Übungsblatt 2

Ausgabe: 2.12.2017
Abgabe: 14.12.2017

Aufgabe 1: BinaryTree

Im ersten Übungsblatt haben Sie bereits die 3 bekannten Traversierungs-Arten für Binäre Bäume geschrieben.
Jetzt soll ein Binärer Suchbaum mit einer absoluten Ordnung (nehmen Sie dazu einfach an, dass int-Werte in Knoten gespeichert werden) implementiert werden, der folgende Funktionen bereitstellt:

`insert tree val`	Fügt `val` in den Baum `tree` ein und gibt als Ergebnis den ergänzten Baum zurück. Am besten wird dabei ein neuer Baum erzeugt.
`insert tree filename`	Fügt die `int`-Werte, die in der Datei stehen in den Baum ein.
`contains tree val`	Testet, ob `val` im Baum vorhanden ist.
`size tree`	Ermittelt die Anzahl der Knoten im Baum.
`height tree`	Ermittelt die Höhe des Baums.
`getMax tree`	Liefert das größte Element im Baum.
`getMin tree`	Liefert das kleinste Element im Baum.
`remove tree val`	Entfernt `val` aus dem Baum und gibt als Ergebnis den geänderten Baum zurück. Wenn ein innerer Knoten gelöscht wird, dann erstetzen Sie ihn durch den kleinsten Knoten in dessen rechtem Teilbaum.
`isEmpty tree`	`true` genau dann, wenn der Baum leer ist.
`addAll tree otherTree`	Fügt alle Elemente des übergebenen Baums (`otherTree`) in den aktuellen Baum `tree` ein.
`printLevelorder tree`	Gibt Baum in Levelorder aus.

Beschreiben Sie (im Kommentar), welche Ergebnisse herauskommen.

Achten Sie bei der Implementierung darauf, dass der Code möglichst kompakt und gut verständlich ist (dass er korrekt ist, versteht sich von selbst); Effizienz ist zweitrangig. Nutzen Sie Rekursion!.

Überlegen Sie, wie man die Eingabe möglichst komfortabel gestalten kann. D.h. wie könnte eine „Benutzerschnittstelle“ aussehen.