Beantworten Sie die Fragen in den Aufgaben 1 und 2 mit einer kurzen, prägnanten Antwort.

Aufgabe 1 (8 Punkte)

1. Nennen Sie ein Verfahren, mit dessen Hilfe nachgewiesen werden kann, dass sich ein Programm für *alle* Eingaben korrekt verhält.

2. Aus welchen Komponenten besteht ein endlicher Automat?

3. Nach welchem Prinzip arbeitet der Quicksort-Algorithmus?

4. Welches ist die bestmögliche asymptotische Laufzeit zur Bestimmung des Medians von n Zahlen?

5. Welche Datenstruktur sollte verwendet werden, wenn ein Algorithmus häufig das kleinste Element aus einer Menge entnehmen muss?

6. Welcher ADT wird üblicherweise bei einer Breitensuche auf Graphen eingesetzt?

7. Welche Eigenschaft muss ein gerichteter Graph besitzen, damit er topologisch sortiert werden kann?

8. Welche asymptotische Laufzeit benötigt die Suche nach einem beliebigen Element in einem AVL-Baum, der n Elemente enthält?

Aufgabe 2 (8 Punkte)

Alle Fragen beziehen sich auf die Programmiersprache Java ab Version 5.

1. Mit welchem Aufruf testet man zwei Strings `s` und `t` auf inhaltliche Gleichheit?

2. Woraus besteht die Signatur einer Methode?

3. Mit welchem Schlüsselwort kennzeichnet man Variablen, denen man nur einmal einen Wert zuweisen darf?

4. Was bedeutet der Aufruf `super();` innerhalb eines Konstruktors?

5. Welchen Wert hat der Ausdruck `(a || b) && (!a ^ b)` für `a = true` und `b = false`?

6. Mit welchem Ausdruck erzeugt man ein zweidimensionales `int`-Array mit der Größe 4×4?

7. Mit welcher Anweisung weist man einem `int i` den Wert der Variablen `double d` zu?

8. Von wie vielen Klassen kann eine Klasse erben?

Tragen Sie bei den Aufgaben 3 bis 8 Ihre Lösungen in den vorgesehenen Platz ein.

Aufgabe 3 (7 Punkte)

a) Tragen Sie die 32-Bit-Gleitkommadarstellung nach Definition des Skriptes für die Zahl `-23` in die vorgesehenen Kästchen ein (3 Punkte).

Vorzeichen	Exponent	reduzierte Mantisse

b) Stellen Sie die Dezimalzahlen `11` und `14` als Dualzahlen im 5-Bit-Zweierkomplement dar und berechnen Sie in dieser Darstellung das Ergebnis der Subtraktion `11 - 14`. Überprüfen Sie Ihre Rechnung mit Hilfe des Verfahrens aus der Vorlesung auf Korrektheit und wandeln Sie ggf. das Ergebnis in die dezimale Darstellung um. Schreiben Sie zu Ihren Rechenschritten **stichwortartig** auf, was Sie tun. (4 Punkte)

Aufgabe 4 (7 Punkte)

Betrachten Sie die Java-Klasse `Sichtbar` und geben Sie die Werte der Variablen `a`, `b` und `c` zu den Zeitpunkten `1 - 7` an. Falls eine oder mehrere der Variablen zu einem Zeitpunkt Ihrer Meinung nach nicht definiert sind, so setzen Sie an der entsprechenden Stelle ein Minus (`-`) in die Tabelle.

```
public class Sichtbar {

    public static int   a = 30;
    public static int[] b = { 14, 3 };

    public static void main(String[] args) {
      /* Zeitpunkt 1 */
      int a = methode1(b);
      /* Zeitpunkt 5 */
      a = methode2();
      /* Zeitpunkt 7 */
    }

    public static int methode1(int[] c) {
      /* Zeitpunkt 2 */
      c[0] = c[0] * c[1];
      /* Zeitpunkt 3 */
      c = new int[1];
      /* Zeitpunkt 4 */
      return c[0];
    }

    public static int methode2() {
      /* Zeitpunkt 6 */
      return 3 * a;
    }
}
```

