Lehrstuhl für Praktische Informatik IV
Holger Füßler
A5, 6, Raum B 219
68131 Mannheim
Telefon: (0621) 181–2605
Email: fuessler@informatik.uni-mannheim.de

Robert Schiele
B6, 29, Raum C0.04
68131 Mannheim
Telefon: (0621) 181–2214
Email: rschiele@uni-mannheim.de

UNIVERSITÄT MANNHEIM

Praktische Informatik I — 12. Übungsblatt
Wintersemester 2005/2006 — Abgabe: 1. Februar 2006

Allgemeine Bearbeitungshinweise

Verwenden Sie für die Programmieraufgaben die in den Aufgaben angegebenen Klassennamen und geben Sie den Klassen-Quellcode sowohl auf Papier als auch per e-mail bei Ihrem Tutor ab. Falls nicht anders gefordert, lagern Sie sinnvoll Funktionalität in Methoden aus. Verwenden Sie Javadoc-Kommentare zur Dokumentation.
Aufgaben, bei denen die Punktzahl mit einem $*$ gekennzeichnet ist, zählen als Zusatzaufgaben zum Punktekonto, d.h. sie werden für die Berechnung der Zulassungsvoraussetzung im Zähler gewertet, aber nicht im Nenner.

Aufgabe 1 — 7 Punkte

Installieren Sie sich für die folgende Aufgabe eine Scheme-Umgebung auf Ihrem Rechner. Scheme-Umgebungen finden Sie unter anderem bei `http://www.gnu.org/software/guile/` oder für den Anfänger vielleicht leichter verständlich unter `http://www.plt-scheme.org/software/drscheme/`.

Aufgabe 1 a) — 3 Punkte

Implementieren Sie in Scheme eine Prozedur `fib`, welche die Fibonacci-Zahl (siehe Übungsblatt 7) einer gegebenen Zahl berechnet.

Aufgabe 1 b) — 4 Punkte

Im Folgenden soll die Ableitung des Ausdrucks `(fib 4)` genauer betrachtet werden. Benutzen Sie hier möglichst nicht die automatische Ableitungsfunktion der Scheme-Umgebung, denn obwohl dies bei der Korrektur natürlich nicht nachgeprüft werden kann, sollten Sie bedenken, dass Ihnen bei der Klausur auch kein Rechner zur Verfügung steht.
Warum ist eine Ableitung der Auswertung in rein applikativer Reihenfolge nicht möglich?
Warum ist eine Ableitung der Auswertung in rein „normaler“ Reihenfolge nicht möglich?

Leiten Sie den Ausdruck ab, indem Sie die beiden Ableitungsreihenfolgen so kombinieren, dass Sie zum Ziel kommen.

Aufgabe 2 — 4 Punkte

Schreiben Sie eine Prozedur, welche den Binomialkoeffizienten, wie auf dem vorigen Übungsblatt definiert, in Scheme implementiert.

Aufgabe 3 — 10 Punkte

Aufgabe 3 a) — 9 Punkte

Auf dem vorletzten Übungsblatt sollten Sie ein Programm, welches die Logelei vom ersten Übungsblatt löst, in Java implementieren. Versuchen Sie nun, eine entsprechende Implementation in Prolog zu schreiben. **(Geben Sie Ihren Quell-Code auch elektronisch ab.)**
Hinweise:

- Eine Prolog-Umgebung für die gebräuchlichen Systeme finden Sie unter `http://www.swi-prolog.org/`.
- Analog zur Java-Implementation erhalten Sie eine Infrastruktur zur Prüfung der Bedingungen bereits geliefert. Bauen Sie dazu die folgenden Prozeduren (deren Implementation Sie nicht unbedingt verstehen müssen) in Ihr Programm ein:

```
bei([A1|A0], A) :- A1 = A; bei(A0, A).
bei([A1|A0], A, [B1|B0], B) :- A1 = A, B1 = B; bei(A0, A, B0, B).
nach(A0, A, [_|B0], B) :- bei(A0, A, B0, B).
neben(A0, A, B0, B) :- nach(A0, A, B0, B); nach(B0, B, A0, A).
```

