FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 0

Präsenzaufgaben, keine Abgabe!

Aufgabe 0.1 Welche der folgenden Zeichenreihen stellen syntaktisch korrekte Haskell-Programme dar?

1. `X = [1,2,3]`
2. `f x = x + x * y`
3. `function f x = f x`
4.
```
n = a * b where a = 2
            b = 3
```

Aufgabe 0.2 Definieren Sie ohne Rückgriff auf die Haskell-Funktion prod eine Haskell-Funktion `produkt`, die für eine Liste von Zahlen das Produkt dieser Zahlen berechnet.
Beweisen Sie `produkt [x] = x`.

Aufgabe 0.3 In den Folien zur Vorlesung ist die Sortierfunktion `qsort` enthalten. Wie müsste die Definition von `qsort` modifiziert werden, damit eine Liste in der umgekehrten Reihenfolge sortiert wird?

Aufgabe 0.4 Schreiben Sie auf 2 verschiedene Weisen eine Haskell-Funktion `lastElem`, die angewandt auf eine nichtleere Liste von Elementen das letzte Element dieser Liste liefert. Sie dürfen hierzu nicht die Bibliotheksfunktion `last` verwenden.

Aufgabe 0.5 Welche Funktion wird hierdurch berechnet:

```
f n = sum [1, 3 .. (2*n-1)]
```

Aufgabe 0.6 Erstellen Sie mit einem Editor eine Datei, die eine Liste von Dezimalzahlen (Typ Float) enthält. Schreiben Sie dann eine Haskell-Funktion `einAusFloat`, die bei Eingabe des Dateinamens (als String) und einer weiteren Dezimalzahl die Liste der Dezimalzahlen zunächst einliest, die Dezimalzahl am Anfang der Liste einfügt und anschließend in der Datei mit Namen „erg" speichert.
Machen Sie dann das Entsprechende für eine Liste von ganzen Zahlen (Typ Int oder Integer) bzw. für eine Liste von Strings.

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 1

Abgabedatum: 26.10.2007 16.00 Uhr

Gemeinsame Abgaben von Gruppen bis zu 3 Personen sind möglich. Die Abgabe erfolgt per Email an den Übungsgruppenleiter.

Aufgabe 1.1 (15 Punkte)

Geben Sie den Typ der folgenden Werte an:

1. `['a','b','a']` (3 Punkte)
2. `(False, True, False)` (3 Punkte)
3. $([1,2],(\texttt{'a'},\texttt{'b'}))$ (4 Punkte)
4. `[tail, init, reverse]` (5 Punkte)

Aufgabe 1.2 (20 Punkte)

Geben Sie den Typ der folgenden Funktionen an:

1. `second xs = head (tail xs)` (5 Punkte)
2. `pair x y = (y,x)` (5 Punkte)
3. `palindrom xs = xs == reverse xs` (4 Punkte)
4. `twice f x = f (f x)` (6 Punkte)

Aufgabe 1.3 (35 Punkte)

1. Eine quadratische Gleichung $x^2 + ax + b = 0$ mit ganzzahligen Koeffizienten a und b kann durch das Tupel (a, b) mit Integer-Werten a und b repräsentiert werden.

   Schreiben Sie eine Haskell-Funktion `loesung,` die als Eingabe eine ganze Zahl n und ein Tupel (a, b) mit ganzen Zahlen a und b erhält und die testet, ob n Lösung der Gleichung $x^2+ax+b = 0$ ist. (10 Punkte)

2. Eine komplexe Zahl $a + i \cdot b,\ a, b \in \mathbb{R}$, kann als ein Tupel $(a, b) \in \mathbb{R}^2$ repräsentiert werden. In Haskell ist der Datentyp für ein Tupel reeller Zahlen `(Float, Float)`.

   Schreiben Sie für die komplexen Zahlen Haskell-Funktionen für Addition, Subtraktion, Multiplikation, und Division unter Verwendung dieser Repräsentation. (25 Punkte)

Aufgabe 1.4 (30 Punkte)

1. `&&` und || sind Haskell-Funktionen in Infix-Schreibweise vom Typ `Bool -> Bool -> Bool`, die 2 Wahrheitswerte unter dem logischen $\wedge$ bzw. dem logischen $\vee$ auswerten.

