

Name:

Vorname:

Matr.-Nr.:

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

20. April 2001

Klausur zu Einführung in die Informatik I

(Gruppe B)

Aufgabe 1 (1.5+ 1.5 + 2 = 5 Punkte) Signaturen und Terme

Gegeben ist folgende Algebra (Signatur und Gesetze):

Sorten: $S = \{\mathbb{N}, \text{Team}\}$

Operationssymbole: $F = \{\text{leeresTeam}, \text{aufstellen}, \text{auswechseln}, \text{teamgroesse}\}$

Funktionalität: $\text{leeresTeam}: \rightarrow \text{Team}$

$\text{aufstellen}: \text{Team} \times \mathbb{N} \rightarrow \text{Team}$

$\text{auswechseln}: \mathbb{N} \times \mathbb{N} \times \text{Team} \rightarrow \text{Team}$

$\text{teamgroesse}: \text{Team} \rightarrow \mathbb{N}$

Gesetze: G1: $\text{auswechseln}(x, y, \text{aufstellen}(t, x)) = \text{aufstellen}(t, y)$

G2: $\text{auswechseln}(x, y, \text{aufstellen}(t, z)) = \text{aufstellen}(\text{auswechseln}(x, y, t), z)$

G3: $\text{aufstellen}(\text{aufstellen}(t, x), y) = \text{aufstellen}(\text{aufstellen}(t, y), x)$

- a) Geben Sie einen Signaturgraphen für diese Signatur an.
- b) Welche der folgenden Ausdrücke sind syntaktisch korrekte Terme der Signatur? Geben Sie für nicht syntaktisch korrekte Terme eine kurze Begründung, warum sie nicht syntaktisch korrekt sind.
- (i) $\text{aufstellen}(\text{auswechseln}(1, 2, \text{leeresTeam}()), \text{leeresTeam}())$
 - (ii) $\text{auswechseln}(\text{teamgroesse}(\text{leeresTeam}()), 1, \text{aufstellen}(\text{leeresTeam}(), 2))$
 - (iii) $\text{teamgroesse}(\text{auswechseln}(\text{teamgroesse}(\text{leeresTeam}()), \text{leeresTeam}(), 2))$
- c) Geben Sie unter Verwendung der Gesetze eine Ableitung für den Term
- $$\text{auswechseln}(1, 3, \text{aufstellen}(\text{aufstellen}(\text{leeresTeam}(), 1), 2))$$
- an, die am Ende die Operation `auswechseln` nicht mehr enthält. Hinweis: Dazu müssen Sie die Gesetze nur einseitig (linke durch rechte Seite ersetzen) verwenden.

Aufgabe 2 (4 Punkte) Markov-Algorithmus

Geben Sie einen Markov-Algorithmus an, der für Wörter über dem Zeichenvorrat $\{a, b, c\}$ mit dem Ergebnis `true` anhält, wenn das Wort gleichviele 'c' enthält, wie 'a' und 'b' zusammengekommen (es soll also gelten: $\text{Anzahl}(c) = \text{Anzahl}(a) + \text{Anzahl}(b)$). Andernfalls soll der Algorithmus nicht terminieren. Beispiel: für das Wort „bccacb“ terminiert er mit `true`, für das Wort „bccacbb“ nie.

Geben Sie zur besseren Verständlichkeit kurz Ihre Lösungsidee an.

Vorbemerkung zu den Aufgaben 3 bis 6: Bei den Aufgaben 3, 4, 5 und 6 sind Problemstellungen im Zusammenhang mit Sequenzen von Zahlen zu lösen. Es stehen auf diesen Sequenzen folgende Operationen 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 3 (3 Punkte) Rekursive Programmierung

Gegeben sei eine beliebige Sequenz von Zeichen. Gesucht ist eine Funktion

```
int anzahl (IntSequenz s, int el)
```

die als Ergebnis angibt, wie oft in einer Sequenz s eine als Parameter übergebene Zahl el enthalten ist.

Beispielsweise würde gelten:

anzahl(append(append(create(),1),1), 1) liefert als Ergebnis 2

anzahl(append(append(create(),1),1), 2) liefert als Ergebnis 0

Es ist hier eine Implementierung in funktionalem Programmierstil verlangt; Einbettung der Problemstellung oder Hilfsfunktionen sollten nicht verwendet werden.

Aufgabe 4 (4 Punkte) Bearbeitung von Zeichensequenzen

Zusätzlich zu den in obiger Vorbemerkung angegebenen Operationen verwenden wir die in Aufgabe 3 implementierte Operation anzahl .

Sie können diese Operation auch dann verwenden, wenn Sie die vorige Aufgabe nicht gelöst haben.

