Aufgabe 1 (Hashing) *25 Punkte*

Fügen Sie die unten angegebene Folge von Strings in eine Hashtabelle der Größe 10 ein. Verwenden Sie als Hashfunktion die Mittel-Quadrat-Methode und Quadratisches Sondieren als Kollisionsstrategie. Um den initialen Zahlenwert für die Mittel-Quadrat-Methode zu erhalten, summieren Sie die Buchstabencodes (’a’ = ’A’ = 1, ’b’=’B’=2 usw.) der ersten drei Buchstaben der Zeichenkette auf. Bei einer geraden Anzahl von Ziffern, wählen Sie bitte die weiter rechts stehende Zahl als mittlere Zahl aus. Geben Sie für jeden Eintrag den initialen Hashwert sowie ggf. die Folge der Positionen an, bei denen eine Kollision auftritt. Geben Sie letztlich die gefüllte Hashtabelle an.

Heidrun, Anne, Hartmut, Christian, Simone, Frank, Markus, Thomas

Aufgabe 2 (Bulkloading für Binäre Suchbäume) *20 Punkte*

Als „Bulkloading“ bezeichnet man das Einfügen einer großen Menge neuer Einträge in eine Datenstruktur „auf einen Schlag“. Auf diese Weise ist es häufig möglich, eine Datenstruktur kosteneffizienter zu aktualisieren, als durch eine lange Folge normaler Einfügeoperationen.

In diesem Fall wollen wir einen besonderen Fall des Bulkloading betrachten: die initiale Erstellung eines binären Suchbaums anhand einer bereits sortiert vorliegenden Wertemenge.

Gegeben sei ein Array A der Größe n, das n verschiedene Schlüsselwerte in aufsteigende sortierter Reihenfolge enthält, d.h. $A[1] < A[2] < \ldots < A[n]$.

Einen leeren binären Suchbaum stellen wir durch das Symbol *empty* dar, einen nicht leeren Suchbaum durch ein Tripel (L, k, R), wobei L und R binäre Suchbäume sind und k ein Schlüsselwert ist.

(a) Geben Sie einen Algorithmus an, der aus dem Array A in $O(n)$ Zeit einen balancierten binären Suchbaum minimaler Höhe erzeugt. *10 Punkte*

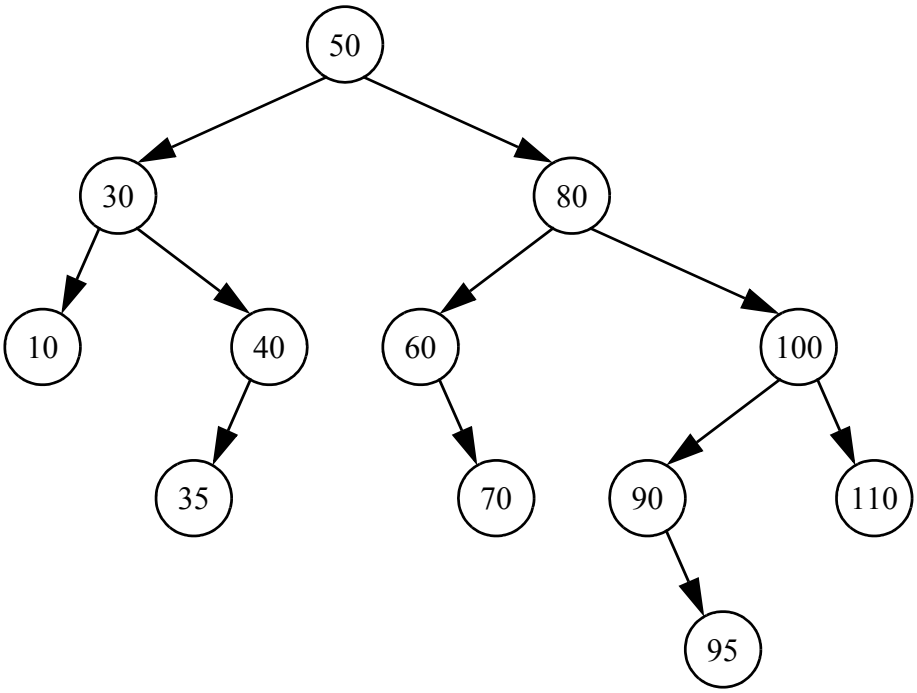
(b) Beweisen Sie, dass Ihr Algorithmus aus Teilaufgabe (a) Laufzeit $O(n)$ hat. Es genügt, wenn Sie die Aussage für eine geeignete Teilfolge der natürlichen Zahlen beweisen. *10 Punkte*

30 Punkte **Aufgabe 3 (AVL-Baum)**

15 Punkte (a) Fügen Sie die unten angegebene Zahlenfolge in einen anfangs leeren AVL-Baum ein. Zeichnen Sie den Baum vor und nach jeder Balancierungsoperation.

20, 10, 15, 30, 40, 18

5 Punkte (b) Löschen Sie aus dem unten angegeben AVL-Baum nacheinander die folgenden Elemente in der angegebenen Reihenfolge. Zeichnen Sie den Baum vor und nach jeder Balancierungsoperation.



Elemente: 10, 110

10 Punkte (c) Geben Sie einen Algorithmus an, der für einen gegebenen Wert *v* den nächstkleineren Wert bestimmt, der in einem AVL-Baum enthalten ist. Der Algorithmus soll unabhängig davon sein, ob *v* im Baum existiert oder nicht. Ist *v* bereits das kleinste im Baum gespeicherte Element oder ist dieser leer, so soll der Algorithmus einen Fehler zurückliefern. Der Algorithmus soll das Ergebnis in logarithmischer Zeit liefern. Begründen Sie die Laufzeitschranke, ein formaler Beweis ist nicht erforderlich.

Aufgabe 4 (Heap) ***25 Punkte***

(a) Fügen Sie die unten angegebene Folge in einen Maximum-Heap ein. Zeichnen Sie den Baum nach jeder Einfügeoperation. *15 Punkte*

3 - 4 - 1 - 8 - 10 - 19 - 2 - 5 - 7 - 16

(b) Löschen Sie fünfmal hintereinander das Maximum aus dem in Aufgabenteil (a) berechneten Heap. Geben Sie den Baum nach jeder Löschoperation an. *10 Punkte*