Universität Mannheim
Fakultät für Mathematik und Informatik
Lehrstuhl für Praktische Informatik IV
Prof. Dr.-Ing. W. Effelsberg

Dezember 2008: Klausur Praktische Informatik I

Name: _____

Matrikel-Nr.: _____ Semester: _____

Studienfach: _____

Anweisungen:

1. Füllen Sie bitte sofort den Kopf des Deckblattes aus!
2. Unterschreiben Sie die Klausur auf der letzten Seite!
3. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit: **12** Seiten!
4. Tragen Sie die Lösungen — soweit möglich — direkt in die Klausur ein!
5. Zugelassene Hilfsmittel: nicht programmierbarer Taschenrechner
6. Bearbeitungszeit: 90 Minuten

Aufgabe	max. Punktzahl	Punkte
1	11	
2	20	
3	8	
4	20	
5	19	
6	12	
Summe	90	

Aufgabe 1 — 11 Punkte

Einführung

(a) [2 Punkte] Was lässt sich über die Attribute einer Klasse `A` und einer Klasse `B` sagen, wenn `A` von `B` erbt?

(b) [2 Punkte] Erklären Sie kurz, warum die Korrektheit eines Programmes in den meisten Fällen nicht durch Testen bewiesen werden kann.

(c) [2 Punkte] Gegeben seien zwei Variablen vom Typ `String` namens `s1` und `s2` in Java. Worin liegt der Unterschied zwischen den beiden Ausdrücken `s1 == s2` und `s1.equals(s2)`?

(d) [2 Punkte] Darf die `return`-Anweisung innerhalb einer Methode mit Ergebnistyp `void` verwendet werden? Falls ja, was bewirkt sie in diesem Zusammenhang?

(e) [3 Punkte] Rechnen Sie die Zahl $(1011100)_2$ in das Dezimalsystem und die Zahl $(116)_{10}$ in das Binärsystem um. Geben Sie in beiden Fällen Ihren Rechenweg an.

Aufgabe 2 — 20 Punkte

Algorithmenentwurf

In dieser Aufgabe sollen Algorithmen zu gegebenen Problemen entworfen werden. Verwenden Sie Pseudocode oder Java-Syntax zur Beschreibung Ihres Algorithmus. Beschreiben Sie Ihren Algorithmus knapp und präzise.

(a) [10 Punkte] Ein achsenparalleles Rechteck kann durch Angabe der Koordinaten seiner linken oberen sowie rechten unteren Ecke eindeutig bestimmt werden:

```
class Rechteck {
    double x1, y1; // linke obere Ecke
    double x2, y2; // rechte untere Ecke
}
```

Geben Sie einen Algorithmus an, der ermittelt, ob sich die Flächen zweier gegebener, achsenparalleler Rechtecke überschneiden, das heißt, ob es einen Punkt gibt, der in beiden Rechtecken liegt. Die Seiten und Eckpunkte eines Rechtecks zählen ebenfalls als im Rechteck liegend.

Gehen Sie wie folgt vor: Geben Sie ein Verfahren zum Prüfen auf Überschneidung zweier eindimensionaler Intervalle an. Verwenden Sie dieses, um Rechtecke bezüglich Überschneidung zu untersuchen.

(b) [10 Punkte] Gegeben sei ein beliebiger Text T und ein Wort W. Geben Sie einen Algorithmus an, der ermittelt, ob W in T enthalten ist.

Aufgabe 3 — 8 Punkte

Auflösung von Rekursion

(a) [8 Punkte] Gegeben sei die unten aufgeführte rekursive Java-Methode. Schreiben Sie die Methode so um, dass zwei geschachtelte Schleifen statt der Rekursion verwendet werden. Die Abbruchbedingungen und die veränderten Parameter der rekursiven Aufrufe sollen – ggf. in leicht modifizierter Form – in der neuen Methode wieder zu finden sein.

Ein Aufruf der Methode könnte so aussehen: `drawBlock(5, 5, 5);`

```
void drawBlock(int width, int height, int i) {
    if (height == 0)
        return;

    if (i > 0) {
        System.out.print("X");
        drawBlock(width, height, i-1);
    }
    else {
        System.out.println("X");
        drawBlock(width, height-1, width);
    }
}
```

Aufgabe 4 — 20 Punkte

Listen

Gegeben sei die im Folgenden aufgeführte Klasse `List`, die eine verkettete Liste repräsentiert. Ihr Attribut `value` speichert einen ganzzahligen Wert; das Attribut `next` speichert einen Verweis auf das nachfolgende Listenelement oder `null`, wenn es sich um das letzte Listenelement handelt. Für die Bearbeitung dieser Aufgabe darf zunächst vereinfachend angenommen werden, dass die Liste immer mindestens ein Element enthält.

```
class List {
    private Integer value;
    private List next;

    List(Integer value, List next) {
        this.value = value;
        this.next = next;
    }
}
```

(a) [5 Punkte] Implementieren Sie die Methode `void addFirst(Integer newValue)` für die Klasse `List` in Java. Sie soll den übergebenen ganzzahligen Wert an die erste Stelle der Liste einfügen. Die Methode wird auf dem ersten Listenelement aufgerufen.

(b) [5 Punkte] Implementieren Sie die Methode `Integer removeFirst()` für die Klasse `List` in Java. Sie soll das erste Element der Liste entfernen und den darin gespeicherten ganzzahligen Wert zurückgeben. Die Methode wird auf dem ersten Listenelement aufgerufen.

