Fachbereich Mathematik und Informatik
Institut für Informatik

AG Datenbanken / Informationssysteme
Prof. Dr. H. Schweppe

Algorithmen und Programmierung I

WS 2004 / 2005 15.12. 2004

1. Klausur / Lösungen

Name,	Vorname,	Matrikelnr

1.1	1.2a	1.2b	1.2c	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10		
6	2	4	4	10	8	10	12	10	10	12	12		Σ

- Jede(r) in die Übungen der VL eingeschriebene Teilnehmer(in) ist teilnahmeberechtigt.
- Dauer: 16:00 – 18:00 .
- Studien- und Lichtbildausweis sichtbar auf den Platz legen.
- Keine Unterlagen, keine elektronischen Geräte (wie Notebook, Mobiltelefon) zugelassen!
- Die Klausur umfasst 10 Seiten, davon 1 weißes.
- Prüfen, ob Sie ein vollständiges Exemplar haben.
- Kein eigenes Papier verwenden, Sie können ggf. weitere Blätter erhalten.
- Die Zeitangaben sind grobe Richtwerte mit einer Summe kleiner als 120 Minuten.
- Abschreiben führt zu sofortigem Ausschluss aller daran Beteiligten (0 Punkte).

Ich bin einverstanden, dass mein Ergebnis unter meiner Matrikelnummer auf der Webseite von Alp1 und auf einem Listenaushang im Institut veröffentlicht wird. Die Ergebniswebseite kann nur innerhalb der FU eingesehen werden.

Alternative: keine Unterschrift und ab 4.1. 2005 Ergebnis im Sekretariat R 167 erfragen.

.....
Unterschrift

Aufgabe K1.1 (6 P, 4 Minuten)

a) Welchen Wert hat der Ausdruck `f [1,2,3,4]` mit

```
f xs = foldr einsPlus 0 xs
  where einsPlus x sum = 1 + sum
```

a `4` ← *r*
b `5`
c `10`
d Fehler, weil f nicht korrekt definiert ist.

b) Welche Eigenschaft müssen `op` bzw. `e` haben, damit gilt:

`(i) foldr op e = foldl op e` — *op assoziativ , e Einselement*

`(ii) foldr op e = foldr1 op` — *e Einselement oder erstes Element des nicht leeren Listenarguments von foldr1 op ist e*

Name,	Vorname,	Matrikelnr

Aufgabe K1.2 (10 Punkte, 10 Min.)

a) (2 P)
Welche Ausdrücke sind nicht korrekt und welchen Typ und Wert haben die korrekten?
(Kurze Begründung falls nicht korrekt, maximal ein Satz)

`quadrat (length "abc")` ← `Wert: 9 -- Typ: Int` mit `quadrat x = x*x`

`quadrat length []` *Falsch, quadrat wird auf length angewendet*

`quadrat . length []` *Falsch, quadrat . (length []) falscher Argumenttyp .*

`(quadrat . length) ["abcd"]` ← `Wert 1, natürlich auch Type Int`

b) (4 P)
Welche zwei der folgenden Ausdrücke sind typ- und wertgleich? Begründen Sie, warum die beiden anderen nicht äquivalent zu den Erstgenannten sind.

(i) `f(4 (False 'z'))` nicht äquiv. zu (ii) wg. Rechtsklammerung der Funktionsapplikation

(ii) `(((f 4) False) 'z')` ← äquiv. zu (iii)

(iii) `f 4` False `'z'` ← äquivalent zu (ii)

(iv) `(f 4) (False 'z')` syntaktisch falsch: False ist keine Funktion (genauer: ist nullstellige Fkt.)

c) (4 P)
Welche zwei der folgenden Typen sind identisch? Begründen Sie bei den anderen beiden, warum sie nicht zu den erstgenannten äquivalent sind.

(i) `(a -> Char) -> (b -> Int)` Bildet Funktion in Funktion ab (Ein Argument)

(ii) `a -> (Char -> (b -> Int))` ← identisch zu (iv)

(iii) `a -> (Char -> b) -> Int` bildet Wert vom Typ a und Funktion (Char -> b) in Int ab (Zwei Argumente)

(iv) `a -> Char -> b -> Int` ← identisch zu (ii): Funktion mit 3 Argumenten

Name,	Vorname,	Matrikelnr

Aufgabe K1.3 (10 Punkte, 10 Min.)