   Schreiben Sie eine Haskell-Funktion `sortiert :: [Integer] -> Bool`, die überprüft, ob eine Liste von ganzen Zahlen sortiert ist. (18 Punkte)

2. Schreiben Sie ein Haskell-Programm `pref`, das als Eingabe 2 Zeichenfolgen (vom Typ `[Char]`) xs und ys erhält und das True ausgibt, wenn die erste Zeichenfolge xs ein Präfix der zweiten Zeichenfolge ys ist. (12 Punkte)

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 2

Abgabedatum: 02.11.2007 16.00 Uhr

Aufgabe 2.1 (30 Punkte)

1. Schreiben Sie eine Haskell-Funktion `teilWort :: String -> String -> Bool`, die bei Eingabe zweier Strings x und y prüft, ob einer der beiden Strings im anderen als Teilwort vorkommt. (15 Punkte)

2. Ein Polynom $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ in x kann als Liste $[a_n, a_{n-1}, \ldots, a_1, a_0]$ vom Typ `[Float]` repräsentiert werden.

   Schreiben Sie eine Haskell-Funktion, die zu einem Polynom in x seine 1. Ableitung berechnet. (15 Punkte)

Aufgabe 2.2 (25 Punkte)

Ein beliebtes Kinderspiel zur Auswahl einer Kindes ist das folgende:

- Eine Gruppe von n Kindern bildet einen geschlossenen Kreis.
- Mit einem ersten Kind beginnend wird jeweils bis zum k-ten Kind weitergezählt. Dieses Kind scheidet aus, der Kreis wird wieder geschlossen.
- Vom Nachbarn (Nachfolger) des ausgeschiedenen Kindes beginnend verfährt man wie zuvor, so lange bis, nur noch 1 Kind übrigbleibt.

Schreiben Sie eine Haskell-Funktion, die bei Eingabe einer Liste von Kindern (z.B. mit den Nummern $1, \ldots, n$) und einer positiven ganzen Zahl k errechnet, welches Kind am Ende übrigbleibt. (Das Ergebnis darf auch in Form einer Einerliste vorliegen.)

Aufgabe 2.3 (20 Punkte)

Neben dem einfachen arithmetischen Mittel existieren eine Reihe weiterer Mittelwert-Funktionen. Für Werte $v_1, \ldots, v_n$ und einen Gewichtsvektor $w = (w_1, \ldots, w_n)$ ist das *gewichtete arithmetische Mittel* definiert als

$$\overline{v} = \frac{\sum_{i=1}^{n} w_i v_i}{\sum_{i=1}^{n} w_i}$$

1. Geben Sie den Typ der Haskell-Funktion an, die den gewichteten arithmetischen Mittelwert berechnet. (5 Punkte)

2. Schreiben sie eine Haskell-Funktion, die für zwei Vektoren gleicher Länge das gewichtete arithmetische Mittel berechnet, wobei der erste Vektor den Gewichtsvektor bezeichnet. (15 Punkte)

Aufgabe 2.4 (25 Punkte)

Seien Gegenstände $g_1, \ldots, g_n$ gegeben, wobei g_i das Gewicht des i-ten Gegenstandes ist. Ein LKW kann mit einem maximalen Gewicht von G beladen werden. Die Beladung des LKW lässt sich über eine Liste $[b_1, \ldots, b_n]$ binärer Werte beschreiben, wobei $b_i = 1$ angibt, dass der i-te Gegenstand aufgeladen wird und für $b_i = 0$ der i-te Gegenstand nicht aufgeladen wird.

1. Schreiben sie eine Haskell-Funktion `gewicht`, die für eine Liste binärer Werte und eine gleich lange Liste von ganzzahligen Gewichten das Gesamtgewicht der Ladung berechnet. (10 Punkte)

2. Schreiben sie eine Haskell-Funktion `ladung`, die für die gegebenen Gewichte

   $$5600, 2500, 7800, 2900, 1300, 2700, 4400, 3200, 5900, 4300, 3700$$

   und ein maximales Gesamtgewicht von $G = 25000$ eine möglichst maximale Beladung des LKWs in Form einer Liste binärer Werte berechnet. (15 Punkte)

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 3

Abgabedatum: 09.11.2007 16.00 Uhr

Aufgabe 3.1 (20 Punkte)

