

Name:

Vorname:

Matr.-Nr.:

Technische Universität München
Fakultät für Informatik
Prof. B. Brügge, Ph.D.

WS 2000/2001
10. Februar 2001

Klausur zu Einführung in die Informatik I

(Gruppe A)

Aufgabe 1 (2+2=4 Punkte) Signaturen und Terme

Gegeben ist folgende Signatur:

Sorten:	$S = \{\text{Container}, \text{Dose}, \text{bool}\}$
Operationssymbole:	$F = \{\text{leererContainer}, \text{wirfEin}, \text{leereAus}, \text{istVoll}, \text{einwurfErlaubt}\}$
Funktionalität:	$\text{leererContainer}: \rightarrow \text{Container}$ $\text{wirfEin}: \text{Container} \times \text{Dose} \rightarrow \text{Container}$ $\text{leereAus}: \text{Container} \rightarrow \text{Container}$ $\text{istVoll}: \text{Container} \rightarrow \text{bool}$ $\text{einwurfErlaubt}: \text{Container} \times \text{Dose} \rightarrow \text{bool}$

- a) Geben Sie einen Signaturgraphen für diese Signatur an.
- b) Welche der folgenden Ausdrücke sind syntaktisch korrekte Terme der Signatur (d_1, d_2 seien Elementaroperanden der Sorte Dose)? Geben Sie für die nicht syntaktisch korrekten Terme eine kurze Begründung, warum sie nicht syntaktisch korrekt sind.
- (i) $\text{wirfEin}(\text{leererContainer}(), \text{istVoll}(\text{leererContainer}()))$
 - (ii) $\text{einwurfErlaubt}(\text{wirfEin}(\text{leererContainer}(), d_1), d_2, \text{einwurfErlaubt}(\text{leererContainer}(), d_2))$
 - (iii) $\text{istVoll}(\text{leereAus}(\text{wirfEin}(\text{leererContainer}(), d_1)))$
 - (iv) $\text{leereAus}(\text{wirfEin}(\text{wirfEin}(d_1, \text{leererContainer}()), d_2))$

Aufgabe 2 (3 Punkte) Boolesche Funktionen

Zeigen oder widerlegen sie, dass folgender boolescher Ausdruck allgemeingültig, also eine Tautologie, ist: $\neg a \Rightarrow ((c \vee \neg d) \Rightarrow \neg(b \wedge a))$

Aufgabe 3 (4 Punkte) Markov-Algorithmus

Geben Sie einen Markov-Algorithmus an, der für Wörter über dem Zeichenvorrat $\{a, b\}$ mit dem Ergebnis true anhält, wenn sowohl die Anzahl an 'a' im Wort gerade ist, als auch die Anzahl der 'b' im Wort gerade ist. Andernfalls soll der Algorithmus nicht terminieren. Beispiel: für das Wort „baaaba“ terminiert er mit true, für das Wort „baaab“ nie.

Geben Sie zur besseren Verständlichkeit kurz Ihre Lösungsidee an.

Vorbemerkung zu den Aufgaben 4 und 5: Bei den Aufgaben 4 und 5 sind Problemstellungen im Zusammenhang mit Listen von Zahlen zu lösen. Es stehen auf diesen Listen folgende Methoden zur Verfügung:

```
IntSequenz create(); // erzeugt leere Sequenz
boolean isEmpty(IntSequenz a); // Ist die Sequenz leer ?
int first(IntSequenz a); // 1. Element der Sequenz
int last(IntSequenz a); // Letztes Element der Sequenz
IntSequenz rest(IntSequenz a); // Sequenz ohne 1.Element
IntSequenz lrest(IntSequenz a); // Sequenz ohne letztes Element
IntSequenz append(IntSequenz a, int el); // Haengt ein Element am Ende an
IntSequenz lappend(IntSequenz a, int el); // Haengt ein Element am Anfang an
int laenge(IntSequenz a); // Berechnet die Laenge der Sequenz
```

Aufgabe 4 (4 Punkte) Rekursive Programmierung

Gegeben sei eine beliebige Liste von Zahlen. Gesucht ist eine Funktion

```
IntSequenz verschiebeMax (IntSequenz a)
```

die das Maximum einer Liste ans Ende dieser Liste schiebt; ein Algorithmus dafür lautet:

- Ist die Liste leer oder besteht sie aus einem einzigen Element, so ist das Verfahren beendet und die Liste selbst ist das gesuchte Resultat.
- Ist das erste Element kleiner als das zweite, so entferne man das erste Element aus der Liste, ansonsten das zweite Element. Anschließend ermittle man das Resultat der Anwendung des Algorithmus auf die so verkürzte Liste, füge am Beginn dieses Resultates das zuvor entfernte Element ein und gebe die gesamte Liste als Resultat zurück!

Formulieren Sie in Java-Notation die Funktion `verschiebeMax`, die diesen Algorithmus implementiert. Es ist hier eine Implementierung in funktionalem Programmierstil verlangt; Einbettung der Problemstellung oder Hilfsfunktionen sollten nicht verwendet werden.

Aufgabe 5 (1+2+3+2=8 Punkte) Sortieren von Sequenzen von Zahlen

Zusätzlich zu den in obiger Vorbemerkung angegebenen Methoden verwenden wir die in Aufgabe 5 implementierte Methode `verschiebeMax`.

