Name, Vorname, Matrikelnummer Studiengang

Klausur
Algorithmen und Datenstrukturen
SS 2019 – 22.08.2019

Hinweise:

- Die Bearbeitungszeit beträgt 120 Minuten.
- Zum Bestehen der Klausur sind 50 Punkte erforderlich.
- Schreiben Sie auf die ersten beiden Blätter Ihren Namen, Matrikelnummer und Studiengang.
- Erlaubte Hilfsmittel: keine
- Lösen Sie nicht die Klammerung der Klausur!
- Tragen Sie Ihre Lösungsvorschläge in die Klausurvorlage (evtl. auf den Rückseiten) ein. Weiteres Papier können Sie bei der Klausuraufsicht anfordern.
- 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)	12	
2 (Graphen)	18	
3 (Listen/Generics)	14	
4 (Komplexität)	12	
5 (Rekursion und Komplexität)	14	
6 (B-Bäume)	13	
7 (Sortieren)	10	
8 (AVL-Bäume)	12	
Summe	105	

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 1 (Multiple Choice, Wissen) *12 Punkte*

a) Kreuzen Sie in den folgenden Teilaufgaben an, welche Lösungen richtig und welche falsch sind. Für jede korrekt markierte Aussage erhalten Sie 0,5 Punkte, für jede falsch markierte werden 0,5 Punkte abgezogen. Pro Teilaufgabe erhalten Sie mindestens 0 Punkte.

Bewerten Sie folgende Aussagen	**Richtig**	**Falsch**
Für Collections vom Typ LinkedList ist Einfügen am Anfang in Zeit O(1) möglich.		
Für Collections vom Typ ArrayList ist Einfügen am Anfang in Zeit O(1) möglich.		
Für Collections vom Typ LinkedList ist Entfernen am Ende in Zeit O(1) möglich.		
Für Collections vom Typ ArrayList ist Entfernen am Ende in Zeit O(1) möglich.		

Für welche der folgenden Sortierverfahren ist der Zeitaufwand abhängig von der Eingabeverteilung (sensible Verfahren)?	**sensibel**	**Nicht sensibel**
Mergesort		
Selectionsort		
Quicksort		
Insertionsort		

Bewerten Sie folgende Aussagen	**Richtig**	**Falsch**
Die Interpolationssuche ist immer schneller als die binäre Suche.		
Bei der Interpolationssuche wird auch der gesuchte Schlüssel in die Berechnung des Vergleichselements einbezogen.		
Das Suchen in einer Hash-Tabelle ist im Worst-Case schneller als das Suchen in einem AVL-Baum.		
Das Suchen in einer Hash-Tabelle besitzt im Mittel eine Zeitkomplexität von O(1).		

Welche der folgenden Aussagen sind korrekt?	**Richtig**	**Falsch**
Jede Funktion ist berechenbar.		
Das Halte-Problem ist entscheidbar.		
Das Rucksack-Problem ist NP-vollständig.		
NP ist eine Teilmenge von P.		

Name, Vorname, Matrikelnummer Studiengang

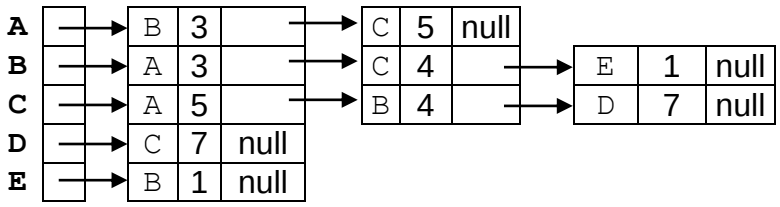
b) Wozu dient eine Hash-Funktion?

c) Was versteht man bei einer Hash-Tabelle unter einer Kollision?

