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

Aufgabe 1 (8 Punkte)

1. Wie wird bei der Zusicherungsmethode die Zusicherung genannt, die vor Eintritt und vor Austritt bei Schleifen gilt?

2. Mit welcher Traversierung durch einen Suchbaum erhält man eine aufsteigend sortierte Folge?

3. Wieviele Bits werden für die Codierung einer `double`-Zahl verwendet?

4. Nach welchem Prinzip arbeitet ein Keller?

5. Wie ist die Laufzeit (O-Notation) von QuickSort im *Worst Case* abhängig von der Array-Länge `n`?

6. Wie sieht `-2` in der 4Bit 2-er Komplementdarstellung aus?

7. Wie nennt man minimal ausgeglichene AVL-Bäume?

8. Welche zwei Typen von Knoten besitzt ein Spielbaum?

Aufgabe 2 (8 Punkte)

Alle Fragen beziehen sich auf die Programmiersprache Java 5.

1. Welche Methode muss eine ausführbare Klasse immer besitzen?

2. Wie nennt man eine Methode, wenn sie in einer abgeleiteten Klasse die gleiche Signatur hat wie in der Oberklasse?

3. Was verwendet man, wenn man mit Hilfe der Klasse `Collections` eine Liste sortieren will, die jedoch keine `Comparable`-Objekte enthält?

4. Mit welchem Schlüsselwort kennzeichnet man in Java Klassenmethoden?

5. Wie lautet der Ausdruck, um zu überprüfen, ob eine Referenz `f` auf ein Objekt vom Typ `String` verweist?

6. Was bedeutet der Rückgabewert `void` bei Methoden in Java?

7. Gegeben ist folgender Ausdruck: `int a = 1 == 2 ? 3 : 4;` Welchen Wert hat a?

8. Die Klasse `MyException` erbt von der Klasse `Exception`. Wie lautet der Code innerhalb eines Methodenkopfes, mit dem Sie deklarieren, dass eine Methode eine `MyException` werfen könnte?

Beantworten Sie die Aufgabe 3, indem Sie unten genau ein Kreuz machen. Ist Ihre Antwort richtig, erhalten Sie alle Punkte. Ist sie falsch, erhalten Sie keinen Punkt.

Aufgabe 3 (5 Punkte)

Gegeben sei folgende Java-Klasse:

```
import AlgoTools.IO;

public class Raum{

  public Raum vor;
  public Raum hinter;
  public Raum neben;
  public String name;

  public void baueRaum(Raum vor, Raum hinter,
                       Raum neben, String name){
     this.vor = vor;
     this.hinter = hinter;
     this.neben = neben;
     this.name = name;
  }

  public static void main(String[] args){

     Raum kueche = new Raum();
     Raum bad = new Raum();
     Raum wohnzimmer = new Raum();
     Raum schlafzimmer = new Raum();

     kueche.baueRaum(bad, wohnzimmer, schlafzimmer, "Kueche");
     bad.baueRaum(kueche, wohnzimmer, schlafzimmer, "Bad");
     wohnzimmer.baueRaum(kueche, schlafzimmer, bad, "Wohnzimmer");
     schlafzimmer.baueRaum(bad, wohnzimmer, kueche, "Schlafzimmer");

     bad.hinter = kueche.neben;
     bad.name = schlafzimmer.hinter.name;
     wohnzimmer.vor = schlafzimmer.hinter;

     IO.println(bad.hinter.hinter.vor.neben.name);
   }
}
```

Welche Ausgabe hat die Methode `main` dieser Klasse?

◯ `Kueche` ◯ `Bad` ◯ `Wohnzimmer` ◯ `Schlafzimmer`

Tragen Sie bei den Aufgaben 4 bis 10 Ihre Lösungen in den vorgesehenen Platz ein.

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 = 42;
    public static int[] b ={7};

    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]= 5;
      /* Zeitpunkt 3 */
      c = new int[1];
      /* Zeitpunkt 4 */
      return c[0];
    }

    public static int methode2(){
      /* Zeitpunkt 6 */
      return 2+a;
    }
}
```

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

Aufgabe 5 (5 Punkte)

a) Tragen Sie die 32-Bit-Codierung nach Definition des Skriptes für die Zahl `-6.5` in die vorgesehen Kästchen ein (3 Punkte).

Vorzeichen	Exponent	reduzierte Mantisse

b) Welche Zahl wird durch folgende Bitfolge codiert (2 Punkte)?

