Fachbereich Mathematik und Informatik
Institut für Informatik

AG Datenbanken / Informationssysteme
Prof. Dr. H. Schweppe

Algorithmen und Programmierung I

WS 2005 / 2006

Übung 15 (Zur Klausurvorbereitung)

Hier der Vorspann zur Klausur, damit sie den nicht erst während der Klausur lesen müssen.

- Hilfsmittel sind nicht erlaubt, auch keine Rechner, PDAs, Mobiltelefone.
- Legen Sie Ihren Studienausweis und Personalausweis neben sich auf den Platz.
- Jeder Täuschungsversuch hat den sofortigen Ausschluss von der Klausur zur Folge.
- Fragen dürfen nur mit den Aufsicht führenden Personen geklärt werden.
- Diejenigen, die nicht Deutsch als Muttersprache haben, sollten sich vergewissern, ob sie die Aufgaben sprachlich verstehen und sich gegebenenfalls an die Aufsicht wenden.
- Für Multiple Choice Aufgaben gilt: es kann, falls nichts anderes gesagt wird, keine, eine oder mehre richtige Antworten geben. Eine falsche Antwort wird mit einem Punktabzug bewertet.
- Bei den Programmieraufgaben dürfen die in der Vorlesung behandelten Standardfunktionen verwendet werden, wenn nichts anderes angegeben ist.
- Unterschreiben Sie hier, wenn Sie einverstanden sind, dass ihr Ergebnis unter Ihrer Matrikelnummer im Web (Zugang nur über VPN) veröffentlicht wird:

……

Die Klausur dürfte subjektiv etwas schwerer erscheinen, da es sich im Gegensatz zu vielen Aufgaben auf diesem Blatt, um bisher noch nicht behandelte Aufgaben handeln wird. In der Klausur werden manche Voraussetzungen (etwa das genaue Schema der primitiven Rekursion – A8) im Gegensatz zu diesem Blatt in der Aufgabenstellung angegeben.

Aufgabe 1

Geben Sie ein Beispiel mit einer Funktion **`foo a b`** und einer lokalen Definition an, bei der die Variable **`b`** durch die lokale Definition verdeckt wird.

Aufgabe 2

Definieren Sie eine Haskellfunktion **`allButOne`**, die die Elemente einer Liste darauf überprüft, ob genau ein Element einem Prädikat p nicht genügt.

Beispiel:
```
allButOne  prim [1,2,7,9,11] = True
allButOne (> 10) [ 7, 22,23,9,17] = False
```

Aufgabe 3 (P)

Gegeben eine Liste **`xs`** von Gleitkommazahlen. Gesucht ist eine Funktion

```
median ::[Float] -> Float ,
```

die den Median von xs bestimmt. Der Median einer Liste von verschiedenenZahlen **xs** ist die Zahl **x**, für die gilt: die Hälfte der Zahlen aus **xs** ist kleiner, die Hälfte ist größer als **x**.
Beachte: der Median von **xs** ist nicht notwendig Element von **xs**.

Es wird nicht vorausgesetzt, dass die Elemente von **xs** sortiert sind. Verwenden Sie dazu gegebenenfalls die Funktion **qsort' :: [Float]-> [Float]**, die den Quicksort.
Ergänzen Sie folgende Implementierung

```
qsort' [] = []
qsort' (a:rest) =   …  [x| x <-rest x<=a] ++  …       ++
```

Aufgabe 4
Beweisen Sie mit begründender Notation aller Teilschritte:
Für alle numerischen Listen vom Typ x**s : : [Num]** gilt:

```
sum (map (1+) xs) = length xs + sum xs
```

Schreiben Sie die definierenden Gleichungen der verwendeten Funktionen (**sum** etc) auf, nummerieren Sie die Gleichungen und beziehen Sie sich in Ihrem Beweis darauf.

Aufgabe 5

Definieren Sie eine Funktion **join,** die das Folgende leistet.

Argumente sind Listen, deren Basistyp Paare vom Typ **(a,b)** bzw **(b,c)** sind. a,b und c gehören zur Typklasse **Ord**. Der Wert ist eine Liste mit Basistyp **(a,c)**. **(x,y)** ist Element des Wertes von **join xs ys**, wenn es Elemente **(x,a)** in **xs** und **(a',y)** in **ys** gibt und **a==a'**.

Aufgabe 6

Geben Sie Folgen der Zahlen 1 bis 16 an, so dass bei Einfügen der Zahlen entsprechend der jeweiligen Folge a) ein Baum der Höhe 4, b) der Höhe 8 und c) der Höhe 15 entsteht.

Aufgabe 7

Leiten Sie den Typ von

```
map . zip
```

durch Typ-Unifikation ab. Zur Erinnerung:
Schreiben Sie sich die Signaturen der beteiligten Funktionen so hin, dass es keine Konflikte bei den Bezeichnern der Typvariablen gibt und unifizieren Sie entsprechende Ausdrücke.

Aufgabe 8

Definieren Sie die Funktion **plus** mit Hilfe des Schemas der primitiven Rekursion

Aufgabe 9

Was versteht man unter:

a) Generizität

b) Definieren Sie die Syntax arithmetischer Ausdrücke mit zweistelligen Operationen +,*,- . Operatorpräferenzen sollen nur durch Klammerung ausgedrückt werden.
Der Operandentyp <zahl> muss nicht definiert werden.
Beispiel eines Ausdrucks: (3+5)-(2*(3-5))

c) Was versteht man unter verzögerter Auswertung?

d) Was sind Lambda-Ausdrücke in Haskell?

e) Geben Sie je ein Beispiel einer assoziativen, rechts- und linksassoziativen Operation an.

f) Definieren Sie: linear rekursiv, nicht linear rekursiv, endrekursiv, primitiv rekursiv.

Aufgabe 10

Definieren Sie die Fibonacci-Zahlen als endrekursive Funktion.

Aufgabe 11

Definieren Sie die Funktion **`hanoi n`**, die für das Problem der Türme von Hanoi (wenn Sie nicht wissen, was das ist: in den VL-Unterlagen oder im Netz nachschauen!) und geben Sie die Rekursionsidee als kurzen Text wieder. (Es bringt rein gar nichts, die Definition und die Idee irgendwo abzuschreiben oder einfach nachzulesen ohne sich ernsthaft um die Lösung zu bemühen. Das gilt übrigens für alle Aufgaben.)