(c) [2 Punkte] Welcher Spezialfall der verketteten Liste wurde hier implementiert? Geben Sie den Namen der Datenstruktur an.

(d) [8 Punkte] Implementieren Sie die Methode `double average()` für die Klasse `List` in Java. Sie soll alle Elemente der Liste *iterativ* durchlaufen und den Durchschnitt aller darin gespeicherten Ganzzahlen als Gleitkommazahl berechnen. Die Methode wird auf dem ersten Listenelement aufgerufen.

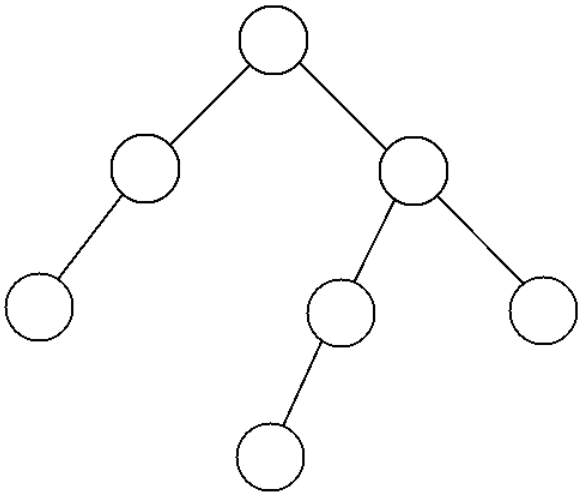
Berücksichtigen Sie diesmal auch den Fall, dass die Liste leer ist. Dies äußert sich durch eine `null`-Referenz als `value` des ersten Listenelements. Werfen Sie in diesem Fall eine geprüfte `Exception`.

Aufgabe 5 — 19 Punkte

Binärbäume

Die Zahlen $\{1, 2, 3, 4, 5, 6, 7\}$ sollen in einen sortierten Binärbaum eingefügt werden.

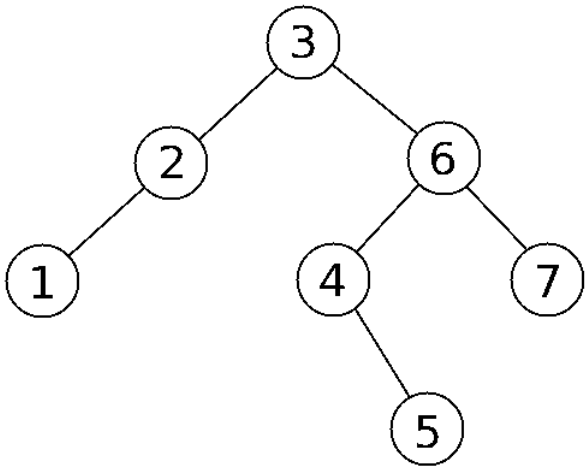
(a) [3 Punkte] Geben Sie eine Einfügereihenfolge der Zahlen an, so dass ein Binärbaum mit der hier abgebildeten Struktur entsteht.



(b) [2 Punkte] Geben Sie eine Einfügereihenfolge der Zahlen an, so dass ein Binärbaum mit minimaler Höhe entsteht.

(c) [2 Punkte] Geben Sie eine Einfügereihenfolge der Zahlen an, so dass ein Binärbaum mit maximaler Höhe entsteht.

(d) [6 Punkte] Gegeben sei der unten abgebildete sortierte Binärbaum. Entfernen Sie daraus nacheinander die Knoten $3, 6, 4$. Zeichnen Sie den veränderten Binärbaum nach jedem Löschvorgang auf.



(e) [6 Punkte] Gegeben sei die Klasse `BinTree`, die einen Binärbaum in Java repräsentiert.

```
class BinTree
{
    Integer item;
    BinTree left, right;
}
```

Alle Operationen der Klasse seien so implementiert, dass die Attribute in einem Knoten zu jeder Zeit entweder alle gleich `null` oder alle ungleich `null` sind. Implementieren Sie eine Methode `static void preOrderRek(BinTree tree)`, die den gesamten übergebenen Binärbaum in *preorder*-Reihenfolge durchläuft und die Schlüssel ausgibt. Die Methode soll rekursiv arbeiten.

Aufgabe 6 — 12 Punkte

Komplexität von Algorithmen

(a) [4 Punkte] Es treffen sich n Personen in einem Raum. Diese n Personen kennen sich alle untereinander und möchten sich nun gegenseitig persönlich mit Namen begrüßen, so dass jeder jeden anderen genau einmal begrüßt hat. Da die Gruppe sehr höflich ist, sprechen nie zwei Personen gleichzeitig. Nehmen Sie an, dass das Aussprechen eines persönlichen Grußes konstant viel Zeit in Anspruch nimmt. Geben Sie die Komplexität des Prozesses "Jeder grüßt jeden" in Abhängigkeit der Personenanzahl n in O-Notation an.

(b) [8 Punkte] Nehmen Sie an, Sie haben ein wichtiges Passwort "verlegt"... Nun möchten Sie durch Ausprobieren aller Möglichkeiten das Passwort herausfinden. Sie wissen, dass das Passwort n Kleinbuchstaben enthält. Das Eintippen und Ausprobieren eines Passwortes der Länge n benötigt $O(n)$ Zeit. Geben Sie die Komplexität des Problems des Herausfindens eines Passwortes in O-Notation an.