Binary Search Tree

Given an ordered sequence of integers, write a Java program that returns a binary search tree with an ideal topology. For example for the sequence

```
1, 2, 3, 4, 5, 6
```

the Java program should output the tree with the following form:



Exercise 9-3 Max Key in a BST

Write a method that returns the largest item in a Binary Search Tree: `public Comparable maxKey()`

Exercise 9-4 Maximum of a Binary Tree

Write a **recursive** method that finds the maximum value contained in a binary tree of non-negative integers. If the tree is empty, then the method returns -1.

Exercise 9-5 Binary Search Tree

Add the following **recursive** methods to your binary search tree class:

a) `public int size()`
that returns the number of nodes in the tree.

b) `public int numLeaves()`
that returns the number of leaves in the tree.

c) `public int sum()`

that returns the sum of all the nodes in a tree of integers.

d) `public boolean isBST()`

which returns `true` if the tree is a binary search tree, and `false` otherwise.

e) `public int numLeftChildNodes()`

which returns the number of nodes having **a left child and no right child**.

f) `public int countOccur(Comparable key)`

which returns the number of occurrences of nodes in a binary tree with the value `key`.

g) `public boolean hasDups(Comparable key)`

which returns `true` if the tree has duplicates of the value `key`, i.e. occurs more than once anywhere in the tree, and `false` otherwise.

h) `public void mirror()`

that converts the binary tree into its mirror.

For example the mirror of the following tree

```
      4
     / \
    2   5
   / \
  1   3
```

is

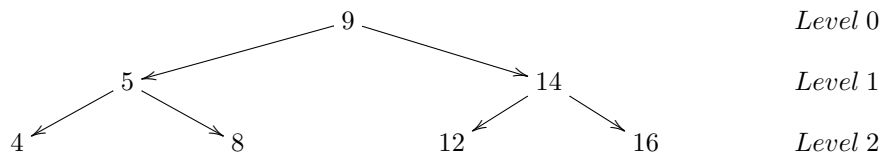
```
      4
     / \
    5   2
       / \
      3   1
```

i) `public String oddNodes()`

which returns a sequence (as a `String`) of the odd numbers occuring in a tree.

Exercise 9-6 Level of a Node

The **level** or **depth** of a node `n` in a tree `T` is the length of the unique path in `T` from its root to `n`. In particular, the root itself is at level 0 and its children are at level 1. Levels are demonstrated in the figure shown below. You are required to write a method `int level(Comparable key)` that returns the level of a given node within a binary tree using iteration and once more with recursion. If the given key does not exist in the tree, the method should return -1.



Exercise 9-7 Double Value of a Tree

Given a binary tree of integers, write a recursive method that returns a tree with all the original values doubled:

```
public BTree doubleValues()
```

Exercise 9-8 Identical Trees

Create a method that takes two binary trees and returns `true` if the trees are identical and `false` otherwise:

```
public boolean equal(BTree t2)
```