Aufgabe 3: Rekursion auf Bäumen

In dieser Aufgabe werden AVL-Bäume beleuchtet und damit die Rekursion vertieft. **Dabei sind die Verfahren zu verwenden, die in der Vorlesung vorgestellt wurden!**

Aufgabenstellung

Implementieren Sie:

1. ADT **AVLTree**:
   Vorgabe:
   Funktional (nach außen)
   1. Definition wie in der Vorlesung vorgestellt;
   2. Die Elemente sind vom Typ „ganze Zahl“.
   3. Duplikate von Elementen sind nicht zulässig.
   4. Alle für die ADT BTree bestehenden Funktionen müssen auch auf dem AVL Baum laufen und umgekehrt.

   Technisch (nach innen)
   1. Die ADT AVLTree ist mittels ADT BTree zu realisieren, indem diese erweitert wird.
   2. Analog zu `isBT` sind bei `insertBT` und `deleteBT` der Baum nur einmal von oben nach unten zu durchlaufen und wegen der Rekursion dann einmal von unten nach oben. Lediglich beim Löschen darf einmal von der löschenden Position zu einem Blatt und zurück zusätzlich gelaufen werden.
   3. Die zugehörige Datei heißt avltree.erl

   Objektmengen: elem, btree, dot
   Operationen: (semantische Signatur) / (syntaktische Struktur)

initBT: ∅ → btree	/ `initBT()`
isEmptyBT: btree → bool	/ `isEmptyBT(<BTree>)`
equalBT: btree × btree → bool	/ `equalBT(<BTree>,<BTree>)`
isBT: btree → bool	/ `isBT(<BTree>)`
insertBT: btree × elem → btree	/ `insertBT(<BTree>,<Element>)`
deleteBT: btree × elem → btree	/ `deleteBT(<BTree>,<Element>)`
printBT: btree × filename → dot	/ `printBT(<BTree>,<Filename>)`

   dot ist dabei eine dot-Datei in dem von GraphViz verwendetem Dateiformat. In der Konsole kann dann z.B. per "dot -Tpng avltree.dot > avl.png" die png-Datei erzeugt werden, wobei png das bekannte Portable Network Graphics Format ist.
   Die Funktionen aus Praktikumsaufgabe 1 sind zu übernehmen und ggf. anzupassen.
2. Erweitern Sie die ADT so, dass die Anzahl der linken und rechten Rotationen gezählt werden (Doppelrotationen zählen entsprechend bei beiden! und diese auch separat zählen!). Verwenden Sie dazu den globalen Zähler aus util.erl mit den Namen leftrotate, rightrotate, ddleftrotate, ddrightrotate.
3. Implementieren Sie eine Testumgebung, in der Sie die Laufzeit messen können. Um viele Zahlen verwenden zu können, nutzen Sie die Funktion randomliste/1 aus der Datei util.erl. Implementieren Sie zudem einen Test, der eine vorgegebene Anzahl an Zufallszahlen einfügt und den Baum danach mittels printBT ausgibt. Dann sind von diesen Zufallszahlen 42% zu löschen. Der Baum ist erneut mittels printBT auszugeben. Dokumentieren und interpretieren Sie die Ergebnisse. Nehmen Sie von einem anderen Team die avltree.beam und führen Sie die Tests erneut durch. Vergleichen Sie die Laufzeiten und vergleichen Sie die Bäume. Interpretieren Sie die Ergebnisse.

Nützliche Hilfsfunktionen zum Zählen oder der Zeitmessung finden Sie in der Datei util.erl.

Abnahme

Bis Montag Abend 20:00 Uhr vor Ihrem Praktikumstermin ist ein erster **Entwurf** für die Aufgabe **als *.pdf** Dokument (Dokumentationskopf nicht vergessen!) mir per E-Mail (mit cc an den/die Teampratner_in) zuzusenden.

Am Tag vor dem Praktikumstermin bis **20:00 Uhr** : bitte finaler Stand (als *.zip) zusenden, der in der Vorführung am Anfang des Praktikums eingesetzt wird und alle Vorgaben erfüllen muss.

Am Tag des Praktikums findet eine Besprechung mit Teams statt. Die **Besprechung muss erfolgreich absolviert werden**, um weiter am Praktikum teilnehmen zu können. Ist die Besprechung nicht erfolgreich, gilt die Aufgabe als nicht erfolgreich bearbeitet. Als erfolgreich wird die Besprechung bewertet, wenn Ihre Kenntnisse eine erfolgreiche weitere Teilnahme an dem Praktikumstermin in Aussicht stellen. Bei der Besprechung handelt es sich nicht um die Abnahme.

Zum konkreten Zeitplan im Praktikum: Im Zeitraum **12:35 - 13:05** finden Vorführungen statt, d.h. sie demonstrieren ihren lauffähigen Code z.B. mittels Tests. Danach finden die Besprechungen statt. Im Zeitraum ab **13:15 - 15:40** (siehe Tabelle der Prüfungstermine) finden die Referate statt.

Abgabe: Unmittelbar am Ende des Praktikums ist von **allen Teams** für die **Abgabe** der erstellte und gut dokumentierte Code abzugeben. Zu dem Code gehören die Sourcedateien und eine Readme.txt Datei, in der ausführlich beschrieben wird, wie das System zu starten ist! Des weiteren ist der aktuelle Dokumentationskopf abzugeben. **In den Sourcedateien ist auf den Entwurf zu verweisen**, um die Umsetzung der Vorgaben zu dokumentieren. Zudem sind die **Ergebnisse aus dem Aufgabenpunkt 3. in einer *.pdf Datei** mit abzugeben. Alle Dateien sind als **ein *.zip Ordner** (mit cc an den/die Teampratner_in) per E-Mail an den genannten Verteiler abzugeben. Die Abgabe gehört zu den PVL-Bedingungen und ist einzuhalten, terminlich wie auch inhaltlich!

Wird eine Aufgabe nicht erfolgreich bearbeitet gilt die **PVL** als **nicht bestanden**. Damit eine Aufgabe als erfolgreich gewertet wird, müssen der Entwurf, die Besprechung, die Vorführung sowie die Abgabe als erfolgreich gewertet werden. Ob die Abgabe erfolgreich abgenommen ist, wird Ihnen per E-Mail mitgeteilt. **Alle gesetzten Termine sind einzuhalten**.