Zeitpunkt	1	2	3	4	5	6	7
`int a`							
`int[] b`							
`int[] c`							

Aufgabe 5 (6 Punkte)

Gegeben sei folgende Java-Klasse:

```
public class Fraglich {

  public static int methode(int n) {

    if (n == 1) {
      return 1;
    }

    int a = methode(n / 2);

    a *= 4;

    if (n % 2 == 1) {
      a += 2 * n - 1;
    }

    return a;
  }
}
```

a) Was berechnet die Methode `methode` der Klasse `Fraglich` bei Eingabe einer ganzen Zahl $n > 0$? (4 Punkte)

b) Wie ist die Laufzeit der Methode `methode` in der O-Notation? (2 Punkte)

Aufgabe 6 (6 Punkte)

Sortieren Sie die Zahlenfolge

8 5 4 4 2 9 6

nach der Methode des Quicksort. Stellen Sie die Arbeitsweise des Algorithmus dar, indem Sie jeweils das Ergebnis eines rekursiven Durchlaufs in eine neue Zeile schreiben. Kreisen Sie das Pivotelement ein und markieren Sie die betrachtete Teilfolge. Notieren Sie außerdem neben der Zeile, welche Elemente in der Zeile jeweils miteinander getauscht wurden.

8	5	4	4	2	9	6	getauschte Elemente:

Aufgabe 7 (4 Punkte)

Nennen Sie für die Sortierverfahren in der Tabelle jeweils die Komplexitätsklasse in der O-Notation. Legen Sie dabei jeweils die beste Implementation des Sortieralgorithmus zugrunde.

	best case	average case	worst case
HeapSort			
SelectionSort			
MergeSort			
QuickSort			

Aufgabe 8 (6 Punkte)

Betrachten Sie die Hashfunktion $h(x) = (2 \cdot x + 5) \bmod 11$
Beispiele:

$$h(4) = (2 \cdot 4 + 5) \bmod 11 = 13 \bmod 11 = \mathbf{2}$$
$$h(9) = (2 \cdot 9 + 5) \bmod 11 = 23 \bmod 11 = \mathbf{1}$$

Außerdem ist folgende Hashtabelle gegeben:

$h(x)$	Element x
0	
1	7
2	51
3	
4	16
5	5
6	40
7	
8	71
9	18
10	

a) Geben Sie die Art des Hashings an, mit der diese Hashtabelle erzeugt wurde. (1 Punkt)

b) Geben Sie die Reihenfolge an, in der die Elemente eingefügt wurden. (3 Punkte)

c) Geben Sie den Hashwert für die Zahl 42 an und fügen Sie sie in die Tabelle oben ein. (2 Punkte)

Bei den Aufgaben 9 und 10 müssen Sie selber etwas zeichnen. Nutzen Sie den dafür vorgesehenen Platz.

Aufgabe 9 (4 Punkte)

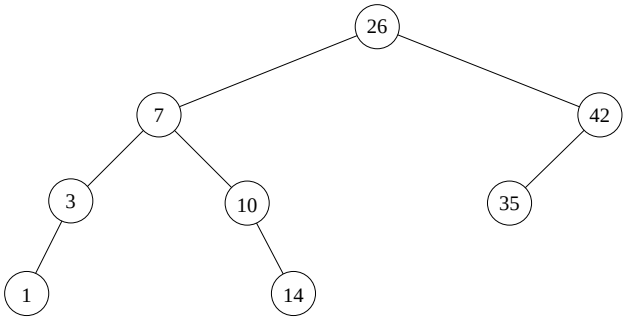
Gegeben sei die folgende Postorder-Traversierung eines binären Suchbaums:

```
2  1  3  5  6  9  8  7  4
```

Zeichnen Sie den zugehörigen binären Suchbaum.

