1. (10 Punkte)

   (a) Zeichnen Sie den Baum auf, der sich ergibt, wenn man nacheinander

   $$11 \quad 5 \quad 26 \quad 24 \quad 30 \quad 21 \quad 7 \quad 17 \quad 14$$

   in dieser Reihenfolge in einen anfangs leeren AVL Baum einfügt. Bitte zeichnen Sie die Bäume aller Zwischenschritte auf.

   (b) Welche Bäume erhält man, wenn man in den eben produzierten AVL Baum den Schlüssel 5 zuerst löscht und dann diesen Schlüssel wieder einfügt?

2. (15 Punkte) Das Löschen eines Knotens x in einem binären Suchbaum T ist einfach zu bewerkstelligen, wenn x höchstens ein Kind v hat: der Teilbaum T_x mit Wurzel x wird einfach durch den Teilbaum T_v mit Wurzel v ersetzt (der Zeiger, der auf x zeigt, wird auf v umgesetzt).

   Wenn x beide Kinder hat, muss man sich anders behelfen. Der in der Vorlesung gezeigte Delete-Algorithmus nutzt aus, das in diesem Fall der Schlüsselnachfolgerknoten z von x in T definitiv kein linkes Kind hat. Es wird also z auf oben genannte Art entfernt, nachdem der Informationsinhalt von z in den Knoten x transferiert worden ist.

   Alternativ ist aber auch folgendes möglich:

   Wenn x nun zwei Kinder hat, dann forme den Suchbaum T durch Rotationen so lange um, bis x nur mehr ein Kind hat, und nimm dann diese Teilbaumersetzung vor. Das folgende Codestück versucht, so eine Umformung zu vollführen:

   ```
   while x.right ≠ TNULL do
   |  Rotate-Left(x)
   end
   ```

   (a) Beschreiben Sie in einem Satz, möglicherweise unter Verwendung eines Bildes, die strukturelle Umformung von T, die durch den gerade skizzieten rotationsbasierten Löschalgorithm erzielt wird. Verwenden Sie dabei die Teilbäume T_x, $T_{x.\text{left}}$ und $T_{x.\text{right}}$.

   (b) Geben Sie einen rotations-basierten Löschalgorithmus an, der strukturell genau den gleichen Baum erzeugt, wie der in der Vorlesung gezeigte Delete-Algorithmus.

3. (10 Punkten)

   (a) Geben Sie eine Familie von Beispielen an, die zeigt, dass bei einer Löschoperation in AVL-Bäumen es zu logarithmisch vielen Rotationen kommen kann.

   (b) Wie viele Rotationen kann es höchstens beim Einfügen geben?

Wählen Sie eine der beiden folgenden Aufgaben aus. Sie bekommen nur Punkte für eine der beiden Aufgaben, und es wird auch nur eine davon korrigiert.

4A. (15 Punkte) In dieser Aufgabe betrachten wir nur endliche Mengen A von reellen Zahlen mit der Eigenschaft, dass $0 \notin A$ und dass keine zwei Elemente von A den gleichen Betrag haben, also für $\{a, b\} \subset A$ gilt immer $|a| \neq |b|$. Nennen wir solche Mengen "hübsch". Für so eine hübsche Menge A mit $|A| = n$ definieren wir $VZW(A)$, die *Anzahl der Vorzeichenwechsel* von A, folgendermaßen: Ordne A nach dem Betrag, also

$$|a_1| < |a_2| < \cdots < |a_n| .$$

$VZW(A)$ ist dann die Anzahl der i, mit $1 \leq i < n$, sodass $a_i \cdot a_{i+1} < 0$, also a_i und a_{i+1} haben verschiedene Vorzeichen.

Für eine endliche Menge S von reellen Zahlen und zwei Zahlen $0 \leq \ell < r$ definiere $S[\ell, r] = \{s \in S \mid \ell \leq |s| \leq r\}$.

Enwickeln Sie eine Methode, die eine hübsche Menge S unter den Operationen Einfügen und Löschen aufrecht erhält, und die folgende Suchanfrage beantworten kann: bei gegebenem $\ell < r$ soll die Zahl $VZW(S[\ell, r])$ bestimmt werden.

Jede der drei Operationen sollte im schlechtesten Fall nur $O(\log n)$ viel Zeit brauchen.

4B. (15 Punkte) Es sei T_1 ein AVL-Baum für eine Schlüsselmenge S_1 und T_2 ein AVL-Baum für S_2. Es gelte, dass jeder Schlüssel in S_1 kleiner ist als jeder Schlüssel in S_2.

Geben Sie eine Methode an, die aus T_1 und T_2 einen AVL-Baum T für $S = S_1 \cup S_2$ erzeugt. Die beiden Ausgangsbäume T_1 und T_2 dürfen dabei zerstört werden.

Welche Laufzeit können Sie (abhängig von der Höhe der Bäume T_1, T_2) erzielen?