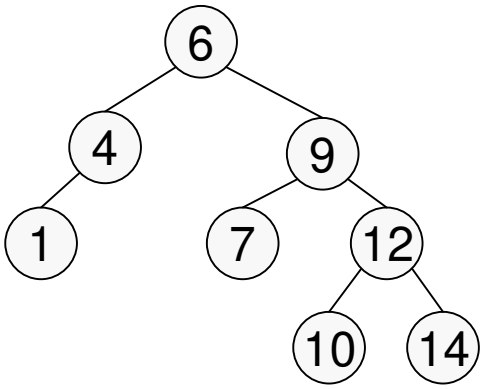
— Übungsblatt 7 (T) —

Aufgabe KV 1 (Suchbäume)

a) Gegeben sei der folgende binäre Suchbaum:



Geben Sie die Reihenfolge der Knoten-Werte an, wenn Sie den Baum in Preorder bzw. in Postorder traversieren.

Reihenfolge in Preorder:

Reihenfolge in Postorder:

b) Bauen Sie einen binären Suchbaum für die Folge

9, 5, 1, 17, 13, 8, 2, 6

auf, indem Sie die Werte der Reihe nach in den Baum einfügen.

Aufgabe KV 2 (Collections und Iteratoren)

In einem Java-Programm arbeiten Sie mit `Collection`-Objekten, von denen Sie wissen, dass alle darin gesammelten Objekte vom Typ `Double` sind. Sie benötigen nun eine Methode, die Ihnen den Mittelwert der in Ihrer Collection gespeicherten Objekte berechnet und zurückliefert.

Vervollständigen Sie eine solche Methode `mittelwert` in der nachfolgenden Klasse `Methoden`. Setzen Sie dabei

- die Methoden `size` und `iterator` des `Collection`-Objekts `c`,
- die Methoden `next` und `hasNext` des Iterators von `c` sowie
- die vorgegebene Methode `doubleValue` für die Wandlung der in `c` gesammelten Objekte nach `double`

ein.

```
import java.util.*;
public class Methoden {

  public static double doubleValue(Object o) {
    return ((Double) o).doubleValue();
  }


  public static double mittelwert (Collection c) {









  }
}
```

Aufgabe KV 3 (Sortieren)

a) Gegeben sei die Buchstaben-Folge

E F A C H B G D

Sortieren Sie diese Folge alphabetisch mit den Verfahren *Insertionsort* (Sortieren durch Einfügen) und *Selectionsort* (Sortieren durch Auswahl).

Geben Sie nach jeder Tauschaktion jeweils die Gesamtfolge an, die sich danach ergibt.

b) Gegeben sei die Buchstaben-Folge

H G F D E C B A

Sortieren Sie diese Folge alphabetisch mit dem Verfahren *Quicksort* (Sortieren durch rekursives Aufteilen) und notieren Sie nur den ersten Zerlegungsschritt für die komplette Folge bei Verwendung von `D` als Pivotelement.

Geben Sie nach jeder Tauschaktion jeweils die Gesamtfolge an, die sich danach ergibt.

H	G	F	D	E	C	B	A

Aufgabe KV 4 (Warteschlange)

Gegeben seien die Klasse

```
public class Wert {
  public double d;   // eingetragener double-Wert
  public Wert next;  // Verweis auf Nachfolger in einer Liste
}
```

und das Interface

```
public interface WarteSchlange {
  public void rein(Wert d); // Methode für die Aufnahme des
                            // Werts d in die Warteschlange
  public Wert raus();       // Methode für das Entfernen und die Rückgabe
                            // des ersten Werts der Warteschlange
}
```

Ergänzen Sie die nachfolgende Klasse `MeineSchlange`, die das Interface `WarteSchlange` implementiert. Realisieren Sie diese Datenstruktur nach dem Prinzip „First In, First Out“. Die Methode `rein` soll also neue Objekte vom Typ `Wert` immer am Ende der Warteschlange einfügen und die Methode `raus` soll jeweils das `Wert`-Objekt am Anfang der Warteschlange zurückliefern und aus der Warteschlange entfernen.

```
public class MeineSchlange implements WarteSchlange {
  private Wert anfang, ende;  // Verweise auf Anfang und Ende der Warteschlange

  public void rein(Wert p) {









  }
  public Wert raus() {









  }
}
```