Aufgabe 10 (7 Punkte)

Gegeben sei folgender AVL-Baum:



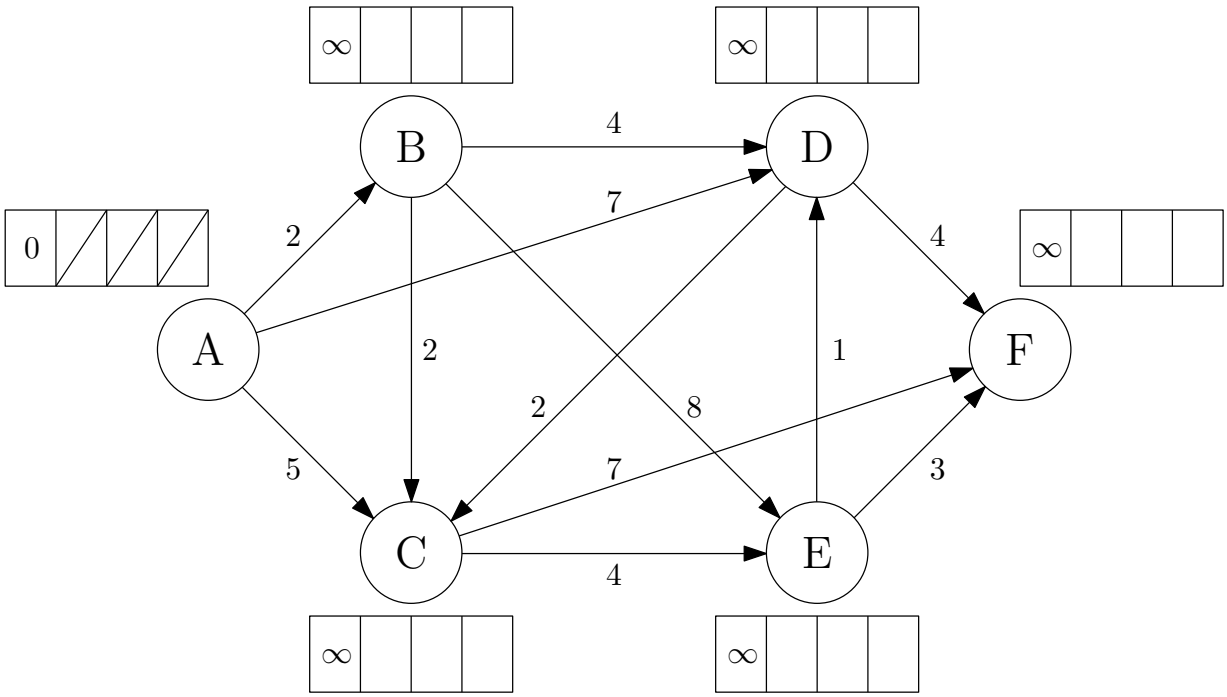
a) Fügen Sie in den Baum den Schlüssel `17` ein, indem Sie den Knoten direkt in den obigen AVL-Baum einzeichnen. Versehen Sie den kompletten Baum mit Balancen. (1 Punkt)
Führen Sie eine eventuell notwendige Rotation aus und zeichnen Sie den reorganisierten Baum erneut. Vermerken Sie, welche Rotation Sie verwenden haben. (2 Punkte)

b) Fügen Sie in den neu gezeichneten Baum in Aufgabenteil a) den Schlüssel `23` ein und versehen Sie den kompletten Baum mit Balancen. (1 Punkt)
Führen Sie die notwendige Rotation aus und zeichnen Sie den reorganisierten Baum unten nochmals. Vermerken Sie die Art der Rotation, die Sie verwendet haben. (3 Punkte)

Tragen Sie bei den Aufgaben 11 und 12 Ihre Lösungen in den vorgesehenen Platz ein.

