Prüfung AuD Sose 2017 - Matrikelnummer

		0	1	2	3	4	5	6	7	8	9
Bitte schreiben Sie Ihre Matrikelnummer in dieses Feld und kreuzen Sie im rechten Bereich Ihre Matrikelnummer an (erste Ziffer = oberste Zeile). Schreiben zusätzlich auf das nächste Blatt und zur Sicherheit auf jedes zu bewertende Blatt Ihre Matrikelnummer.	___	○	○	○	○	○	○	○	○	○	○
	___	○	○	○	○	○	○	○	○	○	○
	___	○	○	○	○	○	○	○	○	○	○
	___	○	○	○	○	○	○	○	○	○	○
	___	○	○	○	○	○	○	○	○	○	○
	___	○	○	○	○	○	○	○	○	○	○
	___	○	○	○	○	○	○	○	○	○	○

Prüfung AuD Sose 2017 - Ab hier: NUR für die Korrektur!

Nabla Aufgabe 1	
Erreichte Punktzahl: ⬠ 0 ⬠ 1 ⬠ 2 ⬠ 3 ⬠ 4 ⬠ 5 ⬠ 6 ⬠ 7 ⬠ 8 ⬠ 9 ⬠ 10	
Nabla Aufgabe 2	
Erreichte Punktzahl: ⬠ 0 ⬠ 1 ⬠ 2 ⬠ 3 ⬠ 4 ⬠ 5 ⬠ 6 ⬠ 7 ⬠ 8 ⬠ 9 ⬠ 10	
Nabla Aufgabe 3	
Erreichte Punktzahl: ⬠ 0 ⬠ 1 ⬠ 2 ⬠ 3 ⬠ 4 ⬠ 5 ⬠ 6 ⬠ 7 ⬠ 8 ⬠ 9 ⬠ 10	
Programmieraufgabe 1	☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J ☐ K ☐ L ☐ M ☐ N ☐ O
Programmieraufgabe 1 Status	☐ unerlaubte S.Konstrukte ☐ Verletzung Regeln: Schleifen/Rekursion ☐ Grob unvollständig ☐ Übermäßige Fehleranzahl aller Art ☐ Konzepte der Datenstrukturen n. verstanden ☐ Sonstiges
Endpunktzahl Programmieraufgabe 1	Erreichte Punktzahl: ⬠ 0 ⬠ 1 ⬠ 2 ⬠ 3 ⬠ 4 ⬠ 5 ⬠ 6 ⬠ 7 ⬠ 8 ⬠ 9 ⬠ 10 ⬠ 11 ⬠ 12 ⬠ 13 ⬠ 14 ⬠ 15
Programmieraufgabe 2	☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H ☐ I ☐ J ☐ K ☐ L ☐ M ☐ N ☐ O
Programmieraufgabe 2 Status	☐ unerlaubte S.Konstrukte ☐ Verletzung Regeln: Schleifen/Rekursion ☐ Grob unvollständig ☐ Übermäßige Fehleranzahl aller Art ☐ Konzepte der Datenstrukturen n. verstanden ☐ Sonstiges
Endpunktzahl Programmieraufgabe 2	Erreichte Punktzahl: ⬠ 0 ⬠ 1 ⬠ 2 ⬠ 3 ⬠ 4 ⬠ 5 ⬠ 6 ⬠ 7 ⬠ 8 ⬠ 9 ⬠ 10 ⬠ 11 ⬠ 12 ⬠ 13 ⬠ 14 ⬠ 15
Theorie Aufgabe 6a	☐ A ☐ B
Theorie Aufgabe 6b	☐ A ☐ B ☐ C
Theorie Aufgabe 6c	☐ A ☐ B ☐ C ☐ D
Theorie Aufgabe 6d	☐ A ☐ B ☐ C ☐ D ☐ E ☐ F
Theorie Aufgabe 6e	☐ A ☐ B ☐ C ☐ D ☐ E
Theorie Aufgabe 7a	☐ A ☐ B ☐ C ☐ D ☐ E
Theorie Aufgabe 7b	☐ A ☐ B ☐ C
Theorie Aufgabe 7c	☐ A ☐ B ☐ C ☐ D
Theorie Aufgabe 7d	☐ A ☐ B
Theorie Aufgabe 7e	☐ A ☐ B ☐ C ☐ D ☐ E ☐ F

Klausur Algorithmen und Datenstrukturen

Sommersemester 2017

27. September 2017

Ihre Matrikelnummer:

Achtung: Geben Sie keine weiteren persönlichen Daten außer der Matrikelnummer an!

