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
   - 5 Aufgaben auf den Seiten 2–6
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 die Aufgabenstellungen (und evtl. die Teilnahmebescheinigung) vor Ihre Bearbeitung. **Sie müssen die Aufgabenstellung also mit abgeben!** Sie erhalten die Aufgabenstellungen zusammen mit Ihrer korrigierten Klausur zurück.
5. Schreiben Sie bitte auf jedes Blatt Ihrer Bearbeitung oben links Ihren Namen und oben rechts Ihre Matrikelnummer. Nummerieren 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 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 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 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 zugelassen. Die Reinschrift der Klausur darf **nicht mit Bleistift** erfolgen.
10. Durch Lösen der Aufgaben sind maximal 100 Punkte erreichbar. Sie dürfen damit rechnen einen Übungsschein bzw. ein Zertifikat zu erhalten, wenn Sie insgesamt mindestens 50 Punkte erreichen.

28 Punkte

Aufgabe 1 Algebra

Kunstmuseen verfügen über zahlreiche Bilder. Sie sollen eine Algebra entwerfen, die solche Kunstsammlungen verwalten kann. Eine Kunstsammlung umfasst mehrere Räume. Jeder Raum in einer Kunstsammlung hat eine eindeutige Nummer. Die Zahl der Bilder je Raum ist durch die Ausstellungsfläche des Raumes in Quadratmetern und die Größe der darin ausgestellten Bilder nach oben begrenzt. Jedes Bild verfügt über einen Titel, einen Maler und eine Größe in Quadratmetern.

Folgende Operationen sollen unterstützt werden:

1. *createPicture*: erzeugt aus einem Titel, der Größenangabe, dem Namen des Malers ein Bild,
2. *createCollection*: erzeugt eine leere Kunstsammlung,
3. *createRoom*: erzeugt aus einer Raumnummer und der maximalen Ausstellungsfläche einen Raum,
4. *insertPicture*: Fügt ein Bild in einen Raum ein, wenn noch genug Ausstellungsfläche vorhanden ist.
5. *insertRoom*: Fügt einen Raum zu einer Sammlung hinzu.
6. *deleteRoom*: Entfernt einen Raum aus einer Kunstsammlung.
7. *deletePicture:* Entfernt ein Bild aus einem Raum, wenn es darin enthalten ist.
8. *getArtist*: Gibt den Maler eines Bildes zurück.
9. *isEqual*: Prüft, ob zwei Bilder gleich sind.
10. *getMaxArea*: Gibt die maximale Ausstellungsfläche eines Raums zurück
11. *getUsedAred*: Gibt die genutzte Ausstellungsfläche eines Raums zurück.
12. *getCapacity*: Gibt die noch freie Ausstellungsfläche eines Raums zurück.
13. *containsPicture*: Prüft, ob ein Bild in einem Raum vorhanden ist.
14. *findPicture*: Gibt alle Räume einer Kunstsammlung, die das Bild enthalten, aus.