Aufgabe 11 (8 Punkte)

Gegeben sei der folgende gerichtete, gewichtete Graph $G = (V, E)$:



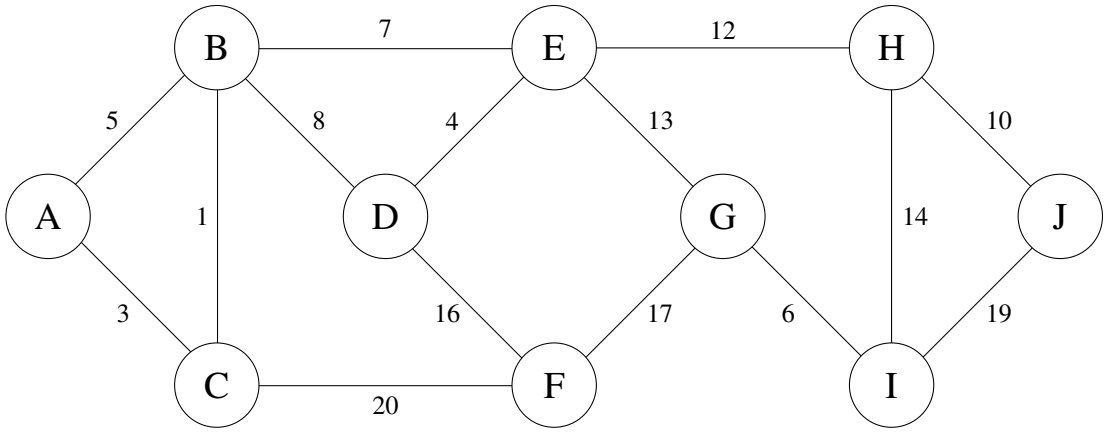
a) Stellen Sie G mit Hilfe einer *Adjazenz-Matrix* dar. Benutzen Sie die folgende Vorlage (2 Punkte):

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

b) Berechnen Sie mit dem *Algorithmus von Dijkstra* kürzeste Wege von Knoten A zu allen anderen Knoten in G. Jedesmal, wenn im Verlauf des Algorithmus der Distanzwert für einen Knoten aktualisiert wird, schreiben Sie den neuen Wert in das nächste freie Kästchen neben den Knoten in der obigen Abbildung. Markieren Sie abschließend diejenigen Kanten in der Abbildung, die zu den gefundenen kürzesten Wegen gehören. (6 Punkte)

Aufgabe 12 (6 Punkte)

Gegeben sei der folgende ungerichtete, gewichtete Graph $G = (V, E)$:



Berechnen Sie mit Hilfe des *Algorithmus von Kruskal* einen minimalen Spannbaum (MSB) von G. Veranschaulichen Sie den Ablauf des Algorithmus in der folgenden Tabelle durch eine chronologische Auflistung der untersuchten Kanten, jeweils mit der Angabe, ob die entsprechende Kante in den MSB aufgenommen wird (ja/nein). Tragen Sie nur im Falle der Ablehnung einer Kante eine kurze Begründung dafür in die rechte Spalte ein. Markieren Sie abschließend in der obigen Abbildung alle Kanten, die zum gefundenen MSB gehören.

Kante	Aufnahme in MSB?	Begründung bei Ablehnung

Bei den Aufgaben 13 bis 16 müssen Sie selbst einige Zeilen Java-Code schreiben. Nutzen Sie den vorgesehenen Platz zwischen den geschweiften Klammern.

Aufgabe 13 (4 Punkte)

Ergänzen Sie in der gegebenen Klasse `BaumTools` die Methode `anzahlKnoten`, die die Anzahl der Knoten des übergebenen Baumes zurückgibt.

```
public class BaumTools {
  public static int anzahlKnoten(Baum b) {









  }
}
```

Aufgabe 14 (9 Punkte)

Implementieren Sie analog zur *VerweisListe* den ADT *Schlange* mit Hilfe von Verweisen. Ergänzen Sie dazu die leeren Methoden in der Klasse `VerweisSchlange`.
Eine `VerweisSchlange` besitzt einen Verweis `head`, der auf das vorderste Element der Schlange verweist, und einen Verweis `tail`, der auf das letzte Element verweist. Die Elemente der Schlange sind Instanzen der Klasse `Eintrag`, die jeweils ein `Object` enthalten und auf den nachfolgenden Eintrag verweisen.

```
public class Eintrag {

  Object inhalt;
  Eintrag next;
}
```

Achten Sie auf eine geeignete Fehlerbehandlung.

```
public class VerweisSchlange implements Schlange {

  private Eintrag head;
  private Eintrag tail;

  public boolean empty() {









  }

  public Object front() {









  }
```