```
0 00000011 01000000000000000000000
```

Aufgabe 6 (6 Punkte)

Gegeben sei folgende Java-Klasse:

```
import AlgoTools.IO;

public class Raetsel {

public static int frage(int n) {
    if (n < 0)
        throw new IllegalArgumentException("n < 0");
    int x = 1, y = 1;
    for (int i = 1; i < n; i++) {
      y = y + x;
      x = y - x;
    }
    return y;
}
```

a) Was berechnet die Methode `public static int frage(int n)` der Klasse `Raetsel` für $n \geq 0$? (4 Punkte)

b) Wie ist die Laufzeit der Methode `public static int frage(int n)` in der O-Notation? (2 Punkte)

Aufgabe 7 (6 Punkte)

Gegeben sei folgende Java-Klasse:

```
public class Fraglich {

 public static int methode(int a, int b, int c) {
    int wert = 0;
    for (int i = a; i > 0; i--) {
      for (int j = 0; j < b; j++) {
        wert += c;
      }
    }
    return wert;
  }
}
```

a) Was berechnet die Methode `public static int methode(int a, int b, int c)` der Klasse `Fraglich` für ganze Zahlen $a, b, c \geq 0$? (4 Punkte)

b) Wie ist die Laufzeit der Methode `public static int methode(int a,int b,int c)` in der O-Notation, wenn $a = b$? (2 Punkte)

Aufgabe 8 (4 Punkte)

Sortieren Sie die Zahlenfolge

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

nach der Methode des **iterativen** MergeSort. Stellen Sie die Arbeitsweise des Algorithmus dar, indem Sie die Gesamtfolge immer dann in eine neue Zeile schreiben, wenn alle Teilfolgen einer Länge fertig sortiert sind. Verwenden Sie senkrechte Striche (|), um die Grenzen der Teilfolgen zu markieren.

Aufgabe 9 (6 Punkte)

Gegeben sei folgendes char-Array:

```
O L I V E R
```

Sortieren Sie das char-Array mit dem Heapsort-Verfahren aus der Vorlesung lexikografisch.

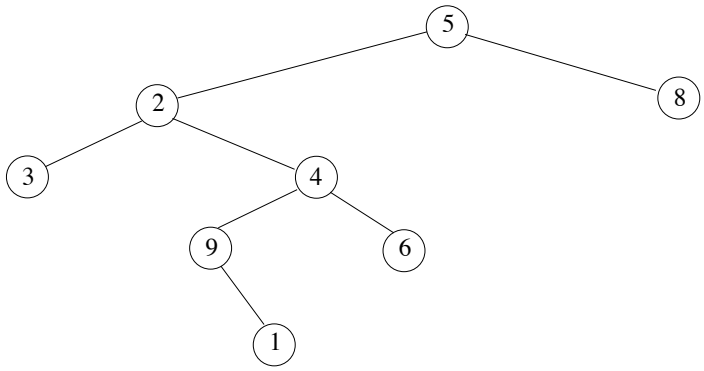
a) Geben Sie für die erste Phase den Baum sowie den daraus erzeugten Heap an.
(2 Punkte)

`Ursprungsbaum:` `Heap:`

b) Für Phase 2 geben Sie jeweils den Baum nach Entfernen der Wurzel, den reorganisierten Heap, sowie das zugehörige aktuelle char-Array für jeden Sortierschritt nach dem Reorganisieren an. (4 Punkte)

Aufgabe 10 (5 Punkte)

Gegeben sei folgender Baum:



Tragen Sie die Traversierungen dieses Baumes in folgender Tabelle ein:

Preorder								
Inorder								
Postorder								
Tiefensuche								
Breitensuche								

Aufgabe 11 (6 Punkte)

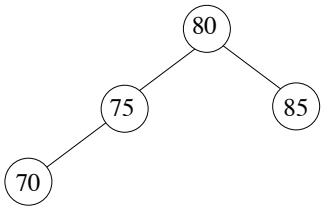
Gegeben sei die Postorder-Traversierung eines Suchbaums:

```
14  9  18  17  19  42  25  23  16
```

Zeichnen Sie den zugehörigen Baum.

Aufgabe 12 (6 Punkte)

Gegeben sei folgender AVL-Baum:



a) Fügen Sie in den Baum die Zahl `60` ein, indem Sie den Knoten direkt in den obigen AVL-Baum einzeichnen. Versehen Sie den kompletten Baum mit Balancen. (1 Punkt)

b) Führen Sie eine eventuell notwendige Rotation aus und zeichnen Sie den reorganisierten Baum erneut. Vermerken Sie, welche Rotation Sie verwenden haben. (2 Punkte)

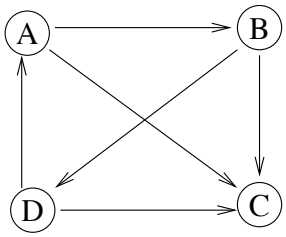
c) Fügen Sie in den neu gezeichneten Baum in Aufgabenteil b) die Zahl `73` ein und versehen Sie den kompletten Baum mit Balancen (1 Punkt).
d) Führen Sie die notwendige Rotation aus und zeichnen sie den reorganisierten Baum unten nochmals. Vermerken Sie die Art der Rotation, die Sie verwendet haben (2 Punkte).

Tragen Sie bei Aufgabe 13 Ihre Lösungen in den vorgesehenen Platz ein.

Aufgabe 13 (6 Punkte)

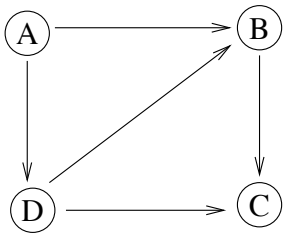
Geben Sie jeweils an, ob der Graph topologisch sortiert werden kann. Ist der Graph topologisch sortierbar, geben Sie die zugehörige Sortierung an. Wenn der Graph nicht topologisch sortierbar ist, begründen Sie, warum.

a)



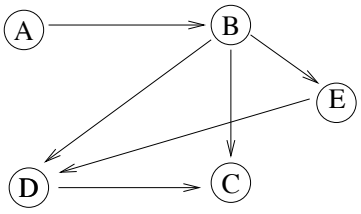
topologisch sortierbar?	
Sortierung oder Begründung	

b)



topologisch sortierbar?	
Sortierung oder Begründung	

c)



topologisch sortierbar?	
Sortierung oder Begründung	

Bei den Aufgaben 14 bis 16 müssen Sie selbst einige Zeilen Java-Code schreiben. Nutzen Sie den vorgesehenen Platz zwischen den geschweiften Klammern.

Aufgabe 14 (6 Punkte)

Ergänzen Sie in der gegebenen Klasse `BaumTools` die Methode `hoehe`, die die Anzahl der Ebenen des Baumes zurückgibt. Hinweis: Verwenden Sie Rekursion!

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









  }
}
```

Aufgabe 15 (8 Punkte)

Ein Spielzeughersteller verkauft Flummis mit besonderer Sprungkraft. Wenn sie aus einer bestimmten Höhe h (in Metern) fallen gelassen werden, erreichen sie nach jedem Aufkommen 90 Prozent der Höhe des letzten Sprunges.

Ergänzen Sie die rekursive Methode `huepferRek(double h)` so, dass sie berechnet und zurückgibt, wie oft der Flummi hüpft, bevor er nur noch weniger als einen Millimeter vom Boden abhebt. Implementieren Sie außerdem die Methode `huepferIter(double h)`, die das gleiche berechnet, jedoch nicht rekursiv, sondern iterativ!

Achten Sie auf eine geeignete Fehlerbehandlung und verwenden Sie als Fehlerklasse die Klasse `RuntimeException`!
Hinweis: Ein Millimeter entspricht einem tausendstel Meter.

```
public class Flummi {

  public static int huepferRek(double h) {









  }

  public static int huepferIter(double h) {









  }
}
```

Aufgabe 16 (7 Punkte)

Schreiben Sie Klasse `KreisListe`, die von `VerweisListe` erbt. Eine `KreisListe` hat die Eigenschaft, dass beim Durchlaufen der Liste niemals an der Endposition stehen geblieben wird. Statt dessen wird in dem Moment, wo die Endposition erreicht wird, wieder an den Anfang der Liste gesprungen.
Implementieren Sie die Klasse `KreisListe` mit der oben beschriebenen Eigenschaft. Dazu müssen sie eine der Listen-Methoden überschreiben. Ergänzen Sie den unten stehenden Code.

```
public class KreisListe                                    {

}
```