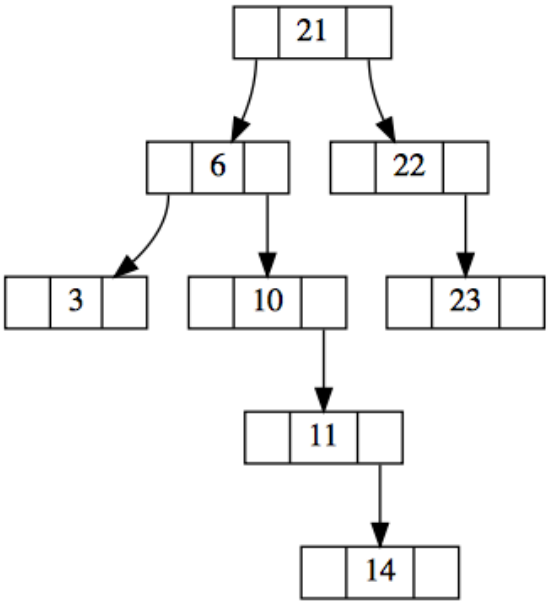
Genereller Hinweis: Sie können selbstverständlich auch Zwischen- und Nebenergebnisse in Ihre Abgabe schreiben, um leichter zum Ergebnis zu kommen. Stellen Sie dann aber die eigentliche Lösung klar heraus.

Diese Seite ab hier nur von den Korrektoren auszufüllen!

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Summe**
Maximale Punktzahl	**10**	**10**	**10**	**15**	**15**	**20**	**20**	**100**
Erreichte Punktzahl								

Aufgabe 1: 10 Punkte

Aus der Vorlesung kennen Sie den Algorithmus Binary search tree: traverse. Geben Sie den Stack und die Liste L des Algorithmus nach der Iteration i=12 an (die erste Iteration ist i=0).



Ihr Ergebnis:

Key	seenChildren

L :

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Freie Seite für Ihre Nebenrechnungen (werden nicht gewertet):

Aufgabe 2: 10 Punkte

In der Vorlesung haben Sie das Verfahren Double Hashing
als typisches Design für Hashfunktionen kennengelernt:

$$F(i, 15, K) := (F_1(15, K) + (i-1) \cdot F_2(15, K)) \bmod 15$$
$$F_1(N_{\max}, K) := (K \bmod N_{\max})$$
$$F_2(N_{\max}, K) := K + (K \bmod N_{\max})$$

Fügen Sie die folgenden Werte in dieser Reihenfolge ein:

15, 31, 21, 40, 12, 25, 41, 43, 29, 34, 42.

Ihr Ergebnis:

0	8
1	9
2	10
3	11
4	12
5	13
6	14
7	

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Freie Seite für Ihre Nebenrechnungen (werden nicht gewertet):

Aufgabe 3: 10 Punkte

Aus der Vorlesung kennen Sie den Algorithmus 'insert' der Datenstruktur Heap. Gegeben sei nun folgender Heap mit N = 15:

Index	1	2	3	4	5	6	7	8	9	10	11	12
ID	3	4	1	8	5	6	7	2	9	10	11	12
Key	**3**	**26**	**19**	**33**	**46**	**33**	**64**	**80**	**93**	**73**	**82**	**43**

Fügen Sie den Schlüssel key=2 ein. Geben Sie den Heap und das Array Positions nach der Iteration i=3 an (i=1 ist Einfügen hinten im Array).

