Hinweise zur Bearbeitung der Klausur zum Kurs 1661 Datenstrukturen I

Bitte lesen Sie sich diese Hinweise vollständig und aufmerksam durch, bevor Sie mit der Bearbeitung der Klausur beginnen. Beachten Sie insbesondere die Punkte 7 und 8!

1. Die Klausurdauer beträgt 2 Stunden.
2. Prüfen Sie bitte die Vollständigkeit Ihrer Unterlagen. Die Klausur umfaßt:
   - 2 Deckblätter
   - diese Hinweise
   - 1 Formblatt für eine Teilnahmebescheinigung zur Vorlage beim Finanzamt
   - 3 Aufgaben auf den Seiten 2 – 4
3. Bevor Sie mit der Bearbeitung der Klausuraufgaben beginnen, füllen Sie bitte die folgenden Teile der Klausur aus:
   (a) *sämtliche* Deckblätter mit Name, Anschrift sowie Matrikelnummer. Markieren Sie vor der Abgabe auf allen Deckblättern die von Ihnen bearbeiteten Aufgaben.
   (b) die Teilnahmebescheinigung, falls Sie diese wünschen.
4. Schreiben Sie Ihre Lösungen auf ihr eigenes Papier (DIN A4) und nicht auf die Seiten mit den Aufgabenstellungen. Heften Sie vor Abgabe der Klausur die Deckblätter (und evtl. die Teilnahmebescheinigung) an Ihre Bearbeitung.
5. Schreiben Sie bitte auf jedes Blatt oben links Ihren Namen und oben rechts Ihre Matrikelnummer. Numerieren Sie Ihre Seiten bitte durch.
6. Wenden Sie bei der Lösung der Aufgaben, soweit dies möglich ist, die Algorithmen und Notationen aus den Kurseinheiten an.
7. Schreiben Sie, wenn ***Algorithmen*** gefordert sind, keine kompletten PASCAL- oder JAVA-Programme, sondern beschränken Sie sich auf die wesentlichen Teile des Algorithmus, die z.T. auch in Prosa formuliert werden können. Formulieren Sie Algorithmen aber so, daß die elementaren Einzelschritte erkennbar werden.

   Sparen Sie aber bei solchen Aufgaben nicht mit Kommentaren. Wenn Ihre Lösung aufgrund fehlender Kommentare nicht verständlich ist, führt das zu Punktabzug.
8. Vermeiden Sie in jedem Fall bei der Definition von Funktionen in ***Algebren*** PASCAL- oder JAVA-Programme sowie Algorithmen. Geben Sie lediglich mathematische Definitionen an wie sie im Kurstext verwandt worden sind!

   Ebenso sind Mengendefinitionen in mathematischer Mengennotation durchzuführen. Geben Sie auf keinen Fall Datentypdefinitionen an, und verwenden Sie keine konkreten Datenstrukturen!
9. Als Hilfsmittel sind nur unbeschriebenes Konzeptpapier und Schreibzeug (kein Bleistift!) zugelassen.
10. Es sind maximal 66 Punkte erreichbar. Zur Erlangung eines Übungsscheins bzw. Zertifikats benötigen Sie etwa 33 Punkte.

22 Punkte Aufgabe 1 Briefmarkensammlung

In dieser Aufgabe soll eine Algebra für eine Briefmarkensammlung angegeben werden. Duplikate sind in der Sammlung nicht erlaubt. Eine einzelne Briefmarke ist definiert durch einen Identifikator, einen Sammlerwert und einen Zustand (Schulnote). Der Identifikator sei nicht eindeutig für alle Marken, sondern für Marken desselben Typs gleich. Wenn gleiche Marken, die aber einen besseren Zustand haben, eingefügt werden sollen, dann werden die Marken mit schlechterem Zustand entfernt.

Folgende Operationen sollen unterstützt werden:
- *neueSammlung*: erzeugt eine leere Sammlung
- *neueMarke*: erzeugt eine neue Marke
- *MarkeEinfügen*: fügt eine Marke in eine Sammlung ein
- *MarkeEntnehmen*: löscht eine Marke aus einer Sammlung
- *MarkeBesser*: liefert *true*, falls die erste von zwei Marken desselben Typs besser ist als die zweite, sonst *false*
- *SammlungAuflösen*: löscht alle Marken aus der Sammlung
- *MarkeFinden*: gibt die gesuchte Marke zurück, wenn die Marke in der gegebenen Sammlung enthalten ist; ein Parameter der Operation ist eine Marke
- *GesamtwertBerechnen*: berechnet den Wert der gesamten Sammlung