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 2 (Graphen) *18 Punkte*

Folgender Graph ist durch seine Adjazenzlistenstruktur gegeben:



a) Geben Sie die zugehörige Adjazenzmatrix an.

b) Kann es sich hierbei um einen ungerichteten Graphen handeln? Begründen Sie Ihre Antwort anhand der Adjazenzmatrix.

c) Zeichnen Sie den gegebenen Graphen.

d) Enthält der Graph Zyklen (Begründung)?

e) Führen Sie auf dem obigen Graphen den Algorithmus von Kruskal zur Bestimmung eines minimalen Spannbaumes aus. Geben Sie neben der Kantenreihenfolge den nach jedem Schritt erzeugten Teilgraphen und die zugehörige Union-Find-Datenstruktur an.

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 3 (Listen/Generics) *14 Punkte*

Betrachten Sie die folgende Implementierung einer einfach verketteten Liste:

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

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

public class Liste
{
   private Link anfang, ende;

    public Liste()
    {
       anfang = ende = null;
    }

   public void anfuegen(int daten)
    {
       // Aufgabenteil a)
    }

   public Liste kopieren()
    {
       // Aufgabenteil b)
    }
}
```

Name, Vorname, Matrikelnummer Studiengang

a) Ergänzen Sie die Klasse Liste um eine Methode anfuegen, die ein Element am Ende der Liste anfügt.

```
public void anfuegen(int daten)
{

}
```

b) Ergänzen Sie nun die Klasse Liste um eine weitere Methode, die die Liste kopiert. Es soll eine neue Liste erzeugt werden, die Kopien aller Elemente der Original-Liste enthält. **Hinweis**: Die Original-Liste darf nicht verändert werden!

```
public Liste kopieren()
{

}
```

c) Wie müssen die obigen Klassen verändert werden, damit die Liste nicht nur Daten vom Typ `int` sondern Daten beliebiger Referenzdatentypen enthalten kann. Geben Sie nur die modifizierten Klassen, Attribute und Methodenköpfe (keine Methodenrümpfe) an.

```
}
```

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 4 (Komplexität) *12 Punkte*

Bestimmen Sie für die nachstehenden Prozeduren folgende Informationen:

- Berechnen Sie die Anzahl der Aufrufe von tuwas() für n=8.
- Bestimmen Sie die Anzahl der Aufrufe von tuwas() als Funktion von n. Hinweis: Sie können für Ihre Formel davon ausgehen, dass n eine Zweierpotenz ist.
- 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<20; b++)
            tuwas();

}
```

```
void proz2(int n)
{
    for (int a=1; a<=n; a++)
        for (int b=0; b<a; b++)
            tuwas();

}
```

```
void proz3(int n)
{
    while (n>1)
    {
        tuwas();
        n /= 2;
    }
}
```

```
void proz4(int n)
{
    for (int a=1; a<=n; a*=2)
        for (int b=0; b<2*n; b++)
            tuwas();
}
```

Methode	**Anzahl Aufrufe für n=8**	**Als Funktion von n**	**O-Notation**
proz1			
proz2			
proz3			
proz4			

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 5 (Rekursion und Komplexität) *14 Punkte*

Gegeben ist die folgende rekursiv definierte Funktion $f(n)$.

$$f(n) = \begin{cases} 1 & \text{für n} = 1 \\ f(n-1) \cdot f(n-1) + 1 & \text{für } n > 1 \end{cases}$$

a) Berechnen Sie f(3) und f(4).

f(3) =

f(4) =

b) Programmieren Sie die Funktion f <u>rekursiv</u> in Java, so dass Sie eine Zeitkomplexität von $O(2^n)$ und eine Rekursionstiefe von O(n) besitzt.

```
public static int fRekursiv1(int n)
{
   assert(n>=1);









}
```

Name, Vorname, Matrikelnummer Studiengang

c) Programmieren Sie die Funktion f <u>rekursiv</u> in Java, so dass Sie eine Zeitkomplexität von O(n) und eine Rekursionstiefe von O(n) besitzt.

```
public static int fRekursiv2(int n)
{
   assert(n>=1);









}
```

d) Programmieren Sie die Funktion f <u>iterativ</u> in Java, so dass Sie eine Zeitkomplexität von O(n) besitzt.
Hinweis: Funktionen der Klasse `Math` dürfen nicht benutzt werden.

```
public static int fIterativ(int n)
{
   assert(n>=1);









}
```