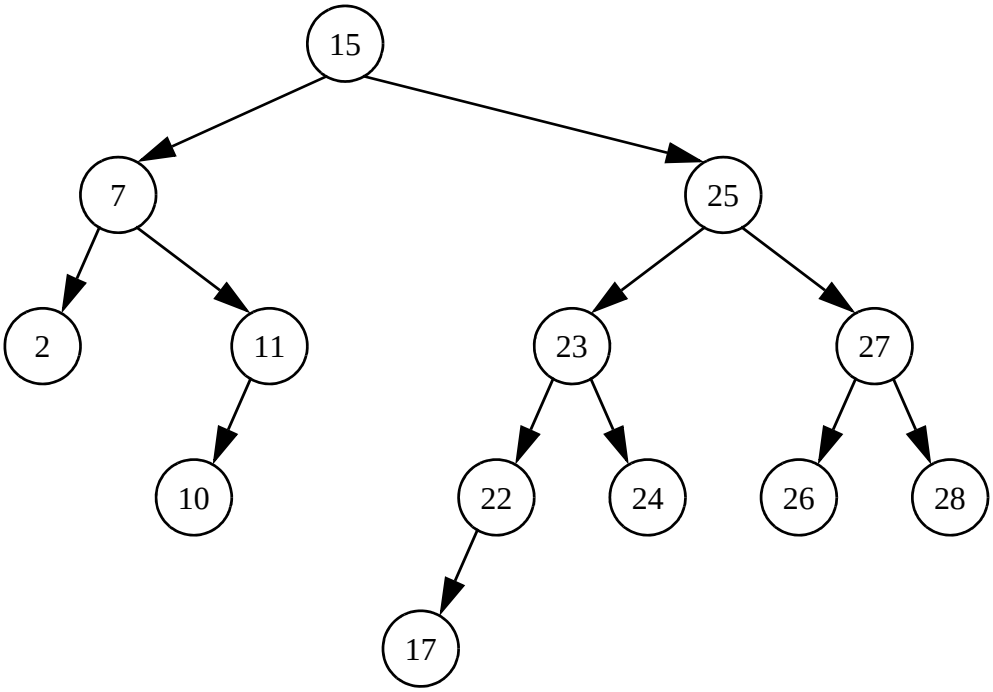
Formulieren Sie die Spezifikation aller erforderlichen Datentypen und Operationen als Algebra.

4 Punkte (a) Geben Sie die Sorten und Operatorsignaturen der Algebra an.

7 Punkte (b) Geben Sie Trägermengen für die Sorten an, die genau die geforderten Objekte enthalten. Dabei können Sie *int*, *real*, *bool* und *string* als gegeben voraussetzen.

17 Punkte (c) Geben Sie die Funktionen für die Operationen mit der oben aufgeführten Semantik an.

Aufgabe 2 AVL-Baum *30 Punkte*



(a) Löschen Sie aus oben gezeigtem AVL-Baum folgende Schlüssel in der angegebenen Reihenfolge. Zeichnen Sie den Baum vor und nach jeder Rebalancierung. Markieren Sie jeweils den Knoten, der aus der Balance geraten ist und geben Sie die jeweilige Balancierungsmethode an. Treten beim Löschen eines Schlüssels fortgesetzte Balancierungsoperationen auf, so genügt es, den Zustand vor der ersten und nach der letzten Operation darzustellen. *10 Punkte*

2, 24, 28, 25, 27

3 Punkte

(b) Geben Sie die Laufzeitkomplexität des im Folgenden beschriebenen, laufzeitoptimalen Algorithmus‘ *arrayToAVLtree*, der aus einem aufsteigend vorsortierten Array einen AVL-Baum erzeugt, in O-Notation als Funktion der Größe des Arrays *n* an.

```
1   algorithm arrayToAVLtree(A, lower, upper)
2     { A      ist ein Array mit aufsteigend sortierten,
3              disjunkten Schlüsselwerten,
4       lower ist der Index des ersten zu berücksichtigenden
5              Schlüsselwertes in A,
6       upper ist der Index nach dem letzten zu berücksichtigenden
7              Schlüsselwert in A, es gelte stets lower ≤ upper.
8       Der Algorithmus gibt einen AVL-Baum über den indizierten
9       Bereich zurück.
10    }
11    if (upper = lower) then
12      return empty;
13    end if;
14    if (upper - lower = 1) then
15      return (empty, A[lower], empty, 0)
16    end if;
17    if (upper - lower = 2) then
18      return ((empty, A[lower], empty, 0), A[upper - 1], empty, 1);
19    else
20      var mid;
21      mid := lower + ⌈(upper − 1 − lower)/2⌉ ;
22      T1 = arrayToAVLtree(A, lower, mid);
23      T2 = arrayToAVLtree(A, mid + 1, upper);
24      return (T1, A[mid], T2, max{height(T1)+1, height(T2)+1});
25    end if;
26  end arrayToAVLtree.
```

Um aus einem Array *A* der Länge *n* einen AVL-Baum *T* zu erzeugen, lautet der Aufruf *T* := *arrayToAVLtree*(*A*, 1, *n* + 1).

17 Punkte

(c) Beweisen Sie Ihre Aussage zur Laufzeit des Algorithmus *arrayToAVLtree* aus Teilaufgabe (b).