Was versteht man unter

a) strikter Typisierung einer Programmiersprache?
Jeder Ausdruck der Sprache besitzt einen Typ.

b) Funktion höherer Ordnung?
Funktion, die als Argument mindestens eine Funktion besitzt.

c) einem Funktional?
Eine Funktion, die als Wert eine Funktion liefert.

d) dem Halteproblem?
Die Frage, ob es ein Programm gibt, das für jedes Programm und jede Eingabe entscheiden kann, ob das Programm terminiert. (Negativ zu beantworten)

e) einer endrekursiven Funktion?
Eine Funktion f, bei der der rekursive Aufruf bereits den Wert der aufrufenden Funktion bestimmt.

f) einer totalen, injektiven Funktion
Eine Funktion , die auf jedem Element des Argumentbereichs definiert ist und gilt : f(a) = f(b) $\Rightarrow$ a=b (Eindeutigkeit der Urbilder von Funktionswerten).

g) einer Variablen in einer *funktionalen* Programmiersprache?
Einen Bezeichner für einen Wert.

h) einem black box-Test?
Test, bei dem nur das Ein- und Ausgabeverhalten einer Funktion getestet wird. Der innere Aufbau des Programms spielt für den black box-Test keine Rolle.

i) dem Begriff "Offshoring"
Auslagern von Aufgaben, besonders in der Informationstechnik, in Billiglohnländer

j) dem Begriff "Outsourcing"
Auslagern von IT-Aufgaben hin zu darauf spezialisierte Unternehmen.

(jeweils maximal zwei Sätze. Stichworte reichen, wenn Sie die Fragen vollständig beantworten.)

Name,	Vorname,	Matrikelnr

Aufgabe **K1.4 (8 Punkte, 8 Min.)**

Gegeben ein Warenkorb als Liste von Tupeln `(bezeichnung, einzelpreis, anzahl)` mit dem Typ `(String, Float, Int)`. Berechnen Sie den Gesamtpreis mit Hilfe von `foldr` oder `foldl` .

```
> gesamtPreis :: [(String, Float, Int )] -> Float
> gesamtPreis xs = foldr op 0 xs
>   where  -- op :: (String,Float,Int) -> Float-> Float
>          op (a,b,c) res =  b*(fromIntegral c) + res
```

Korrekturhinweis:
Kein Abzug, wenn (String, Int, Int) als Basistyp oder fehlende Typwandlung mit `fromIntegral`*. Wenn nur primitive Rekursion ohne fold: 3 Punkte Abzug, wenn sonst korrekt.*

Aufgabe **K1.5 (10 Punkte, 10 Min.)**

a) Gesucht ist eine Funktion `tNr`, die eine Liste aller Telefonnummern einer Person ausgibt. Als erstes Argument erhält die Funktion `tNr` eine Liste, die Tupel enthält. An der ersten Stelle jedes Tupels steht immer der Name einer Person, an zweiter Stelle die Telefonnummer. Als zweites Argument erhält die Funktion `tNr` den Namen, der Person, deren Telefonnummern ausgegeben werden sollen. Es ergibt sich die folgende Signatur: `tNr :: [(String,Int)] -> String -> [Int]`.

b) Abstrahieren Sie die Funktion zu einer Funktion assoc. Einer ihrer Parameter ist eine Liste von Tupeln. Die Tupel sollen auf eine beliebige Eigenschaft überprüft werden können. Wenn die zutrifft gehört die zweite Komponente zur Ergebnisliste. (Der Tupeltyp muss nicht notwendig `(String, Int)` wie in a) sein).
Geben Sie die Signatur an, definieren Sie die Funktion und definieren Sie die Funktion `tNr` mit Hilfe von `assoc`.

```
> -- gleich mit Faltung
> tNr :: String -> [(String, Int)]  -> [Int]
> tNr x = foldr op []
>    where op (a,b) res = if a==x then b:res else res
```

```
> assoc :: (a -> Bool) -> [(a,b)] -> [b]
> assoc p  = foldr p' []
>   where p' (a,b) res = if p a then b:res else res
>         -- Das Prädikat p wird hier nur auf die erste Komponente
>         -- angewendet wie für so genannte Assoziationslisten
>         -- üblich. Lösungen, die p auf das Paar anwenden sind
>         -- selbstverständlich korrekt.

> tNr' x = assoc (==x)
```

Name,	Vorname,	MatrikelNr

Aufgabe K1.6 (12 Punkte, 12 Min.)