Fortsetzung nächste Seite

```
  public void enq(Object x) {

  }

  public void deq() {

  }
}
```

Aufgabe 15 (5 Punkte)

Das Regula-falsi-Verfahren ist ein Iterationsverfahren zur Nullstellenbestimmung von Funktionen. Es startet mit zwei Stellen a_0 und b_0, deren Funktionswerte $f(a_0)$ und $f(b_0)$ unterschiedliches Vorzeichen haben. Nun verkleinert man in mehreren Iterationsschritten das Intervall $[a_k, b_k]$ und bekommt so eine immer genauere Näherung für die Nullstelle. Im k-ten Schritt berechnet man:

$$c_k = \frac{a_k \cdot f(b_k) - b_k \cdot f(a_k)}{f(b_k) - f(a_k)}$$

Das c_k ersetzt dann entweder die linke Intervallgrenze a_k oder die rechte b_k wie folgt:

$a_{k+1} = c_k$ und $b_{k+1} = b_k$, falls $f(a_k)$ und $f(c_k)$ das gleiche Vorzeichen haben
$a_{k+1} = a_k$ und $b_{k+1} = c_k$, sonst

Implementieren Sie dieses Iterationsverfahren **iterativ**, um jeweils die Nullstelle der Funktion $f(x) = x^3 - 2$ zu berechnen. Ergänzen Sie dazu die Methode `nullstelle` in der Klasse `RegulaFalsiIterativ`. Die Methoden werden jeweils mit den aktuellen Intervallgrenzen und einer Schranke epsilon aufgerufen. Sie können dabei annehmen, dass die übergebenen Parameter korrekt sind, also `a` $<$ `b`, `epsilon` > 0 und die Nullstelle im Intervall $[a, b]$ liegt. Gilt für die Grenzen `b - a < epsilon`, soll eine der beiden Grenzen als Nullstelle zurückliefert werden. (Vgl. Aufgabe 16.)

```
public class RegulaFalsiIterativ {
  public static double f(double x) {
    return x * x * x - 2;
 }
 public static double nullstelle(double a, double b, double epsilon) {









  }
}
```

Aufgabe 16 (5 Punkte)

Betrachten Sie erneut das Regula-falsi-Verfahren zur Nullstellenbestimmung von Funktionen aus Aufgabe 15. Es startet mit zwei Stellen a_0 und b_0, deren Funktionswerte $f(a_0)$ und $f(b_0)$ unterschiedliches Vorzeichen haben. Nun verkleinert man in mehreren Iterationsschritten das Intervall $[a_k, b_k]$ und bekommt so eine immer genauere Näherung für die Nullstelle. Im k-ten Schritt berechnet man:

$$c_k = \frac{a_k \cdot f(b_k) - b_k \cdot f(a_k)}{f(b_k) - f(a_k)}$$

Das c_k ersetzt dann entweder die linke Intervallgrenze a_k oder die rechte b_k wie folgt:

$a_{k+1} = c_k$ und $b_{k+1} = b_k$, falls $f(a_k)$ und $f(c_k)$ das gleiche Vorzeichen haben
$a_{k+1} = a_k$ und $b_{k+1} = c_k$, sonst

Implementieren Sie dieses Iterationsverfahren **rekursiv**, um die Nullstelle der Funktion $f(x) = x^3 - 2$ zu berechnen. Ergänzen Sie dazu die Methode `nullstelle` in der Klasse `RegulaFalsiRekursiv`. Die Methoden werden jeweils mit den aktuellen Intervallgrenzen und einer Schranke epsilon aufgerufen. Sie können dabei annehmen, dass die übergebenen Parameter korrekt sind, also `a` $<$ `b`, `epsilon` > 0 und die Nullstelle im Intervall $[a, b]$ liegt. Gilt für die Grenzen `b - a < epsilon`, soll eine der beiden Grenzen als Nullstelle zurückgeliefert werden.

```
public class RegulaFalsiRekursiv {
  public static double f(double x) {
    return x * x * x - 2;
 }
 public static double nullstelle(double a, double b, double epsilon) {









  }
}
```