Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Klausur
Algorithmen und Datenstrukturen
SS 2020 – 23.09.2020
Studiengänge Informatik und SST/SWT dual

Hinweise:

- Die Bearbeitungszeit beträgt 60 Minuten.
- Zum Bestehen der Klausur sind 50 Punkte erforderlich.
- Schreiben Sie auf die ersten beiden Blätter Ihren Namen, Matrikelnummer, Studiengang und Sitzplatznummer.
- Erlaubte Hilfsmittel: keine
- Lösen Sie nicht die Klammerung der Klausur!
- Tragen Sie Ihre Lösungsvorschläge in die Klausurvorlage ein. Ein Zusatzblatt befindet sich am Ende der Klausur.
- Alle vorgegebenen und zu erstellenden Programmtexte beziehen sich auf die Programmiersprache Java.
- Bitte schreiben Sie deutlich.

Viel Erfolg !

Aufgabe	Maximalpunkte	Erreichte Punkte
1 (Multiple Choice)	20	
2 (Graphen)	17	
3 (Generics)	8	
4 (Listen)	16	
5 (Binäre Suchbäume)	30	
6 (Sortieren)	14	
Summe	105	

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 1 (Multiple Choice) *20 Punkte*

Kreuzen Sie in den folgenden Teilaufgaben an, welche Lösungen richtig und welche falsch sind. Für jede korrekt markierte Aussage erhalten Sie 1 Punkt, für jede falsch markierte wird 1 Punkt abgezogen. Pro Teilaufgabe erhalten Sie mindestens 0 Punkte.

Bewerten Sie folgende Aussagen zu Suchbäumen	**Richtig**	**Falsch**
Jeder Binärbaum mit 2 Knoten ist ein Suchbaum.		
Ein AVL-Baum hat niemals eine größere Tiefe als ein Suchbaum, der die gleichen Schlüssel enthält.		
Das Suchen in AVL-Bäumen mit n Knoten erfolgt im schlechtesten Fall mit einem Aufwand von $O(\log_2 n)$.		
Durchläuft man einen AVL-Baum in Preorder-Reihenfolge, so werden die Schlüssel nach ihrer Größe sortiert durchlaufen.		

Welche der folgenden Aussagen sind korrekt?	**Richtig**	**Falsch**
Die Rekursionstiefe ist die max. Anzahl aller rekursiven Aufrufe einer Methode.		
Die Rekursionstiefe ist die max. Anzahl aller **gleichzeitig im Speicher** vorhandenen rekursiven Aufrufe einer Methode.		
Die Rekursionsbasis beschreibt ein einfaches Grundproblem, das sofort gelöst werden kann.		
Rekursive Methoden besitzen die Gefahr eines Stack-Überlaufs.		

Welche der folgenden Suchverfahren sind elementar?	**Richtig**	**Falsch**
Binäre Suche		
Interpolationssuche		
Lineares Suchen		
Hashtabelle mit Quadratischem Sondieren		

Welche der folgenden Aussagen sind korrekt?	**Richtig**	**Falsch**
Rekursive Methoden, die jeweils zwei neue Aufrufe erzeugen, haben eine Laufzeit von $O(2^n)$.		
Rekursive Methoden mit Endrekursionen lassen sich leicht iterativ formulieren.		
Bestimmte Probleme lassen sich ausschließlich rekursiv formulieren.		
Bei rekursiven Methoden werden die Parameter auf dem Heap abgelegt.		

Welche der folgenden Aussagen sind korrekt?	**Richtig**	**Falsch**
Schnelle Sortierverfahren haben im schlechtesten Fall eine Zeitkomplexität von $O(n \log_2 n)$.		
Die Anzahl der benötigten Vergleiche bei Heapsort ist durch die Anzahl der Schlüssel, nicht aber durch ihre ursprüngliche Reihenfolge bestimmt.		
Schnelle Sortierverfahren sind immer schneller als elementare Sortierverfahren.		
Quicksort hat im Mittel eine Zeitkomplexität von $O(n \log_2 n)$.		

