Team participants:

Yitian Chen,
3546325, tallgeezee@gmail.com

Yuecheng Yao,
3561339, yychen.g@qq.com

Shunyi Deng,
3562956, 2545151930@qq.com

Universität Stuttgart

Institut für Visualisierung
und Interaktive Systeme

Übungsblatt 3

Datenstrukturen und Algorithmen (SS 2021)

Abgabe: Montag, 17.05.2021, 12:00 Uhr — Besprechung: ab Montag, 31.05.2021

Bitte lösen Sie die Übungsaufgabe in **Gruppen von 3 Bearbeiter*innen** und wählen Sie EINE Person aus der Gruppe aus, welche die Lösung in ILIAS als **PDF** als **Gruppenabgabe** (unter Angabe aller Gruppenmitglieder) einstellt. Bitte erstellen Sie dazu ein **Titelblatt**, das die Namen der Bearbeiter*innen, die Matrikelnummern, und die E-Mail-Adressen enthält.

Die Aufgaben mit Implementierung sind mit Impl gekennzeichnet. Das entsprechende Eclipse-Projekt kann im ILIAS heruntergeladen werden. Bitte beachten Sie die Hinweise zu den Implementierungsaufgaben, die im ILIAS verfügbar sind.[1]

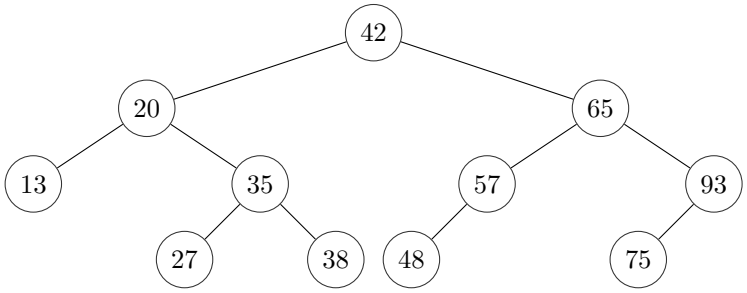
Dieses Übungsblatt beinhaltet 4 Aufgaben mit einer Gesamtzahl von 30 Punkten.

Aufgabe 1 Impl Iterator *[Punkte: 10]*

Gegeben im Eclipse-Projekt sind die Schnittstellen `ISimpleList` und `ISimpleListIterable`. Erweitern Sie die Klasse `SimpleList` (ohne die bestehenden Methoden zu modifizieren), damit sie die Schnittstelle `ISimpleListIterable` implementiert. Erstellen Sie hierzu zwei Iterator-Klassen als innere Klassen der Klasse `SimpleList`. **Die Verwendung existierender Iteratorimplementierungen (z.B. `ArrayList.iterator()` sowie Iteratoren anderer Datenstrukturen aus der Java-Klassenbibliothek) ist dabei nicht erlaubt.** Die Methode `remove` der Iteratoren wird nicht unterstützt und sollte immer `UnsupportedOperationException` werfen. Sie können davon ausgehen, dass die Liste während der Verwendung des Iterators nicht verändert wird.

Aufgabe 2 Binäre Suchbäume, AVL-Bäume *[Punkte: 8]*

Gegeben sei der **binäre Suchbaum** B_1:



(a) *(1 Punkte)* Erstellen Sie den Baum B_2, indem Sie in B_1 die Werte 19 und 60 einfügen.

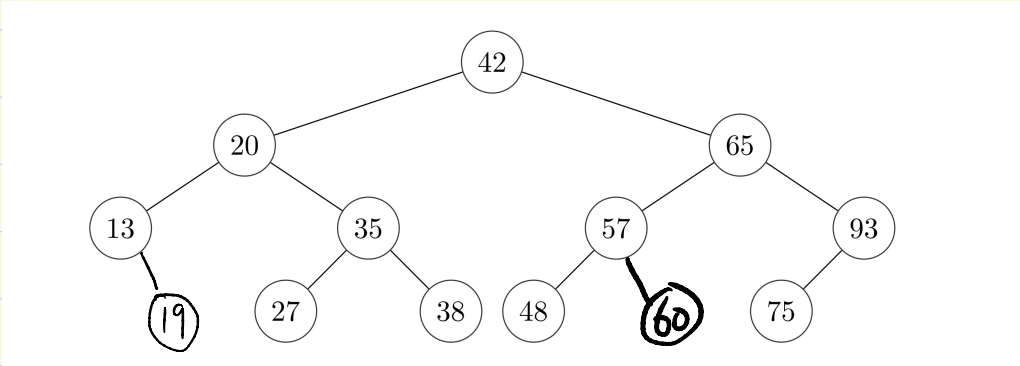
(b) *(1 Punkte)* Erstellen Sie den Baum B_3, indem Sie aus dem ursprünglichen Baum B_1 die Werte 27 und 65 entfernen. Ersetzen Sie, wo nötig, Knoten durch ihre **Inorder-Nachfolger**.

(c) *(1 Punkte)* Geben Sie für jeden Knoten aus dem ursprünglichen Baum B_1 den Wert der AVL-Balance an.

