## Aufgabe 1: AVL-Bäume (10 Punkte)

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

55 16 12 19 38 42

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

**Aufgabe 2: Datenstrukturen (Entwurf)** **(10 Punkte)**

Für ein Logging-System soll eine Datenstruktur entworfen werden, die es ermöglicht, eine große Menge an Strings zu speichern und von dort aus in Dateien zu schreiben. die Datenstruktur soll sozusagen als Puffer fungieren. Die Datenstruktur soll dabei Platz für eine fest Anzahl an Strings bieten, z.B. 1000. Werden die Meldungen nicht schnell genug abgearbeitet, darf die älteste Meldung gelöscht werden.
Eine einfache Lösung wäre, ein Array mit einer festen Größe zu verwenden. Dies wäre zwar leicht umsetzbar, hat aber einige Nachteile bgzl. Performance und Erweiterbarkeit. Analysieren Sie diese Lösung und machen Sie Verbesserungsvorschläge:

- Welche Performance-Probleme können sich durch den o.g. einfachen Ansatz ergeben?
- Wie können diese Probleme gelöst werden?
- Diskutieren, inwiefern Ihre Datenstruktur die Probleme löst und welche neuen Nachteile sie möglicherweise hat.

*Hinweis:* Sie müssen keinen Code für die Datenstruktur angeben, eine Erklärung der Idee genügt.

## Aufgabe 3: Komplexität (10 Punkte)

Betrachten Sie die folgende Funktion `SameElements()`. Die Funktion erwartet zwei `int`-Listen und prüft, ob diese beiden Listen die gleiche Menge an Elementen enthalten. D.h. ob jedes Element aus der einen Liste auch in der anderen enthalten ist. Dabei spielt es keine Rolle, ob die Länge der Listen gleich ist bzw. ob die Elemente in der gleichen Anzahl vorkommen.

```
1  bool same_elements(std::vector<int> list1, std::vector<int> list2) {
2    for (int el : list1) {
3      bool contained = false;
4      for (int el2 : list2) {
5        if (el == el2) {
6          contained = true;
7        }
8      }
9      if (!contained) {
10       return false;
11     }
12   }
13
14   for (int el : list2) {
15     bool contained = false;
16     for (int el2 : list1) {
17       if (el == el2) {
18         contained = true;
19       }
20     }
21     if (!contained) {
22       return false;
23     }
24   }
25   return true;
26 }
```

a) Bestimmen Sie die Komplexität dieser Funktion. Geben Sie in O-Notation an, wie oft die Vergleiche `if v1 == v2` durchgeführt werden. Begründen Sie Ihr Ergebnis.

b) Machen Sie einen Vorschlag, wie diese Funktion optimiert werden kann. Erläutern Sie, inwiefern dieser eine bessere Komplexität hat.

*Hinweis:* Sie müssen keinen konkreten, vollständigen Algorithmus angeben.

## Aufgabe 4: Datenstrukturen (Funktionsweise) (10 Punkte)

Erläutern Sie die Idee und Funktionsweise des folgenden Datentyps. Diskutieren Sie kurz die Vor- und Nachteile gegenüber ähnlichen Datentypen aus der Vorlesung.

```
struct FooType {
  std::vector<int> values;
  std::vector<FooType *> children;

  void AddToChild(int i, int value) {
    while (children.size() <= i) {
      children.push_back(new FooType());
    }
    children[i]->Add(value);
  }

  void Add(int value) {
    if (values.size() < 2) {
      values.push_back(value);
      return;
    }
    if (value < values[0]) {
      AddToChild(0, value);
      return;
    }
    if (value < values[1]) {
      AddToChild(1, value);
      return;
    }
    AddToChild(2, value);
  }
};
```

**Aufgabe 5: Algorithmen (Entwurf)** **(10 Punkte)**

Gegeben eine Liste von Zahlen, die ausschließlich die Werte $0, 1$ und $2$ enthalten, entwerfen Sie einen Algoritzmus, der diese Liste aufsteigend sortiert.

**Anmerkung:** Ihre Lösung wird sowohl anhand der Korrektheit als auch der Laufzeitkomplexität bewertet. Eine einfache Lösung, bei der einer der aus der Vorlesung bekannten Sortieralgorithmen angewendet wird, oder bei der auf einen Bibliotheksalgorithmus (z.B. `std::sort`) zurückgegriffen wird, wird nicht akzeptiert.

**Hinweis:** Sie müssen keinen Code für den Algorithmus angeben, eine Erklärung der Idee genügt. Eine optimale Lösung sollte in $\mathcal{O}(n)$ Zeit laufen. Genauer ist es sogar möglich, die Liste mit einem einzigen Durchlauf zu sortieren.