1. Geben Sie eine Haskell-Funktion an, die in einem String jeden Kleinbuchstaben in den zugehörigen Großbuchstaben umwandelt. Verwenden Sie dazu die Haskell-Funktion `toUpper`, die diese Umwandlung für einzelne Zeichen leistet. (Sie müssen, um diese Funktion verwenden zu können, die Zeile

   import Char

   an den Anfang Ihres Haskell-Programms stellen.) (10 Punkte)

2. Schreiben Sie eine `Haskell`-Funktion `dual2int :: String -> Integer`, die zu einer in Form eines Strings vorliegenden Dualzahl die zugehörige ganze Zahl berechnet. (10 Punkte)

Aufgabe 3.2 (20 Punkte)
Es ist eine Liste $x = [x_0, x_1, x_2, \ldots, x_n]$ so in zwei Teilisten y und z zu zerlegen, dass y alle Elemente (in der entsprechenden Reihenfolge) enthält, die in x an den geraden Positionen vorkommen, während z die Elemente enthält, die an den ungeraden Positionen enthalten sind.
Schreiben Sie hierzu eine Haskell-Funktion `split2 :: [a] -> ([a],[a])`, die bei Eingabe der Liste x das Tupel (y, z) der Teillisten y und z als Ergebnis liefert.

Aufgabe 3.3 (40 Punkte)
Wir wollen ein Polynom $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ in einer Variablen x und mit Koeffizienten, deren Typen in der Klasse *Num* liegen, über eine Liste $[a_0, a_1, \ldots, a_{n-1}, a_n]$ repräsentieren.
Schreiben Sie Haskell-Funktionen zur Berechnung von

1. Summe (10 Punkte)
2. Differenz (10 Punkte)
3. und Produkt (20 Punkte)

von Polynomen in x, die in dieser Weise repräsentiert sind.

Aufgabe 3.4 (20 Punkte)
Geben Sie den Typ der curryfizierten Version von f an.

1. `f :: (a, b, a) -> b` (5 Punkte)
2. `f :: (a, b) -> (c,c)` (4 Punkte)
3. `f :: (a, a -> b, b -> c) -> a -> c` (5 Punkte)
4. `f :: (a, (a -> b, b -> c)) -> (a -> c)` (6 Punkte)

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz

Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 4

Abgabedatum: 16.11.2007 16.00 Uhr

Aufgabe 4.1 (30 Punkte)

1. Geben Sie eine Haskell-Funktion `split` vom Typ `Char -> String -> [String]` an, die angewandt auf ein Zeichen c und eine Zeichenkette s die Zeichenkette s an allen Vorkommnissen von c aufsplittet und die Liste der entstehenden Teilstrings zurückliefert. Das Trennzeichen c soll beim Aufsplitten "verschluckt" werden. Z.B. würde

   `split 'a' "Man kann es nicht glauben"`

   das Ergebnis ["M", "n k", "nn es nicht gl", "uben"] als Ergebnis liefern.

   (10 Punkte)

2. Schreiben sie eine Haskell-Funktion `join`, die ein Zeichen c und eine Liste von Zeichenketten $s_1, \ldots, s_n$ als Argumente erhält und den konkatenierten String $s_1 c s_2 \ldots c s_n$ zurückliefert.

   (10 Punkte)

3. Schreiben sie eine Haskell-Funktion `count`, die für zwei Strings s_1, s_2 zählt wie häufig s_1 in s_2 enthalten ist. (10 Punkte)

Aufgabe 4.2 (30 Punkte)

1. Eine *ideale Zahl* ist eine Zahl, die genauso groß ist, wie die Summe ihrer *echten* Teiler (d.h. aller Teiler die ungleich der Zahl selbst sind). So ist z.B. 6 ideal, da für die echten Teiler $1, 2$ und 3 gilt: $1 + 2 + 3 = 6$.