Ihr Ergebnis:

Index	1	2	3	4	5	6	7	8
ID								
Key								

Index	9	10	11	12	13	14	15
ID							
Key							

Index	1	2	3	4	5	6	7	8
Position								

Index	9	10	11	12	13	14	15
Position							

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Freie Seite für Ihre Nebenrechnungen (werden nicht gewertet):

Aufgabe 4: 15 Punkte

Aus der Vorlesung und vom Java-Übungsblatt kennen Sie lineare Listen und diese Klasse für Listenelemente:

```
public class ListItem <T>  {
     public T            key;
     public ListItem<T>  next;
}
```

Konkrete Aufgabe: Schreiben Sie eine Methode

```
<T> ListItem<ListItem<T>> decompose ( ListItem<T> list, T key,
                                      Comparator<T> cmp )
```

Die Methode darf ohne Überprüfung davon ausgehen, dass `list` auf den Kopf einer korrekt gebildeten Liste verweist, die auch leer sein kann (also `list == null` möglich), dass `key != null` ist und `cmp` auf ein Objekt einer Klasse verweist, die `Comparator<T>` implementiert. Die Methode soll die Liste, auf deren Kopf `list` verweist, in eine Liste von nichtleeren Teillisten zerlegen, dabei aber alle Vorkommen von `key` auslassen. Jede Teilliste ist ein Abschnitt zwischen zwei Vorkommen von `key` (`key` kann auch am Anfang und Ende der Liste beliebig häufig vorkommen).

Beispiel: Liste von Character, Leerzeichen als Key:

(␣Hello,␣World,␣␣how␣are␣␣␣you?) → ((Hello,)(World,)(how)(are)(you?))

Verbindliche Anforderungen:

- Die Methode muss *iterativ* sein, das heißt, Rekursion ist nicht erlaubt.
- Neben der Methode `compare` von `Comparator<T>` darf keine Funktionalität aus der Standardbibliothek von Java oder anderen Bibliotheken verwendet werden.
- Es dürfen keine neuen Objekte von `ListItem<T>` erzeugt oder Werte von `T` kopiert werden, das heißt, die Aufgabe muss durch Modifikationen des `next`-Attributs gelöst werden (Objekte von `ListItem<ListItem<T>>` dürfen natürlich mit `new` erzeugt werden).
- Die asymptotische Komplexität muss insgesamt linear bleiben.

Erinnerung: Methode `compare` von `Comparator<T>` liefert $+1$ (bzw. -1) zurück, falls der erste Parameter größer (bzw. kleiner) als der zweite ist, bei Gleichheit 0.

Von den Korrektoren auszufüllen:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Ihre Lösung schreiben Sie auf die folgenden Seiten bis zur nächsten Aufgabe:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Aufgabe 5: 15 Punkte

Aus der Vorlesung und vom Java-Übungsblatt kennen Sie binäre Suchbäume. Betrachten Sie folgende Klasse für Baumknoten:

```
public class TreeNode <T> {
    public T key;
    public TreeNode<T> left;
    public TreeNode<T> right;
    public TreeNode<T> predecessor;
    public TreeNode<T> successor;
}
```

Konkrete Aufgabe: Schreiben Sie eine Methode

```
void setPredecessorsAndSuccessors ( TreeNode<T> root )
```

So wie immer, wird durch die Attribute `left` und `right` ein binärer Baum gebildet (der auch leer sein kann, das heißt, `root==null` ist möglich). Die Methode darf davon ausgehen, dass dieser Baum korrekt gebildet ist und die Suchbaumeigenschaft erfüllt, dass `predecessor` und `successor` in allen Knoten `null` sind und dass alle Werte `key` paarweise verschieden sind. Für jeden Knoten `node` im Baum soll die Methode die Attribute `predecessor` und `successor` setzen, und zwar auf den Knoten, in dem der unmittelbare Vorgänger bzw. Nachfolger von `node.key` in aufsteigender Sortierreihenfolge steht. Beim Knoten mit dem kleinsten bzw. größten Schlüsselwert soll dagegen `null` in `predecessor` bzw. `successor` stehen.