6 Punkte (a) Geben Sie die Sorten und Operationen für die Algebra an.

8 Punkte (b) Geben Sie Trägermengen und Funktionen für die Algebra an.

8 Punkte (c) Geben Sie Algorithmen für die Funktionen aus (b) an. Die Lösung muß nicht Laufzeit-optimal sein.

25 Punkte Aufgabe 2 Sortierverfahren

5 Punkte (a) In dieser Aufgabe soll Mergesort analysiert werden, wenn die zu sortierende Menge als einfache Liste implementiert ist. So kann beispielsweise die Länge einer Liste nur durch einen vollständigen Durchlauf durch die Liste ermittelt werden. Die Mergesort-Implementierung soll zudem stabil sein, was es verbietet, die Elemente der zu sortierenden Liste im Divide-Schritt einfach abwechselnd auf die beiden Teillisten zu verteilen.
Folgendes Programmfragment beschreibt den Divide-Schritt einer solchen Mergesort-Implementierung:

```
public static List mergeSort(List unsorted){
   unsorted.first(); // gehe zum ersten Listenelement
   // Bestimmen der Listenlänge
   int length := 0;
   while( ! unsorted.onEnd() ){
        length++;
        unsorted.next();
   }
   if(length<2) return unsorted;
```

```
    List s1 = new List();
    List s2 = new List();
    unsorted.first();
    int pos = 0;
    while(!unsorted.onEnd()){
      if(pos<=length/2)
        s1.append(unsorted.currentElement());
      else
        s2.append(unsorted.currentElement());
      pos++;
      unsorted.next();
    }
    .....
}
```

• Wie ändert sich das Laufzeitverhalten von Mergesort durch den Aufwand, der im Divide-Schritt zusätzlich anfällt?

• Welchen Speicherplatzbedarf benötigt ein so implementierter Mergesort Algorithmus? Geben Sie eine entsprechende Rekursionsgleichung an und lösen Sie diese auf. Verwenden Sie als Maß für den Speicherplatz die maximale Gesamtanzahl der Listenelemente der gleichzeitig vorhandenen Listen. Zur Vereinfachung können Sie davon ausgehen, daß die Länge der ursprünglichen Liste eine Zweierpotenz ist. Der Speicherplatz für die Ausgangsliste soll unberücksichtigt bleiben.

(b) Sortieren Sie die Folge S mittels Mergesort. Geben Sie zunächst an, wie die Folge aufgeteilt und anschließend wieder zusammengesetzt wird. *7 Punkte*

S := 4 - 6 - 18 - 12 - 1 - 3 - 9 - 10 - 6 - 3 - 2 - 19

(c) Überlegen Sie, ob es günstiger ist, bei Mergesort die zu sortierende Menge in mehr als zwei Teilmengen aufzuspalten. Vergleichen Sie dazu die Laufzeit eines entsprechenden Algorithmus mit der Laufzeit einer Standard-Implementierung von Mergesort in O-Notation. *3 Punkte*

(d) Sortieren Sie die in Aufgabenteil (b) gegebene Folge S mittels (Standard-) Quicksort. Markieren Sie das Splitelement und geben Sie an, wie jeweils die beiden neuen Folgen aussehen. *10 Punkte*
Hinweis: Sie können zur Darstellung einen Binärbaum verwenden, bei dem jeder Knoten die gerade aktuelle Folge (mit markiertem Splitelement) und jeder Sohn eine der beiden neuen Teilfolgen enthält.
Das Zusammenfügen der Folgen braucht nicht angegeben zu werden.

19 Punkte **Aufgabe 3 Partitionen von Mengen**

Im Kurstext wird ein Datentyp *partition* zur Verwaltung einer Partition einer Menge vorgestellt. Dieser bietet die Operationen

empty: $\rightarrow$ *partition*

addcomp: *partition* $\times$ *compname* $\times$ *elem* $\rightarrow$ *partition*

merge: partition $\times$ *compname* $\times$ *compname* $\rightarrow$ *partition*

find: *partition* $\times$ *elem* $\rightarrow$ *compname*

an.

7 Punkte (a) Geben Sie einen Algorithmus an, der aus einer Menge M und einer Menge A von Äquivalenz-Anweisungen eine Zerlegung in Äquivalenzklassen berechnet. Benutzen Sie dabei die oben dargestellten Operationen. Interne Details einer Implementierung von *partition* über Arrays oder Bäume dürfen also nicht benutzt werden.

6 Punkte (b) Gegeben Sei die Menge $M = \{1, 2, 3, 4, 7, 8, 9\}$ und die Äquivalenz-Anweisungen $1 \equiv 4$, $4 \equiv 7$, $3 \equiv 9$. Der Datentyp *partition* sei mittels Bäumen implementiert. Stellen Sie eine Partition, die die Zerlegung in Äquivalenzklassen von M darstellt, graphisch dar.

6 Punkte (c) Implementieren Sie einen Algorithmus für die Operation *find*, der die Idee der Pfadkompression realisiert. Dabei soll die Ermittlung des Komponentennamens unberücksichtigt bleiben. Es soll also lediglich die Verzeigerung geändert werden.