Aufgabe 1: Sortierverfahren (5 Punkte)

Sortieren Sie die folgende Liste von Zahlen aufsteigend mit dem Verfahren *Insertion Sort*:

38 18 5 21 29 14 35

Geben Sie die Liste nach jedem Durchlauf der inneren Schleife an, d.h. nach jedem vollständigen Einsortieren eines Elements.

Lösung

Die folgenden Zwischenergebnisse entstehen bei einem In-Place durchgeführten Insertion Sort. Sortierter und unsortierter Teil der Liste sind jeweils durch einen senkrechten Strich (|) getrennt:

38	|	18	5	21	29	14	35
18	38	|	5	21	29	14	35
5	18	38	|	21	29	14	35
5	18	21	38	|	29	14	35
5	18	21	29	38	|	14	35
5	14	18	21	29	38	|	35
5	14	18	21	29	35	38	|

Aufgabe 2: AVL-Bäume (5 Punkte)

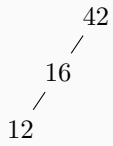
Fügen Sie die folgenden Zahlen nacheinander in einen *AVL-Baum* ein:

$$42 \quad 16 \quad 12 \quad 19 \quad 38 \quad 1$$

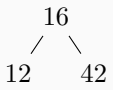
Zeichnen Sie den Baum vor und nach jeder durchgeführten Rotation. Geben Sie auch jeweils an, was für Rotationen Sie durchführen.

Lösung

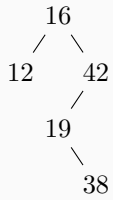
Einfügen von $42, 16, 12$



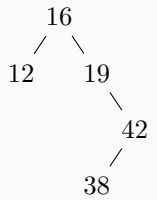
R-Rotation um 42



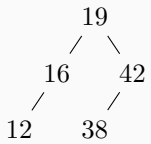
Einfügen von $19, 38$



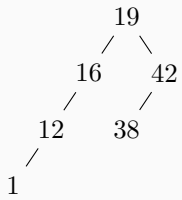
RL-Rotation um 42 (1)



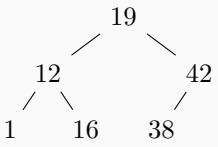
RL-Rotation um 42 (2)



Einfügen von 1



R-Rotation um 16



Aufgabe 3: Heaps (5 Punkte)

Fügen Sie die folgenden Zahlen nacheinander in einen *Min-Heap* ein:

45 18 19 7 13 2 22

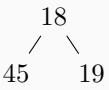
Zeichnen Sie den Baum vor und nach jedem vollständigen Einfügen.

Lösung

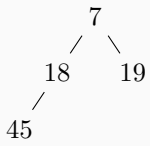
Einfügen von 45

Einfügen von 18

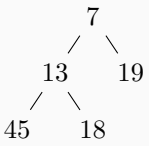
Einfügen von 19



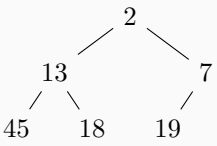
Einfügen von 7



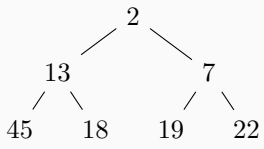
Einfügen von 13



Einfügen von 2



Einfügen von 22



Aufgabe 4: Sortierverfahren (5 Punkte)

Erläutern Sie die Funktionsweise des folgenden Sortierverfahrens. Erläutern Sie auch, welche Einschränkungen bzw. Anforderungen an die Liste gelten müssen, damit das Verfahren korrekt funktioniert.

```
1     public static void foosort(List<Integer> list) {
2         if (list.size() == 0) {
3             return;
4         }
5         int max = list.get(0);
6         for (int i = 1; i < list.size(); i++) {
7             if (list.get(i) > max) {
8                 max = list.get(i);
9             }
10        }
11        List<Integer> values = new ArrayList<>();
12        for (int i = 0; i <= max; i++) {
13            values.add(0);
14        }
15        for (int i = 0; i < list.size(); i++) {
16            values.set(list.get(i), values.get(list.get(i)) + 1);
17        }
18        int k = 0;
19        for (int i = 1; i < values.size(); i++) {
20            for (int j = 0; j < values.get(i); j++) {
21                list.set(k, i);
22                k++;
23            }
24        }
25    }
```

Lösung

Das Verfahren bestimmt zuerst das größte Element der Liste. Dann zählt es für jeden Wert der Liste, wie oft er vorkommt. Diese Häufigkeiten werden in einem Hilfsarray gespeichert.
Die sortierte Liste wird anschließend wieder aufgebaut, indem das Hilfsarray durchlaufen wird und dabei entsprechend jeder Häufigkeit die Elemente der Liste gesetzt werden.