Verbindliche Anforderungen:

- Die Aufgabe soll vollständig durch Rekursion gelöst werden, das heißt, Schleifen sind nicht zulässig.
- Es darf keine Funktionalität aus der Standardbibliothek von Java oder anderen Bibliotheken verwendet werden.
- Abgesehen von der Setzung der Attribute `predecessor` und `successor` darf nichts am Baum verändert werden.
- Die asymptotische Komplexität darf insgesamt höchstens quadratisch in der Anzahl der Baumknoten sein.

Unverbindlicher Hinweis: Wenn Sie von einem Knoten aus den Vorgänger weiter unten im Baum finden, können Sie gleich dessen Nachfolger setzen (und umgekehrt). Dadurch ist es nicht notwendig, von einem Baumknoten aus den Vorgänger oder Nachfolger weiter oben im Baum zu suchen.

Von den Korrektoren auszufüllen:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Ihre Lösung schreiben Sie auf die folgenden Seiten bis zur nächsten Aufgabe:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Aufgabe 6: 20 Punkte

In gewisser Weise transponiert `foo` zweidimensionale Arrays (die nicht unbedingt Matrizen sein müssen):

```
public double[][] foo ( double[][] a ) {
    int m = 0;
    for ( int i = 0; i < a.length; i++ )
        if ( a[i].length > m )
            m = a[i].length;
    int[] b = new int [ m ];   // alle Komponenten sind 0
    bar ( a, b, 0, 0 );
    double[][] c = new double [ m ] [];
    for ( int i = 0; i < m; i++ )
        c[i] = new double [ b [ i ] ];
    fooBar ( a, c, m );
    return c;
}


public void bar ( double[][] a, int[] b, int i, int j ) {
   if ( i >= a.length || j >= a[i].length )
       return;
   if ( j == 0 )
       bar ( a, b, i+1, 0 );
   b[j]++;
   bar ( a, b, i, j+1 );
}


public void fooBar ( double[][] a, double[][] c, int m ) {
    int[] d = new int [ m ];   // alle Komponenten sind 0
    for ( int i = 0; i < a.length; i++ )
        for ( int j = 0; j < a[i].length; j++ ) {
            c [ j ] [ d[j] ] = a[i][j];
            d[j]++;
    }
}
```

Konkrete Aufgaben (a) – (e)

Formulieren Sie jeweils **kurz und bündig, aber präzise und unmissverständlich**:

(a) Die Voraussetzungen, die der Parameter `a` von `foo` erfüllen muss, damit alle drei Methoden wohldefiniert sind. **2 Punkte**

(b) Die Voraussetzungen, die der Parameter `c` von `fooBar` erfüllen muss, damit `fooBar` wohldefiniert ist. **3 Punkte**

(c) Den Output von `foo` (also eine Präzisierung der einleitenden Umschreibung vor dem Java-Code) sowie die Nachbedingung von `bar`. **4 Punkte**

Unverbindlicher Hinweis: Die Nachbedingung von `bar` sollte einfacher zu formulieren sein, wenn Sie zwei Fallunterscheidungen einbauen: $i = 0$ vs. $i > 0$ und `b[`h`]` mit $h < j$ vs. `b[`h`]` mit $h \geq j$.

(d) Teile des Korrektheitsbeweises für `bar`: Invariante und Variante und für den Beweis der Invariante den Induktionsanfang und den Induktionsschritt. **6 Punkte**

Verbindliche Hinweise: Benennen Sie die Indizes `i` und `j` sowie den Inhalt von `b[`j`]` überall *explizit*. Arbeiten Sie die Induktionsvoraussetzung *explizit* in den Induktionsschritt ein.