Definieren Sie eine Funktion **`replaceFirst`**, die das erste Auftreten eines Elements (erstes Argument) in einer Liste (drittes Argument) durch einen anderen Wert (2. Argument) ersetzt.
Beispiel: **`replaceFirst 3 2 [1, 4,3,6,5,3,1,8,3] = [1,4,2,6,5,3,1,8,3]`**
a) linearer rekursiv aber nicht endrekursiv
b) Wandeln Sie a) mit der Akkumulatortechnik in einen endrekursiven Algorithmus um
c) Definieren Sie die Funktion mit Hilfe von **`takeWhile`** und **`dropWhile`**.
Zur Erinnerung:
`takeWhile p xs` liefert das längste Anfangsstück **`ys`** von **`xs`**, bei dem **`p x`** für jedes Element **`x`** von **`ys`** gilt, **`dropWhile p xs`** die Restliste, also **`xs`** ohne das Anfangsstück **`takeWhile p xs`**.
d) Warum ist das schwierig und nicht ohne weiteres möglich, **`replaceFirst`** mit Hilfe von **`map`** zu formulieren?

a)

```
> replaceFirst:: Eq a => a -> a -> [a] -> [a]
> replaceFirst _ _ [] = []
> replaceFirst o n (x:xs)
>     | x == o     = (n:xs)
>     | otherwise  = x:replaceFirst o n xs
```

b)

```
> repF o n = repF'[] o n
>   where repF' res o n [] = res
>         repF' res o n (x:xs)
>              | o == x    = res ++[n] ++ xs
>              | otherwise = repF' (res++[x]) o n xs
```

c)

```
> repF2 o n xs = takeWhile (o/=) xs  ++ [n] ++ tail (dropWhile (o/=)
xs)
>
```

d) *Die Ersetzung wird nicht auf alle Listenelemente sondern nur auf ein Anfangsstück der Liste angewendet*.

Name,	Vorname,	Matrikelnr

Aufgabe K1.7 (10 Punkte, 10 Min.)

Beweisen Sie durch strukturelle Induktion über Listen:

```
length (xs ++ ys) = length xs + length ys
```

wobei:

```
length [] = 0                          (L1)
length (x:xs) = 1+length xs            (L2)

(++) [] ys = ys                        (++1)
(++) (x:xs) ys = x:((++) xs ys)        (++2)
```

Induktion über das erste Argument xs.
Induktionsanfang: xs = []

length ([] ++ ys)	(++1)
= length ys	(Arith)
= 0 + length ys	
= length [] + length ys	(L1) ok

Induktionsvoraussetzung.:
Behauptung gilt für alle Listen mit 0 <= length xs < n

Induktionsschluss:

length((x:xs) ++ ys)	(++2)
= length (x:(xs++ys))	(L2)
= 1+ length (xs++ys)	(Ind.vorauss.)
= 1+ (length xs + length ys)	(*)

length (x:xs) + length ys	(L2)
= (1+length xs) + length ys	Assoziativität von + ,
= 1+ (length xs + length ys)	(**)

(*) = (**) was zu zeigen war.

Korrekturhinweis: korrekter Induktionsanfang: 3 Punkte, korrekte Induktionsvoraussetzung und –Behauptung: 2 Punkte.

Name,	Vorname,	MatrikelNr

Aufgabe K1.8 (10 Punkte, 12 Min.)

Von der folgenden Funktion ist nicht bekannt, ob sie für jedes Argument terminiert. Sie liefert für ein beliebiges Argument den Wert 1, vorausgesetzt sie terminiert.

```
ulam n
    | n==1                  = 1
    | n `mod` 2 == 0        =  ulam (n `div` 2)
    | otherwise             =  ulam (3*n+1)
```

Gesucht ist eine Funktion **`ulMax :: Int -> Int`**, die für ein Argument **`n`** den maximalen Argumentwert liefert, der während der Berechung von **`ulam n`** auszuwerten ist.
Beispiel: Die Berechnung von **`ulam 3`** erfordert **`ulam n`** für die Werte **`n=10,5,16,8,4,2,1`** zu berechnen.
Hinweis: Definieren Sie zunächst eine Funktion, die die Liste *aller* Argumente liefert, die von **`ulam`** berechnet werden müssen, um den Wert von **`ulam n`** zu erhalten.

```
> ulmax n = maxL (ulam2 n)
>  where maxL xs = foldl1 max xs

> ulam2 n = ulam' [] n
> ulam' arg 1 = 1:arg
> ulam' arg n
>    |n `mod`2 == 0   = ulam' (n:arg) x
>    |otherwise       = ulam' (n:arg) y
>               where x = n `div`2
>                     y = 3*n+1
```

Name,	Vorname,	Matrikelnr

Aufgabe K1.9 (12 Punkte, 15 Min.)