[1] `https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_crs_2348772.html`

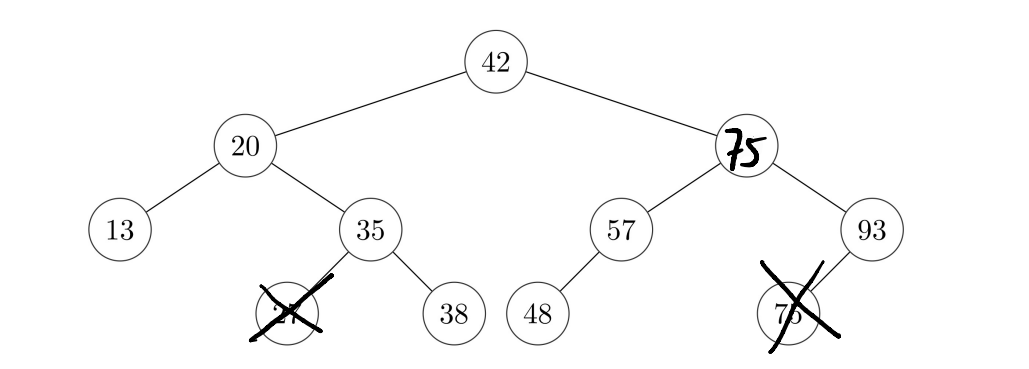
a)

B_2



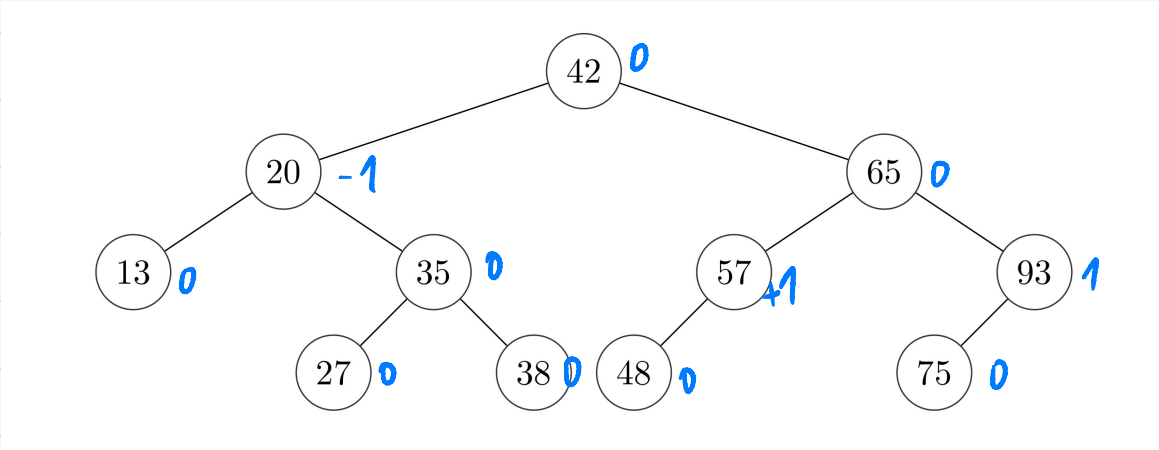
+19
+60

b) B_3

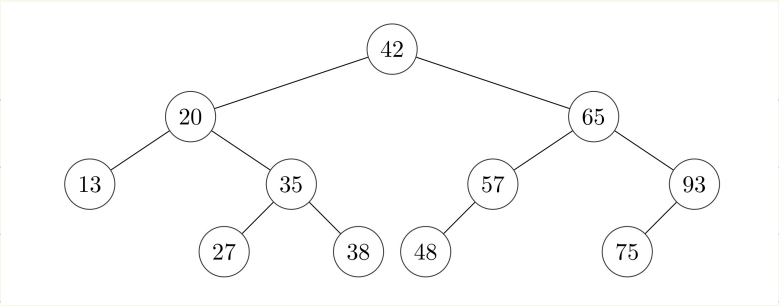


-27
-65

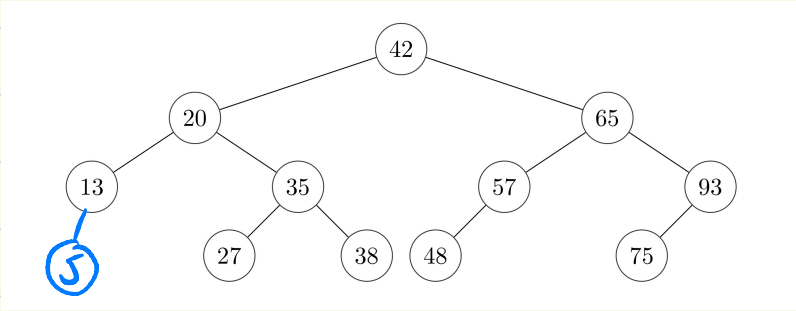
c)



