

Semesterklausur
**Einführung in die Informatik:
Programmierung und Software-Entwicklung**
Wintersemester 1999/2000

Gruppe A

Angaben zur Person:

Name, Vorname:

Matrikelnummer: Login:

Hauptfach:

Nicht von der Kandidatin bzw. vom Kandidaten auszufüllen

Bewertung:

Aufgabe	1	2	3	4	Σ
mögliche Punkte	6	6	10	8	30
erreichte Punkte					

Note:

Schein: ja: nein:

Organisatorische Hinweise

- Bitte zuerst das Deckblatt ausfüllen!

Stapel auf Vollständigkeit (zehn Blätter) und einwandfreies Druckbild überprüfen.

Studentenausweis und Lichtbildausweis zur Überprüfung bereitlegen.

Studenten mit Hauptfach Betriebswirtschaftslehre müssen die „Grüne Berechtigungskarte“ vorweisen.

Studenten mit Hauptfach Volkswirtschaftslehre müssen das Klausuranmeldeformular ausfüllen.

- Elektronische Hilfsmittel (Laptop, Taschenrechner, etc.) sind verboten.

Mitschriften und andere schriftliche Unterlagen sind als Hilfsmittel zulässig.

- Keine losen Blätter. Diesen Stapel zusammengeheftet lassen.

Als Lösung wird nur gewertet, was auf die Blätter dieses Stapels geschrieben ist.

Hinter jedem Aufgabentext ist genügend Platz für die Lösung. Wenn Sie mehr Platz benötigen, können Sie die Rückseiten und die leeren Zusatzblätter 7–10 verwenden. In diesen Fällen verweisen Sie bitte jeweils beim Aufgabentext deutlich auf die Stelle, an der die Lösung steht.

Bei Bedarf können Sie von der Aufsicht weitere leere Zusatzblätter *einheften* lassen.

- Bearbeitungszeit: 120 Minuten.

Wir wünschen Ihnen viel Erfolg bei der Klausur!

Aufgabe 1 Zweierkomplementdarstellung

(6 = 2 + 4 Punkte)

1. Stellen Sie folgende vierstellige Zweierkomplementzahlen als Dezimalzahlen dar:
 $(1110)_2$ und $(0110)_2$.
2. Ganze Zahlen in Zweierkomplementdarstellung kann man in Java als ein Feld von Booleschen Werten repräsentieren. Dabei wird folgende Konvention getroffen: n -stellige Zweierkomplementzahlen $(b_1 \dots b_n)_2$ werden durch Felder der Länge n repräsentiert, sodaß im i -ten Feldeintrag die $(i + 1)$ -te Ziffer von links b_{i+1} steht ($0 \leq i \leq n - 1$). So wird etwa $-4 = (1100)_2$ repräsentiert durch

```
boolean[] minusfour = { true, true, false, false };
```

Implementieren Sie eine Java-Funktion

```
public static int twocompval(boolean[] c)
```

die den Wert einer Zweierkomplementzahl c berechnet.

Aufgabe 2 BNF-Grammatik

(6 = 4 + 2 Punkte)

Beispiele für die Deklaration zweidimensionaler `boolean`-Feldes mit Initialisierung in Java — so wie aus der Vorlesung bekannt — sind etwa:

```
boolean[] [] x={{true,true,true},{true,false}};  
boolean[] [] y={{true,false},{true,true,false}};  
boolean[] [] xy={{true},{},{false}};
```

Die Feldinitialisierung kann in beiden Felddimensionen beliebig viele Elemente enthalten. Wir nehmen an, daß der Name des Feldes nur aus Kleinbuchstaben („a–z“) besteht.

1. Geben Sie eine BNF-Grammatik für solche Java-Deklarationen von zweidimensionalen `boolean`-Feldes mit Initialisierung an, wobei `boolean[] []`, `=`, `{`, `}`, `;`, `,`, `"a"`, `...`, `"z"`, `true` und `false` die Terminalsymbole der Grammatik sind. Zwischen den Terminalsymbolen sind keine Leerzeichen vorzusehen.
2. Geben Sie eine Ableitung an für: `boolean[] [] xy={{true},{}}`;

Aufgabe 3 Objektorientierung

(10 = 3 + 4 + 3 Punkte)

Eine Firma hat mehrere Warenlager, aber mindestens eines. Die Warenlager halten jeweils mehrere Produkte vorrätig, aber mindestens eines. Produkte haben einen Namen, eine Nummer und einen Preis. Jacken und Hosen sind Produkte. Jacken und Hosen haben eine Größe.

