**Team**: < 04 >, < Schäfer & Ahmad >

**Aufgabenaufteilung**:

1. <Aufgaben, für die Teammitglied 1 verantwortlich ist>,   
   <Dateien, die komplett/zum Teil von Teammitglied 1 implementiert/bearbeitet wurden>
2. <Aufgaben, für die Teammitglied 2 verantwortlich ist>,   
   <Dateien, die komplett/zum Teil von Teammitglied 2 implementiert/bearbeitet wurden>

**Quellenangaben**: <

>

**Begründung für Codeübernahme**: < Es wurde kein Code übernommen. >

**Bearbeitungszeitraum**: <

Für den Entwurf: 2 Stunden 15.11.2014

1 Stunde 21.11.2014

http://users.informatik.haw-hamburg.de/~klauck/AlguDat/AD.pdf

>

**Aktueller Stand**: < Entwurf fertig >

**Änderungen im Entwurf**: < KEINE >

***Implementieren sie die ADT AVL-Baum reskursiv:***

Dieser soll als höhenbalancierter binärer Suchbaum realisiert werden, gemäss der Definition aus der Vorlesung.   
(Siehe Quelle)

***Was ist ein höhenbalancierter binärer Suchbaum?***

Definition AVL-Baum: Ein binärer Suchbaum heißt **AVL-Baum** oder höhenbalancierter Baum, wenn für jeden Knoten *v* gilt, dass sich die Höhe des rechten Teilbaumes ***h*(*Tr*)** von *v* und die Höhe des linken Teilbaumes ***h*(*Tl*)** von *v* um maximal 1 unterscheiden.

*bal(v) = h(Tr) – h(Tl)* ∈ {-1,0,1}

***Es sind folgende Funktionen für die ADT AVL-Baum zu implementieren, die nach Außen geliefert werden!***

1. AVL-Baum Initialisieren
2. Einfügen
3. Löschen
4. Linksrotation
5. Rechtsrotation
6. Doppellinksrotation
7. Doppelrechtsrotation

Alle diese Funktionen sollen den modifizierten AVL-Baum zurück liefern!

***Die Signaturen der zu implementierenden Funktionen:***

*%% Erstellt einen AVL-Baum mit einen Wurzel Knoten*

*%% @param Integer Root - Die Wurzel des AVL Baumes*

*init(Root) ->*

*Funktion haengt genau ein Knoten an einen Baum an.*

*@param avlBaum Tree - Die Datenstruktur des AVL Baumes.*

*@param String Key - Das neu hin zugefuegte Element.*

einfuegen(Tree, Key) ->

*Diese Funktion entfernt einen Knoten aus einem Baum.*

*@param avlBaum Tree - Der AVL-Baum auf dem ein Schluessel entfernt werden soll*

*@param Integer Key - Der zu entfernende Schluessel.*

*@return avlBaum - Der modifizierte Baum.*

loeschen(Tree, Key) ->

*Diese Funktion implementiert die einfache Linksrotation.*

*@param avlBaum Tree - Der Baum in den rotiert werden soll.*

*@param Integer Key - Der Vertex um den rotiert werden soll.*

linksrotation(Tree, Key) ->

*Diese Funktion implementiert die einfache Rechtsrotation.*

*@param avlBaum Tree - Der Baum in den rotiert werden soll.*

*@param Integer Key - Der Vertex um den rotiert werden soll.*

rechtsrotation(Tree, Key) ->

*Diese Funktion implementiert die doppelte Linksrotation.*

*@param avlBaum Tree - Der Baum in den rotiert werden soll.*

*@param Integer Key - Der Vertex um den rotiert werden soll.*

doppelLinksrotation(Tree, Key) ->

*Diese Funktion implementiert die doppelte Rechtsrotation.*

*@param avlBaum Tree - Der Baum in den rotiert werden soll.*

*@param Integer Key - Der Vertex um den rotiert werden soll.*

doppelRechtsrotation(Tree, Key) ->

**Tipps:**

- Sie können bei aufrufen der 7 Funktionen die nach außen   
geliefert werden, jedes mal ein Bild generieren mit Hilfe von Graphviz, dies dient der besseren Kontrolle, ob der zu implementierende Algorithmus richtig funktioniert.