Die Potenzmenge einer Menge M ist die Menge aller Teilmengen von M. Zu den Teilmengen jeder Menge gehört die leere Menge,
Eine Menge wird hier durch eine Liste ohne doppelte Elemente repräsentiert. Gesucht ist eine Funktion

```
pot :: [a] -> [[a]]
```

die die Potenzmenge des Arguments liefert.
Beispiel: `pot [1,2,3] = [[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]]`
Es gibt mehrere Möglichkeiten, die Potenzmenge zu bestimmen. Auf die Reihenfolge der Elemente der Potenzmenge soll es nicht ankommen, allerdings ist **`pot xs`** selbst eine Menge und darf deshalb keine doppelten Elemente enthalten.
Hinweis: Zur Lösung dieserAufgabe muss man eine gute Idee haben. Ich empfehle sie erst zu lösen, wenn am Schluss noch Zeit ist.

```
> pot :: [a] -> [[a]]

> pot [] = [[]]       -- die leere Menge ist Teilmenge jeder Menge
> pot (x:xs)      =   pot1 ++ map (x:) pot1
>      where pot1 = pot xs

>                     -- kennt man die Potenzmenge P(M) einer
>                     -- Menge M, so erhält man die Potenzmenge von
>                     -- M ∪{x},x∉M, dadurch, dass man sie mit der
>                     -- Menge vereinigt, die man durch Hinzufügen
>                     -- von x zu jedem Element von P(M) erhält.
```

Name,	Vorname,	Matrikelnr

Aufgabe K1.10 (12 Punkte, 12 Min.)

Eine Grammatik für Listen sei durch folgende BNF-Regeln beschrieben.
Achtung: es handelt sich nicht um Haskelllisten!

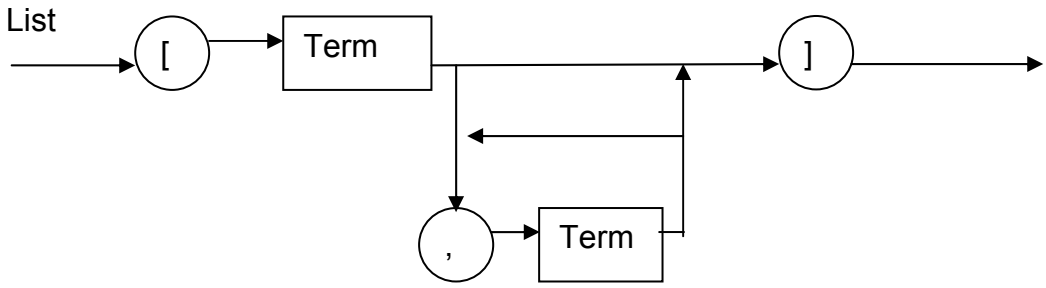
```
<Liste>    ::= '[' <Expr> ']'
<Expr>     ::= <Term> | <Term> ',' <Expr>
<Term>     ::= <Liste> | <Atom>
<Atom>     ::= 'a' | … | 'z'
```

a) Vereinfachen Sie die Grammatik zu einer EBNF-Grammatik (Syntax für Wiederholung etc. angeben)

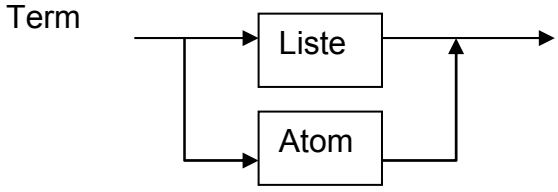
```
<liste>    ::= '[' <Term>{,Term}0..n ']'   -- {...}0..n
                                           -- n-malige Wiederholung n>=0

<Term>     ::= <Liste> | <Atom>
<Atom>     ::= 'a' | … | 'z'
```

b) Geben Sie die Syntaxdiagramme zu dieser Grammatik an



Atom: analog



c) Sind die folgenden Listen korrekt gemäß obiger Grammatik gebildet? Begründung falls nicht.

(i) `[a,[b],[[c]],[[[d]]]]` *ja*

(ii) `[a,[[[b,x],x],e],c,[]]` *nein: [] keine Liste erlaubt*

(iii) `[[a,x],b,[c,[d],[e,[f]]],h,[i]]` *ja*

Korrekturhinweis: a) 4,5 Punkte, b) 4,5 Punkte, c) je 1 Punkt

Zusatzblatt

Name,	Vorname,	Matrikelnr