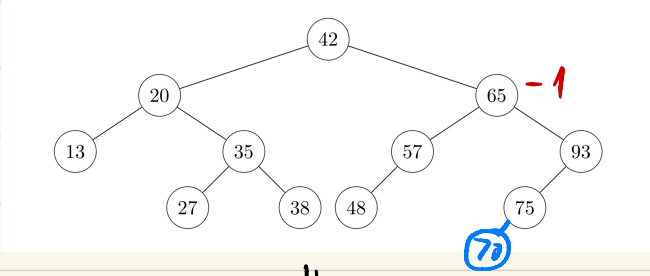
d)



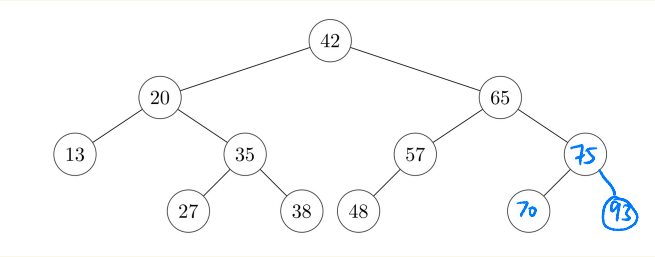
B4:



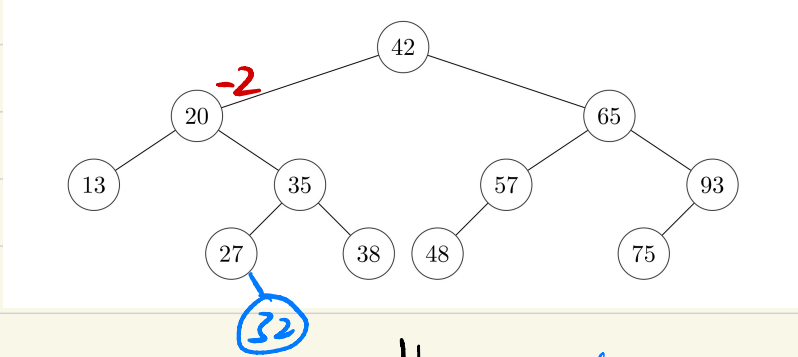
B5:



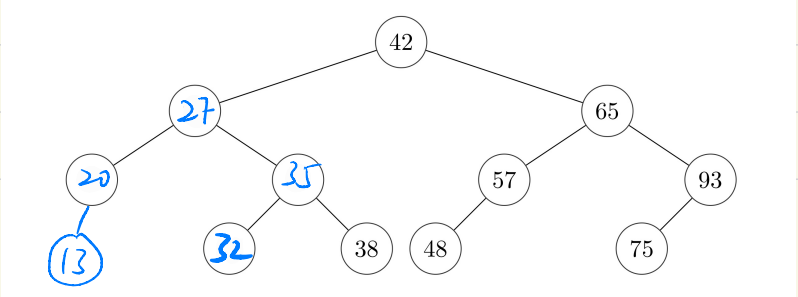
⇓ R(75, 93)



B6:



⇓ DR(20, 27, 35)



(d) *(5 Punkte)* Erstellen Sie folgende **AVL-Bäume** durch Einfügen von Werten in den **ursprünglichen** Baum B_1:

- B_4 durch Einfügen von 5 in B_1
- B_5 durch Einfügen von 70 in B_1
- B_6 durch Einfügen von 32 in B_1

Verwenden Sie Zwischenergebnisse um zu zeigen welche etwaigen Schritte notwendig sind um die AVL-Balance wiederherzustellen. Nennen Sie auch, welche Rotation um welchen Knoten durchgeführt wurde.

Aufgabe 3 Impl Binäre Suchbäume *[Punkte: 8]*

Im Eclipse-Projekt sind folgende Codefragmente für binäre Suchbäume gegeben:

- Schnittstelle `IBinaryTreeNode` und Klasse `BinaryTreeNode`
- Schnittstelle `IBinarySearchTree` und unvollständige Klasse `BinarySearchTree`

(a) *(8 Punkte)* Vervollständigen Sie die Klasse `BinarySearchTree`. Sie muss `IBinarySearchTree` implementieren und einen Standard-Konstruktor (d.h. ohne Parameter) enthalten.

Ignorieren Sie Duplikate! Wenn z.B. ein Schlüsselwert eingefügt werden soll, welcher im Baum bereits vorhanden ist, dann soll nichts passieren. Ignorieren Sie in solchen Fällen den Versuch des Einfügens.

Aufgabe 4 Binäre Suchbäume *[Punkte: 4]*

Gegeben ist die folgende Menge von Schlüsselwerten: 35, ~~5~~, ~~16~~, ~~98~~, 54, ~~59~~, ~~70~~, ~~12~~, ~~76~~, ~~20~~, ~~8~~, ~~62~~, ~~24~~, ~~85~~, ~~13~~, ~~63~~, ~~49~~, ~~29~~, ~~84~~.

Zeichnen Sie den dazugehörigen *Binären Suchbaum*, so dass dessen *Preorder Traversierung* mit der Sequenz 49, 16, 12, 8, 5, 13, 29, 24, 20, 35, 70 *beginnt* und dessen *Postorder Traversierung* mit der Sequenz 62, 63, 59, 76, 85, 84, 98, 70, 49 *endet*.