Es sei nun folgende Implementierung eines Verfahrens zur Entfernung mehrfacher Vorkommen aus einer Sequenz von Zahlen gegeben:

```
IntSequenz entferneMehrfache (IntSequenz s){
    return uebertrageZahlen(create(), s);
}

IntSequenz uebertrageZahlen(IntSequenz a, IntSequenz b){
    return isEmpty(b) ? a
        : (anzahl(a,last(b)) > 0) ?
        uebertrageZahlen(a, lrest(b))
        : uebertrageZahlen(lappend(a, last(b)), lrest(b));
}
```

Das Verfahren ist in repetitiver Form gegeben. Bringen Sie es unter Verwendung einer while-Schleife auf iterative Form! Formulieren Sie dazu eine Funktion

```
IntSequenz entferneMehrfacheIterativ(IntSequenz s).
```

Aufgabe 5 (4 Punkte) Strukturelle Induktion

Gegeben sei eine Sequenz s von Zahlen. Die Funktion f_{summe} berechnet die Summe aller Elemente der Sequenz; es gilt also:

$$f_{\text{summe}}(s) = \sum_{i=1}^{\text{laenge}(s)} s_i,$$

wobei s_i das i -te Element der Sequenz bezeichnet. Die Funktion f_{summe} sei durch die Operation `int summe(IntSequenz s)` folgendermaßen implementiert:

```
int summe(IntSequenz s) {
    return isEmpty(s) ? 0
        : summe(lrest(s)) + last(s);
}
```

Zeigen Sie durch strukturelle Induktion über Terme in Normalform, d.h. über Terme, die ausschließlich die Operationen `create` und `append` verwenden, dass die Operation `summe` partiell korrekt ist, d.h. dass sie die Funktion f_{summe} korrekt implementiert!

Aufgabe 6 (5 Punkte) Terminierung

Gegeben seien **sortierte** Sequenzen von Zahlen; hierbei sei das größte Element am Ende der Sequenz. Folgende Funktion ermittelt die Schnittmenge zweier Sequenzen:

```
IntSequenz schnittmenge(IntSequenz a, IntSequenz b){
    return (isEmpty(a) || isEmpty(b)) ? create()
        : last(a) == last(b)
            ? append(schnittmenge(lrest(a), lrest(b)), last(a))
            : last(a) < last(b)
                ? schnittmenge(a, lrest(b))
                : schnittmenge(lrest(a), b);
}
```

Zeigen Sie, dass die Funktion terminiert!

Aufgabe 7 (3 + 2 = 5 Punkte) Modellierung

Wir betrachten hier ein vereinfachtes Schema, das die Struktur von Büchern beschreibt: Ein Buch besteht aus mindestens einem Kapitel. Bücher und Kapitel sind Texte. Ein Text zeichnet sich durch eine bestimmte Länge (d.h. Anzahl von Zeichen) aus.

- Modellieren Sie diesen Sachverhalt in graphischer Darstellung und verwenden Sie dabei Generalisierung und Aggregation!
- Instanzieren Sie ein Buch „Einführung in die Informatik“! Es bestehe aus dem Kapitel „Funktionale Programmierung“ mit 333 Zeichen und dem Kapitel „Imperative Programmierung“ mit 555 Zeichen. Das Buch habe eine Länge von 1001 Zeichen.

Aufgabe 8 (3 + 2 = 5 Punkte) Referenzgeflecht

In dieser Aufgabe betrachten wir Listen von Zahlen. Es handelt sich hierbei um Listen, bei denen jedes Listenelement eine Referenz auf den Nachfolger enthält. Die Liste selbst enthält eine Referenz auf den Listenanfang und das Listeneende. Folgende Klassen sind gegeben:

```

class Liste {
    ListElement listenanfang;
    ListElement listenende;

    Liste() {listenanfang = null;
            listenende = null;
    }

    // ... Methoden zum Einfuegen, Loeschen, etc.
}

class ListElement {
    int zahl;
    ListElement nachfolger;

    ListElement(int z) {zahl = z;
                       nachfolger = null;
    }
}

```

Im Rahmen dieser Aufgabe kann auf die Attribute direkt zugegriffen werden, es müssen also keine get- und set-Methoden verwendet werden.

Folgender Programmausschnitt sei gegeben:

```

Liste l = new Liste();
l.listenanfang = new ListElement(1);
l.listenanfang.nachfolger = new ListElement(3);
l.listenanfang.nachfolger.nachfolger = new ListElement(4);
l.listenende = l.listenanfang.nachfolger.nachfolger;

```

- a) Geben Sie das Referenzgeflecht an, das bei der Ausführung des angegebenen Programmausschnittes entsteht!
- b) Zwischen dem Element mit der Zahl 1 und dem Element mit der Zahl 3 soll ein Element mit der Zahl 2 eingefügt werden. Geben Sie einen entsprechenden Programmausschnitt an!

Aufgabe 9 (5 Punkte) Suchen in Reihungen

Ein binärer Suchbaum (sortierter Binärbaum), wie er aus Vorlesungen und Übungen bekannt ist, kann auch in einer Reihung r repräsentiert werden: Die Wurzel des Baumes wird in $r[1]$ gespeichert (zur Vereinfachung lassen wir $r[0]$ unbesetzt). Wird ein Baumknoten in $r[i]$ abgespeichert, so findet sich das linke Kind unter $r[2*i]$ und das rechte Kind unter $r[2*i+1]$.

Geben Sie eine Java-Methode `int suchen (int[] r, int suchelement)` an, die in einer als Parameter übergebenen Darstellung eines sortierten Binärbaums als Reihung r (gemäß obiger Beschreibung) effizient (mit logarithmischem Aufwand) nach einem als Parameter übergebenen Element `suchelement` sucht: Als Ergebnis soll die Indexposition der Reihung, an der das Element gefunden wurde, zurückgegeben werden; ist das Element nicht enthalten, so soll 0 das Ergebnis sein.