(e) Sei n die Länge von `a` und m das Maximum aus `a[0].length, a[1].length,...,` `a[`$n-1$`].length`. Geben Sie Fälle an, in denen die asymptotische Gesamtkomplexität der Methode `foo` (inklusive aller Aufrufe von `bar` und `foobar`) den Best Case bzw. den Worst Case erreicht, und geben Sie jeweils die Komplexität (als Θ) an. Begründen Sie Ihre Θ-Angaben. **5 Punkte**

Von den Korrektoren auszufüllen:

(a)		(b)			(c)				(d)						(e)				
A	B	A	B	C	A	B	C	D	A	B	C	D	E	F	A	B	C	D	E

Ihre Lösung schreiben Sie auf die folgenden Seiten bis zur nächsten Aufgabe:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Aufgabe 7: 20 Punkte

(a) Gegeben seien die Funktionen f und g durch $f(x) = x^2/\log(x)$ und $g(x) = x \cdot \log(x)$ für alle $x \in \mathbb{R}_0^+$, wobei $\log(\cdot)$ den natürlichen Logarithmus bezeichnet. Beweisen Sie mathematisch, dass f asymptotisch schneller wächst als g. **5 Punkte**

Verbindlicher Hinweis: durch mehrfache Anwendung der Regel von l'Hospital.

Unverbindlicher Hinweis: Sie können ohne Beweis voraussetzen, dass der mathematische Ausdruck $(a \cdot h(x) + b)/(c \cdot h(x) + d)$ gegen a/c geht für $x \to \infty$.

Erinnerung Produkt- und Quotientenregel: Für Funktion u und v ist die Ableitung von $u \cdot v$ gegeben durch $u'v + v'u$ und die von u/v durch $(u' \cdot v - v' \cdot u) \ / \ v^2$.

(b) Gegeben seien die Funktionen f und g durch $f(x, y) = x^2 \cdot e^y$ und $g(x, y) = e^x \cdot y^2$ für alle x, $y \in \mathbb{R}_0^+$. Beweisen Sie mathematisch, dass f und g asymptotisch nicht vergleichbar sind. **3 Punkte**

Unverbindicher Hinweis: Betrachten Sie die Fälle $x = y^2$ und $y = x^2$ und verwenden Sie $e^{(a^2)} \geq e^a \cdot e^a$ für $a \geq 2$ ohne Beweis.

(c) *Zu B-Bäumen*: Wie Sie wissen, muss jeder Knoten außer der Wurzel mindestens $N-1$ Schlüsselwerte enthalten, wenn N die Ordnung des B-Baumes ist. Was würde so alles nicht funktionieren – und warum nicht –, wenn jeder Knoten weiterhin Platz für $2N-1$ Schlüsselwerte hätte, aber (außer bei der Wurzel) mindestens N Schlüsselwerte enthalten müsste? **4 Punkte**

(d) *Zu All-Pairs-Shortest-Paths*: Wie – und an welchen Stellen genau – drückt es sich in der Ergebnismatrix aus, wenn der zugrundeliegende gerichtete Graph nicht stark zusammenhängend ist, das heißt, wenn es nicht von jedem Knoten zu jedem anderen Knoten einen Pfad gibt? **2 Punkte**

(e) Wie Sie wissen, können Sie aus jedem Minimierungsproblem ein Entscheidungsproblem konstruieren, indem Sie eine Zahl als zusätzlichen Input hinzugeben und die Entscheidungsfrage stellen, ob es eine Lösung gibt, deren Wert nicht höher als diese Zahl ist. Führen Sie nun das Prinzip der polynomiellen Reduktion anhand des folgenden Beispiels vor: Das Minimal-Spannbaum-Problem soll auf das Steinerbaumproblem polynomiell reduziert werden. Formulieren Sie dazu als erstes beide Probleme jeweils als Optimierungs- und als Entscheidungsproblem. **6 Punkte**

Hinweis: Die Argumentation ist eigentlich sehr einfach – wenn Sie kompliziert denken, sind Sie wahrscheinlich auf Abwegen. Führen Sie das Prinzip der polynomiellen Reduktion trotzdem explizit durch, auch wenn die einzelnen zu berücksichtigenden Punkte offensichtlich zu sein scheinen.

Von den Korrektoren auszufüllen:

(a)					(b)			(c)				(d)		(e)					
A	B	C	D	E	A	B	C	A	B	C	D	A	B	A	B	C	D	E	F

Ihre Lösung schreiben Sie auf die folgenden Seiten:

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____

Ihre Matrikelnummer (zu Ihrer Sicherheit):_____