Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Klausur
Algorithmen und Datenstrukturen
SS 2020 – 22.09.2020

Studiengänge Medizinische Informatik und Wirtschaftsinformatik

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 Ihre Sitzplatznummer ein.
- 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, Wissen)	20	
2 (Komplexität, Rekursion)	12	
3 (Listen)	16	
4 (Binäre Suchbäume)	30	
5 (Sortieren)	10	
6 (Graphen)	17	
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.

Welche der folgenden Aussagen sind korrekt?	**Richtig**	**Falsch**
$n = O(\log n)$		
$\sqrt{n} = O(n)$		
$n = O(n \log n)$		
$2^n = O(n^3)$		

Bewerten Sie folgende Aussagen	**Richtig**	**Falsch**
Die Rekursionstiefe ist die maximale Anzahl aller rekursiven Aufrufe einer Methode.		
Die Rekursionstiefe ist die maximale 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 Überlaufs des Aufrufstapels (engl. stack overflow).		

Bewerten Sie folgende Aussagen. In einem B-Baum der Ordnung m …	**Richtig**	**Falsch**
… sind alle Schlüssel eines Knotens größer als die Schlüssel der Kinder.		
… enthält **jeder** Knoten mindestens m Schlüssel.		
… haben alle Blätter die gleiche Tiefe.		
… enthält jeder Knoten höchsten 2m Schlüssel.		

Bewerten Sie folgende Aussagen	**Richtig**	**Falsch**
Für die Interpolationssuche benötigen wir numerische Schlüssel, also Schlüssel die Zahlen sind.		
Die Interpolationssuche in n Elementen besitzt im schlechtesten Fall (worst-case) eine Zeitkomplexität von O(log n).		
Es seien n Schlüssel in einem AVL-Baum gespeichert. Die Suche nach einem Schlüssel benötigt dann höchstens O(log n) Vergleiche.		
Für die lineare Suche müssen die Schlüssel geordnet sein.		

Bewerten Sie folgende Aussagen.	**Richtig**	**Falsch**
Heapsort ist ein elementares Sortierverfahren.		
Insertionsort ist ein elementares Sortierverfahren.		
Quicksort ist ein Divide-and-Conquer Verfahren.		
Selectionsort ist ein Divide-and-Conquer Verfahren.		

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 2 (Komplexität, Rekursion) *12 Punkte*

Bestimmen Sie für die nachstehenden Prozeduren folgende Informationen:

- Berechnen Sie die Anzahl der Aufrufe von tuwas() für n=4.
- Bestimmen Sie die Anzahl der Aufrufe von tuwas() als Funktion von n.
- Bestimmen Sie die asymptotische Zeitkomplexität in O-Notation unter der Annahme, dass die asymptotische Zeitkomplexität von tuwas() O(1) ist.

```
void proz1(int n)
{
    for (int a=0; a<n; a++)
        for (int b=0; b<=a; b++)
            tuwas();

}
```

```
void proz2(int n)
{
    if (n > 0)
       proz2(n-1);

    tuwas();

    if (n > 0)
       proz2(n-1);
}
```

Methode	**Anzahl Aufrufe für n=4**	**Als Funktion von n**	**O-Notation**
proz1			
proz2			

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 3 (Listen) *16 Punkte*

Betrachten Sie die folgende Implementierung einer einfach verketteten Liste:

```
public class Link
{
   public String daten;
   public Link naechster;

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

public class Liste
{
   private Link anfang, ende;

   public Liste(){
      // Leere Liste besitzt zwei Dummy-Elemente
      anfang = new Link(null, null);  // 1. Dummy-Element
      ende = new Link(null, null);    // 2. Dummy-Element
      anfang.naechster = ende;
   }

   public void einfuegen(String s)
   {
      assert(s!=null);
      // Aufgabenteil a)
   }

   public void anfuegen(String s)
   {
      assert(s!=null);
      // Aufgabenteil b)
   }

}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

a) Vervollständigen Sie die Methode `einfuegen`, die ein neues Element hinter dem ersten Dummy-Element einfügen soll.
Hinweis: Sie können davon ausgehen, dass der Parameter s ungleich null ist.

```
public void einfuegen(String s)
{
   assert(s!=null);









}
```

b) Vervollständigen Sie die Methode `anfuegen`, die ein neues Element vor dem zweiten Dummy-Element einfügen soll.
Hinweis: Sie können davon ausgehen, dass der Parameter s ungleich null ist.

```
public void anfuegen(String s)
{
   assert(s!=null);









}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 4 (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;

   // … übliche Methoden zum Einfügen und Entfernen

   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)
   }
}
```

a) Vervollständigen Sie die Methode `groessterSchluessel`, die iterativ den größten Schlüssel im Binärbaum bestimmen soll.
Hinweis: Sie können davon ausgehen, dass die Wurzel existiert.

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









}
```

b) Vervollständigen Sie die Methode `istInnererKnoten`, die prüft, ob `knoten` ein innerer Knoten ist, d.h. mindestens ein Kind besitzt.

```
private boolean istInnererKnoten(Knoten knoten)
{









}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

c) Vervollständigen Sie die Methode `anzahlInnereKnoten`, die rekursiv die Anzahl der inneren Knoten im Teilbaum mit der Wurzel `knoten` bestimmen soll.

```
private int anzahlInnereKnoten(Knoten knoten)
{




}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 5 (Sortieren) *10 Punkte*

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

Implementierung	**Name**	**Anzahl Vergleiche im Worst-Case**

`public static void sortiere(int[] feld)`
`{`
`  for (int i=0; i<feld.length; i++)`
`  for (int j=feld.length-1; j>0; j--)`
`  {`
`  if (feld[j]>feld[j-1]) {`
`  int tmp=feld[j];`
`  feld[j]=feld[j-1];`
`  feld[j-1]=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 mehr Zeilen als erforderlich sind.

2	**8**	**4**	**9**	**5**

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 6 (Graphen) — *17 Punkte*

Folgender Graph ist durch seine Adjazenzmatrix gegeben:

Von \ Nach	A	B	C	D	E	F
A			1			
B	1			1		1
C					1	
D	1				1	
E						
F				1		

a) Zeichnen Sie den gegebenen Graphen.

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

c) Führen Sie für den zugehörigen ungerichteten Graphen eine Breitensuche beginnend ab C durch und notieren Sie für jeden Schritt den Zustand der Warteschlange.

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Zusatzblatt