Sie können diese Methode auch dann verwenden, wenn Sie die vorige Aufgabe nicht gelöst haben. Es sei nun folgende Implementierung eines Sortierverfahren für Listen von Zahlen gegeben:

```
IntSequenz sortiere (IntSequenz a){
    return sort(verschiebeMax(a), create());
}
IntSequenz sort(IntSequenz a, IntSequenz b){
    return isEmpty(a) ? b
        : sort(verschiebeMax(lrest(a)), lappend(b, last(a)));
}
```

- a) Um welches, Ihnen bekannte, Sortierprinzip bzw. Sortierverfahren handelt es sich?
- b) Zeigen Sie, dass die Funktion `sort` terminiert.
- c) Das Sortierverfahren ist in repetitiver Form gegeben. Bringen Sie es unter Verwendung einer `while`-Schleife auf iterative Form! Formulieren Sie dazu eine Funktion `IntSequenz sortIterativ(IntSequenz a)`.

- d) Die Funktion `verschiebeMax` habe die Komplexität $O(n)$, wobei n die Länge der Liste ist; die in obigem Hinweis angegebenen Funktionen `create`, `isEmpty`, `last`, `lrest` und `lappend` besitzen die Komplexität $O(1)$. Geben Sie die Komplexität des Sortierverfahrens `sortiere` im schlechtesten Fall an und begründen Sie kurz Ihre Antwort!

Aufgabe 6 (4 Punkte) Reihungen

Ein Graph mit der Knotenmenge $V = \{0, 1, 2, \dots, n\}$ lässt sich durch eine zweidimensionale Reihung `boolean[][] g = new boolean[n+1][n+1]` darstellen; eine Kante (Verbindung) von Knoten i nach Knoten j wird repräsentiert durch den Reihungseintrag `g[i][j] = true`. Existiert keine Kante zwischen den Knoten, so ist der Eintrag `false`.

Geben Sie eine Java-Methode `boolean[][] weglänge2 (boolean[][] g)` an, die für eine als Parameter übergebene Graphendarstellung g als Ergebnis folgende Reihung zurückgibt: In ihr sind alle Wege der Länge 2 bezogen auf die Reihung g eingetragen, d.h. existiert in g ein Weg von Knoten i nach Knoten j über den Zwischenknoten k , so soll in der Ergebnisreihung ein Eintrag `true` von Knoten i nach Knoten j vorliegen.

Aufgabe 7 (5 Punkte) Binärbäume

Gegeben ist eine ausschnittsweise Klassendefinition für allgemeine (nicht sortierte) Binärbäume:

```
class Binaerbaum {
    private boolean leer;
    private int inhalt;
    private Binaerbaum linkesKind;
    private Binaerbaum rechtesKind;

    public Binaerbaum() {leer = true;}
    public Binaerbaum(int wert) {
        leer = false;
        inhalt = wert;
        linkesKind = new Binaerbaum();
        rechtesKind = new Binaerbaum();
    }
    //... Methoden fuer Einfuegen, Suchen, Loeschen
}
```

Erweitern Sie die Klasse `Binaerbaum` um eine rekursiv arbeitende Methode `int gibMaximum()`, die in einem Binärbaum nach dem größten Wert des Attributs `inhalt` sucht und diesen als Ergebnis zurückgibt. Die Methode setzt voraus, dass der Binärbaum nicht leer ist. Geben Sie kurz Ihre Lösungsidee an und eine Implementierung der Methode. Beachten Sie dabei, dass in allgemeinen Binärbäumen keine Sortierungseigenschaft vorliegt.

Hinweis: Innerhalb einer Klassendefinition kann auf alle Attribute (auch anderer Instanzen der Klasse) zugegriffen werden. `get`- und `set`-Methoden werden somit nicht benötigt.

Aufgabe 8 (3+1=4 Punkte) Modellierung

Eine Prüfung kann entweder mündlich oder schriftlich abgelegt werden oder als mehrteilige Prüfung stattfinden. Jede Prüfung besitzt außerdem eine Note.

- Modellieren Sie diesen Sachverhalt in graphischer Darstellung und verwenden Sie dabei Ihre Kenntnisse über Entwurfsmuster.
- Geben Sie graphisch ein Instanzdiagramm der Prüfung „Einführung in die Informatik I“ an, die Sie gerade ablegen.

Aufgabe 9 (4 Punkte) Referenzgeflecht/Instanzdiagramm

In dieser Aufgabe betrachten wir die Klasse `Person` und Listen von Personen. Die `Person` habe die Attribute `name` und `adressen`. Hierbei enthält `name` den Nachnamen der `Person`. Das Attribut `adressen` bildet das Adressbuch der jeweiligen `Person`. Es seien folgende Klassen gegeben:

```
class Liste {
    ListElement listenAnfang;

    Liste() {}

    void setListenAnfang(ListElement e) {listenAnfang = e;}
}

class ListElement {
    Person inhalt;
    ListElement nachfolger;

    ListElement (Person p) {
        inhalt = p;
        nachfolger = null;
    }
}

class Person {
    String name;
    Liste adressen;

    Person (String s){
        name = s;
        adressen = new Liste();
    }

    void setAddressEintrag(Person p){
        adressen.setListenAnfang(new ListElement(p));
    }
}
```

Gegeben ist folgender Programmausschnitt:

```
Person p1 = new Person ("Maier");
Person p2 = new Person ("Schmidt");
p2.setAddressEintrag(p1);
p1.setAddressEintrag(p2);
```

Geben Sie das Referenzgeflecht an, das bei Ausführung des angegebenen Programmausschnittes entsteht. (Verlangt ist eine graphische Darstellung des Referenzgeflechtes in Form eines Instanzdiagramms, wie sie aus der Vorlesung und Übung bekannt sind.)