Erstellen Sie eine Java-Implementierung für diese Situation, die den folgenden Anforderungen genügt:

- Die Implementierung stellt eine geeignete Klassenhierarchie zur Verfügung. Firmen haben ein Attribut mit der Anzahl ihrer Warenlager. Warenlager haben ein Attribut mit der Anzahl der in ihnen geführten Produkte. Produktnamen werden durch den Datentyp **String** repräsentiert, Produktnummern durch **int**, Preise durch **int** und Größen durch **char**.
Es müssen keine Konstruktoren angegeben werden.
- Es gibt eine Instanzmethode, die es ermöglicht, zu einer Produktnummer in einem Lager den Preis zu ermitteln. Die Methode soll -1 zurückliefern, wenn kein Produkt mit dieser Nummer im Lager vorhanden ist. Es darf davon ausgegangen werden, daß ein Lager keine zwei Produkte mit gleicher Nummer vorrätig hält.
- Es gibt eine Instanzmethode, die es ermöglicht, zu einer Produktnummer dasjenige Lager einer Firma zu finden, das ein Produkt mit dieser Nummer zum minimalen Preis vorrätig hat. Dabei darf davon ausgegangen werden, daß mindestens ein Lager der Firma ein Produkt mit dieser Nummer vorrätig hat.
- Die geforderten Methoden müssen in den passenden Klassen deklariert sein.

Aufgabe 4 Listen

(8 = 3 + 5 Punkte)

Listen aus ganzen Zahlen können in Java — wie aus der Übung bekannt — als Objektstrukturen aus Instanzen der folgenden Klasse repräsentiert werden:

```
class List {
    boolean isempty;
    int contents;
    List next;

    public List() {
        isempty = true;
    }

    public List(int c, List n) {
        contents = c;
        next = n;
        isempty = false;
    }
}
```

Die folgende iterative Methode der Klasse `List` spaltet die aktuelle Liste in zwei Teillisten auf: Die Methode liefert ein Feld der Länge zwei zurück. Im ersten Eintrag des zurückgegebenen Feldes steht die Teilliste aus genau all denjenigen Elementen der ursprünglichen Liste, die kleiner oder gleich dem Wert von `x` sind; im zweiten Eintrag des zurückgegebenen Feldes aber die Teilliste aus allen restlichen Elemente der ursprünglichen Liste. Dabei wird die Reihenfolge der Elemente erhalten werden. Es wird die Hilfsmethode `List append(List l)` aus den Übungen in der Klasse `List` vorausgesetzt, die die Liste `l` an die aktuelle Liste anhängt.

Die Methode `List[] split(int x)` liefert also zum Beispiel für die Liste `[3, 9, 7, 4, 12, 1, 27]` und für `x = 7` das Paar der Listen `[3, 7, 4, 1]` und `[9, 12, 27]` als Ergebnis.

```
List[] split(int x) {
    List[] ret = new List[2];
    ret[0] = new List();
    ret[1] = new List();

    int c = contents; List n = next; boolean e = isempty;
    for (; e == false; c = n.contents, e = n.isempty, n = n.next) {
        if (c <= x)
            ret[0] = ret[0].append(new List(c, new List()));
        else
            ret[1] = ret[1].append(new List(c, new List()));
    }
    return ret;
}
```

1. Geben Sie die Worst-case-Zeitkomplexität dieser Methode `List[] split(int x)` an, wobei die Länge der Liste, für die die Methode aufgerufen wird, als Komplexitätsparameter betrachtet wird. Die Methode `List append(List l)` hat die Worst-case-Zeitkomplexität $O(n)$, wobei n die Länge der Liste bezeichnet, für die `append` aufgerufen wird. Begründen Sie Ihre Antwort.
2. Implementieren Sie eine rekursive Methode `List[] splitrec(int x)` für die Klasse `List`, die das gleiche leistet wie `List[] split(int x)`. (*Hinweis*: Die Methode `append` muß nicht verwendet werden.)

Zusatzblatt

(Bitte angeben, welche Aufgabe Sie hier beantworten)

Zusatzblatt

(Bitte angeben, welche Aufgabe Sie hier beantworten)

Zusatzblatt

(Bitte angeben, welche Aufgabe Sie hier beantworten)

Zusatzblatt

(Bitte angeben, welche Aufgabe Sie hier beantworten)