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 2 (Graphen) *17 Punkte*

Folgender Graph ist durch seine Adjazenzmatrix gegeben:

	A	**B**	**C**	**D**	**E**
A		3		7	
B			4	9	
C					2
D					6
E		1			

a) Zeichnen Sie den gegebenen Graphen. Nutzen Sie dabei alle Informationen der Matrix.

b) Kann man für obigen Graphen eine topologische Sortierung angeben (Begründung)? Falls eine topologische Sortierung existiert, so geben Sie sie an.

c) Führen Sie für den zugehörigen ungerichteten Graphen eine **Tiefensuche** beginnend ab A durch.

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 3 (Generics) *8 Punkte*

Gegeben ist die Klasse `Fahrzeug`, die ein Attribut `name` hat und mit einem `Kraftstoff` betankt werden kann. Ändern Sie die Klasse `Fahrzeug` so, dass Generics benutzt werden. Der `traktor` soll nur mit `Diesel` befüllt werden können, das `auto` ausschließlich mit `Benzin`, und das `flugzeug` nur mit `JetA1`.

Hinweis: Sie können den Typparameter analog zu `class X<Y extends Z>` eingrenzen!

```
public class Kraftstoff
{
    …
}

public class Diesel extends Kraftstoff
{
    …
}

public class Benzin extends Kraftstoff
{
    …
}

public class JetA1 extends Kraftstoff
{
    …
}

public class Fahrzeug
{
    private String name;
    private Kraftstoff kraftstoff;

    public Fahrzeug(String name)
    {
        this.name = name;
    }

    public void betanken(Kraftstoff kraftstoff)
    {
        this.kraftstoff = kraftstoff;
    }

    public static void main(String[] args)
    {
        Fahrzeug traktor = new Fahrzeug("Traktor");
        traktor.betanken(new Diesel());

        Fahrzeug auto = new Fahrzeug("Auto");
        auto.betanken(new Benzin());

        Fahrzeug flugzeug = new Fahrzeug("Flugzeug");
        flugzeug.betanken(new JetA1());
    }
}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Schreiben Sie die Klasse `Fahrzeug` hier komplett neu (einschließlich der `main`-Methode):

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 4 (Listen) *16 Punkte*

Aufgrund der Corona-Pandemie musste ein Freibad den Zugang begrenzen. Vor dem Freibad gibt es eine `warteschlange` (`Liste` vom Typ `Person`), außerdem wird gespeichert wer sich gerade im `freibad` befindet (ebenfalls eine `Liste` vom Typ `Person`).

Der `eintritt` funktioniert wie folgt:

- Wenn die erste Position der `warteschlange` leer ist, soll gar nichts passieren.
- Andernfalls wird das erste Element aus der `warteschlange` entfernt. Alle nachfolgenden `Person`en rücken auf.
- Die `Person`, die aus der `warteschlange` entfernt wurde, soll am Anfang ins `freibad` eingefügt werden. Alle `Person`en im `freibad` rücken dazu eine Position nach hinten.

Vorher:

`warteschlange`
Fritz
Hakan
Konstantin
Jana
Michael
Tanja

`freibad`
Johannes
Daniel
Vivien
Anne
Britta

Nach dem `eintritt`:

`warteschlange`
Hakan
Konstantin
Jana
Michael
Tanja

`freibad`
Fritz
Johannes
Daniel
Vivien
Anne
Britta

a) Implementieren Sie den Rumpf der Methode `eintritt` in der Klasse `Hygienekonzept` (s.u.) so, dass die beschriebene Funktionalität auf den Parametern `warteschlange` und `freibad` umgesetzt wird. Beachten Sie dazu den gegebenen Quellcode von `Link<T>`, `Liste<T>` und `Person`.