Hinweise:

1. Anstatt alle natürlichen Zahlen für *n* zu betrachten, genügt es in Teilaufgabe (c), eine geeignete unendliche Teilfolge der natürlichen Zahlen auszuwählen.
2. Es gilt:

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

Aufgabe 3 Sortierverfahren *25 Punkte*

(a) Geben Sie die Laufzeit von Quicksort (mit *findx* für die Bestimmung des Pivotelements), Heapsort und Bubblesort, für den Fall an, dass die übergebene Folge bereits sortiert ist. Begründen Sie die von Ihnen angegebene Laufzeit. Welches der drei Verfahren ist demnach unter der angegebenen Bedingung optimal? *7 Punkte*

(b) Die Zahlenfolge 15 - 96 - 47 - 32 - 43 - 18 - 5 - 68 - 103 - 55 soll mittels Heapsort *aufsteigend* und *in situ* sortiert werden. Geben Sie das Array nach dem Aufbau des initialen Heaps sowie nach der Entnahme und Bearbeitung jedes Elements an. *9 Punkte*

(c) Normale deutsche KFZ-Kennzeichen bestehen aus 3 Komponenten. Die erste Komponente ist eine ein- bis dreielementige Buchstabenfolge für die Zulassungstelle. Die zweite Komponente besteht aus einem oder zwei Buchstaben. Die dritte Komponente ist eine ein- bis vierstellige Zahl. Sortieren Sie die unten angegebene Folge von KFZ-Kennzeichen entsprechend ihrer Komponenten mittels Radixsort: *7 Punkte*

HA - TO 1234, DO - TO 432, HA - T 1234, DO - AJ 432,

HH - KO 123, HH - TO 1234, BO - AJ 123

Geben Sie für jede Phase an, wonach sortiert wird (Zulassungsstelle, Buchstabenkombination, Zahl). Es reicht aus, wenn jeweils Sie die gefüllten Behälter mit ihren Inhalten angeben.

(d) Wie verändert sich die Laufzeit von Radixsort, wenn die zu sortierenden Werte schon sortiert vorliegen? Begründen Sie Ihre Antwort. *2 Punkte*

16 Punkte **Aufgabe 4 B-Baum**

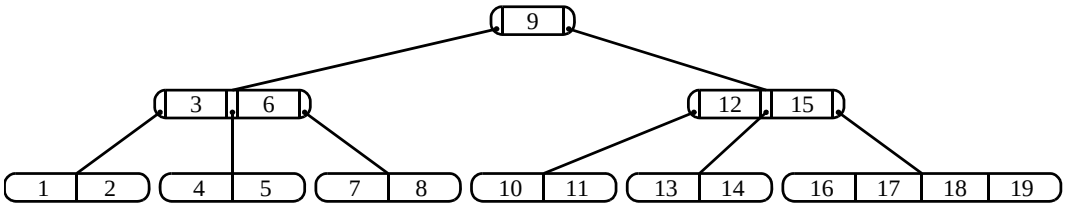
8 Punkt (a) Fügen Sie in einen anfangs leeren B-Baum der Ordnung zwei die folgenden Elemente in der gegebenen Reihenfolge ein. Zeichnen Sie den Baum jeweils vor und nach einer Split-Operation.

15, 23, 12, 17, 19, 32, 25, 6, 9, 10, 35

8 Punkt (b) Löschen Sie aus dem unten stehenden B-Baum der Ordung zwei nacheinander die folgenden Elemente in der gegebenen Reihenfolge.

14, 4

Verwenden Sie bei einer Balance- oder Mergeoperation den linken Nachbarn wenn beide Nachbarn zum Ausgleich geeignet sind. Zeichnen Sie jeweils den Baum vor und nach einem Ausgleich und geben Sie die verwendete Ausgleichsoperation an.



1 Punkt **Aufgabe 5 Deckblatt**

Lesen Sie sich die „Hinweise zur Bearbeitung“ sorgfältig durch. Füllen Sie den dort aufgeführten Anweisungen entsprechend beide Deckblätter vollständig und korrekt aus.