  Mit den Methoden `bei()`, `neben()` und `nach()` können Sie prüfen, ob zwei Eigenschaften im selben Haus, in benachbarten Häusern oder in benachbarten Häusern in bestimmter Reihenfolge auftreten. Zum Beispiel prüft die Klausel „`neben(E, pizza, T, voegel).`“, ob in dem Haus neben demjenigen der gern Pizza ißt, derjenige wohnt, der Vögel hält. `E` und `T` sind dabei die Listen, welche alle Gerichte, bzw. Tiere enthalten, also zum Beispiel „`E = [pizza, schokolade, fisch, bockwurst, pommes]`“.

- Sie können Ihr Programm derart aufbauen, dass Sie eine Prozedur `loesung` schreiben, welche folgendermaßen aufgebaut ist:

```
loesung([P1, F1, E1, D1, T1],
        [P2, F2, E2, D2, T2],
        [P3, F3, E3, D3, T3],
        [P4, F4, E4, D4, T4],
        [P5, F5, E5, D5, T5]) :-
        [P1, P2, P3, P4, P5] = P,
        [F1, F2, F3, F4, F5] = F,
        [E1, E2, E3, E4, E5] = E,
```

[D1, D2, D3, D4, D5] = D,
[T1, T2, T3, T4, T5] = T.

Damit haben Sie schon die Listen für die einzelnen Eigenschaften (`P` für Personen, `F` für Farben, `E` für Gerichte, `D` für Getränke und `T` für Tiere). Sie müssen hier nur noch dir Klauseln für die einzelnen Bedingungen des Problems hinzufügen. Aufrufen können Sie diese Prozedur dann mit „`loesung(H1, H2, H3, H4, H5)`“.

- In SWI-Prolog kann man mit der Prozedur `permutation(A, B)` prüfen, ob die Liste `A` eine Permutation der Liste `B` ist. Wenn Sie also zum Beispiel eine Permutation der Farben auf die Liste `F` zuordnen wollen, dann schreiben Sie „`permutation([gruen, blau, weiss, gelb, rot], F).`“.
- Wenn Sie Ihr Programm in eine Datei geschrieben haben, so können Sie diese in Prolog mit „`[dateiname].`“ laden.

Aufgabe 3 b) — 1 Punkte

Besitzt das Prolog-Programm auch das Problem, welches Sie bei der einfachen Version des Java-Programms entdeckt haben?

Aufgabe 4 — *19 Punkte

Aufgabe 4 a) — 17 Punkte

Schreiben Sie ein Programm in Java, das eine Liste von Zeichenketten (eine Zeichenkette pro Zeile, jede kann aus `a-zA-Z` bestehen) aus einer Datei einliest, intern in eine geeignete Datenstruktur speichert, diese aufsteigend sortiert, und dann auf der Konsole ausgibt. Verwenden Sie zur Sortierung und zum Vergleich der einzelnen Zeichenketten keine Bibliotheksfunktionen. Der Name der Datei soll dem Programm als Argument übergeben werden.
Auf der Übungshomepage finden Sie drei Dateien mit zufälligen Zeichenketten in unsortiertem und sortiertem Zustand. Ihr Programm soll für die Eingabe-Dateien die selbe Sortierung erzeugen.

Aufgabe 4 b) — 1 Punkte

Welche Laufzeitkomplexität hat ihr Programm, wenn eine Zeichenkette höchsten 100 Zeichen hat? Begründen Sie.

Aufgabe 4 c) — 1 Punkte

Welche Laufzeitkomplexität hat ihr Programm, wenn eine Zeichenkette beliebig lang sein kann. Begründen Sie.