```
public class Link<T>
{
    public T daten;
    public Link naechster;

    public Link(T daten, Link naechster)
    {
        this.daten = daten;
        this.naechster = naechster;
    }
}

public class Liste<T>
{
    public Link<T> anfang;
    public Link<T> ende;

    …
}

public class Person
{
    public String name;
    …
}

public class Hygienekonzept
{
    public static void eintritt(Liste<Person> warteschlange,
        Liste<Person> freibad)
    {









    }
}
```

b) Warum war diese Aufgabe besonders einfach zu programmieren?

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 5 (Binäre Suchbäume) *30 Punkte*

Betrachten Sie die folgende teilweise Implementierung eines Binären Suchbaumes:

```
public class Knoten
{
   public int schluessel;
   public Knoten[] kinder = new Knoten[2];

}

public class BinSuchbaum
{
   private Knoten wurzel;

   public int groessterSchluessel()
   {
      // Aufgabenteil a)
   }

   public int anzahlInnereKnoten()
   {
      return anzahlInnereKnoten(wurzel);
   }

   private boolean istInnererKnoten(Knoten knoten)
   {
      // Knoten mit mindestens einem Kind
      // Aufgabenteil b)
   }

   private int anzahlInnereKnoten(Knoten knoten)
   {
      // Aufgabenteil c)
   }
}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

a) Ergänzen Sie in der Klasse `BinSuchbaum` die Methode `groessterSchluessel`, die iterativ den größten Schlüssel im Binärbaum bestimmt. Sie können davon ausgehen, dass die Wurzel existiert.

```
public int groessterSchluessel()
{
   assert(wurzel != null);









}
```

b) Ergänzen Sie die Klasse `BinSuchBaum` um eine Methode `istInnererKnoten`, die prüft, ob `knoten` ein innerer Knoten ist, also mindestens ein Kind besitzt.

```
private boolean istInnererKnoten(Knoten knoten)
{
   assert(knoten != null);









}
```

c) Ergänzen Sie die Klasse `BinSuchbaum` um eine Methode `anzahlInnereKnoten`, die rekursiv die Anzahl der inneren Knoten im Teilbaum mit der Wurzel `knoten` bestimmt.

Hinweis: die Methode `istInnererKnoten` aus der vorigen Teilaufgabe könnte sich als nützlich erweisen. Denken Sie zunächst eine Minute darüber nach, bevor Sie Ihren Lösungsvorschlag hinschreiben!

```
private int anzahlInnereKnoten(Knoten knoten)
{
   assert(knoten != null);









}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 6 (Sortieren) *14 Punkte*

a) In der folgenden Tabelle ist die Implementierung einer Variante eines bekannten Sortieralgorithmus angegeben. Ergänzen Sie die Tabelle um den Namen des Verfahrens und die asymptotische Anzahl an Vergleichen im schlechtesten Fall (worst case).

Implementierung	Name	Worst case

`public static void sortiere(int[] feld)`
`{`
`  assert(feld != null);`
 	
`  for (int a=0; a<feld.length-1; a++)`
`  {`
`  int maxpos = a;`
 	
`  for (int b=a+1; b<feld.length; b++)`
`  if (feld[b] > feld[maxpos])`
`  maxpos = b;`
 	
`  final int tmp = feld[a];`
`  feld[a] = feld[maxpos];`
`  feld[maxpos] = tmp;`
`  }`
`}`		

b) Wenden Sie die **Implementierung aus Aufgabenteil a)** auf das unten angegebene Feld an. Beachten Sie die Sortierrichtung und geben Sie die Reihenfolge der Schlüssel nach jedem Durchlauf der äußeren Schleife an. Benutzen Sie einen senkrechten Strich **|**, um Teilfelder voneinander abzutrennen. Die Tabelle enthält viel mehr Zeilen als erforderlich sind.

3	9	15	1	4	5	7

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Zusatzblatt