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 6 (B-Bäume) *13 Punkte*

Betrachten Sie die folgende teilweise Implementierung eines B-Baumes:

```
public class BKnoten
{
   public int[] schluessel;
   public BKnoten[] kinder;

   // Für Blätter
   public BKnoten(final int[] schluessel)
   {
      assert(schluessel!=null);

      this.schluessel = schluessel;
      this.kinder = new BKnoten[schluessel.length + 1];
   }

   // Für innere Knoten
   public BKnoten(final int[] schluessel, final BKnoten[] kinder)
   {
      assert(schluessel != null);
      assert(kinder != null);
      assert(schluessel.length+1 == kinder.length);

      this.schluessel = schluessel;
      this.kinder = kinder;
   }
}

public class BBaum
{
   private BKnoten wurzel;

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

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

   private int bestimmeAnzahlSchluessel(BKnoten knoten) {
      // Aufgabenteil b)
   }
}
```

Name, Vorname, Matrikelnummer Studiengang

a) Ergänzen Sie die Klasse BBaum um eine Methode

`public int bestimmeKleinstenSchluessel(),`

die iterativ den kleinsten Schlüssel im BBaum bestimmt.

Hinweis: Sie können davon ausgehen, dass die Wurzel mindestens einen Schlüssel enthält.

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









}
```

b) Ergänzen Sie die Klasse BBaum um eine Methode

`private int bestimmeAnzahlSchluessel(BKnoten knoten),`

die rekursiv die Anzahl der Schlüssel im Teilbaum mit der Wurzel `knoten` bestimmt.

```
private int bestimmeAnzahlSchluessel(BKnoten knoten)
{
```

Name, Vorname, Matrikelnummer Studiengang

```
}
```

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 7 (Sortieren) *10 Punkte*

a) In der folgenden Tabelle ist die Implementierung eines Sortieralgorithmus angegeben. Ergänzen Sie die Tabelle um den Namen des Verfahrens, asymptotische Anzahl an Vergleichen im besten (best case) und im schlechtesten Fall (worst case) sowie die Angabe, ob das Verfahren stabil ist.

Implementierung	**Name**	**Anz. Vergleiche** *O-Notation*		**Stabil** *Ja/Nein*
		Best case	**Worst case**	

`public static void sortiere(int[] feld)`
`{`
`  for (int i=1; i<feld.length; i++)`
`  {`
`  int tmp=feld[i];`
`  int j = i-1;`
`  while (j>=0 && feld[j]<tmp)`
`  {`
`  feld[j+1] = feld[j];`
`  j--;`
`  }`
`  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 und kreisen Sie den jeweiligen Wert des lokalen Attributs `tmp` ein. Die Tabelle enthält mehr Zeilen als erforderlich sind.

4	**8**	**5**	**9**	**6**	**3**	**2**	**8**	**7**

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 8 (AVL-Bäume) *12 Punkte*

a) Konstruieren Sie einen AVL-Baum, der mindestens 5 Knoten besitzt und im linken Teilbaum der Wurzel genau doppelt so viele Knoten enthält wie im rechten Teilbaum der Wurzel. Geben Sie für jeden Knoten den Schlüssel und den Balancefaktor an.

b) Zeichnen Sie den AVL-Baum der entsteht, wenn man in einen anfangs leeren Baum nacheinander die Schlüssel 5, 15 und 20 einfügt. Zeichnen Sie auch den Baum vor jeder Rotation. Geben Sie jeweils die Rotationsart an.

c) Zeichnen Sie den AVL-Baum, nachdem die Schlüssel 10 und 7 eingefügt wurden. Zeichnen Sie auch den Baum vor jeder Rotation. Geben Sie jeweils die Rotationsart an. Doppelrotationen können in einem Schritt ausgeführt werden.

d) Zeichnen Sie den AVL-Baum, nachdem der Schlüssel 12 eingefügt wurde. Geben Sie die Rotationsart an. Doppelrotationen können in einem Schritt ausgeführt werden.