   Schreiben sie eine Haskell-Funktion `ideal`, die für eine positive ganze Zahl $n > 1$ die Liste aller idealen Zahlen aus $\{1, \ldots, n\}$ liefert. (15 Punkte)

2. In `Haskell98` gibt es den (Daten-) Typ `Ratio Integer` für die Menge der rationalen Zahlen.

   Schreiben Sie eine `Haskell`-Funktion, die bei Eingabe der positiven ganzen Zahl n alle positiven rationalen Zahlen, deren Nenner $\leq n$ und größer als der Zähler sind, ohne Wiederholungen ausgibt. (Z.B. dürfte also nicht 1%2 und 2%4 ausgegeben werden, da beide Darstellungen dieselbe rationale Zahl darstellen.) (10 Punkte)

3. Schreiben Sie eine `Haskell`-Funktion, die bei Eingabe der positiven ganzen Zahl n die Summe der in der 2. Teilaufgabe ausgegeben rationalen Zahlen berechnet. (5 Punkte)

Aufgabe 4.3 (20 Punkte)

Geben sie eine Haskell-Funktion an, die für eine Liste L und eine natürliche Zahl n die Liste aller Teillisten von L mit Kardinalität n berechnet.

Aufgabe 4.4 (20 Punkte)

1. Stellen sie die Funktion

   ```
   mult a b c = a * b * c
   ```

   als Aufruf verschachtelter Lambda-Ausdrücke dar. (4 Punkte)

2. Geben sie die Funktion `reverse` als Lambda-Ausruck an und wenden sie diesen Ausdruck mit Hilfe der Funktion `map` auf alle Strings einer gegebenen Liste an. (4 Punkte)

3. Schreiben Sie eine Haskell-Funktion `mapIt :: (a -> a) -> [a] -> [a]`, die angewandt auf eine Funktion f und eine Liste $[x_0, x_1, x_2, \ldots, x_n]$ das Ergebnis $[x_0, f(x_1), f^2(x_2), \ldots, f^n(x_n)]$ liefert.

   Hierbei ist $f^0(x) := x$ und $f^{n+1}(x) := f(f^n(x))$. (6 Punkte)

4. Geben Sie einen passenden Lambda-Ausdruck f an, so dass `map` angewandt auf f und eine Liste $[x_0, x_1, \ldots, x_n]$ von ganzen Zahlen die Liste $[x_0 \cdot 2^{x_0}, x_1 \cdot 2^{x_1}, x_2 \cdot 2^{x_2}, \ldots, x_n \cdot 2^{x_n}]$ als Ergebnis liefert. (6 Punkte)

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 5

Abgabedatum: 23.11.2007 16.00 Uhr

Aufgabe 5.1 (25 Punkte)

1. In einer Liste sollen alle Duplikate von Listenelementen entfernt werden. Schreiben Sie eine Haskell-Funktion, die dies leistet. (10 Punkte)

2. In dieser Aufgabe legen wir Listen von ganzen Zahlen zu Grunde. Unter einem *Segment* einer Liste l verstehen wir eine aufeinanderfolgende Folge von Listenelementen der Liste l und unter der *Segmentsumme* die Summe der Elemente des Segments. So ist z.B. $[-3, 5, -2, 1]$ ein Segment der Liste $[-1, 2, -3, 5, -2, 1, 3, -2, -2, -3, 6]$ mit Segmentsumme 1.

   Geben Sie eine Haskell-Funktion an, die zu einer Liste von ganzen Zahlen das (bzw. ein, falls es mehrere gibt,) Segment mit maximaler Segmentsumme bestimmt. (15 Punkte)

Aufgabe 5.2 (25 Punkte)

1. Geben Sie unter Verwendung von Listenkomprehension eine Haskell-Funktion an, die bei Eingabe einer ganzen Zahl $n > 0$ die Summe $1^2 + 2^2 + \ldots + n^2$ der ersten n Quadratahlen > 0 berechnet.

   (10 Punkte)

2. Schreiben Sie eine Haskell-Funktion `list23 :: [a] -> [a]`, die für eine Liste l als Eingabe die Liste der Elemente ausgibt, die sich in l an Positionen befinden, die höchstens 2 oder 3 als Primfaktoren enthalten. Z.B. muss also gelten (man beachte, dass die Listenpositionen mit 0 beginnend nummeriert sind):

   ```
   list23 [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] = [1,2,3,4,6,8,9,12,16]
   ```

   (15 Punkte)

Aufgabe 5.3 (20 Punkte)

Schreiben Sie eine Haskell-Funktion, die zu einer Liste l die Liste aller Permutationen von l berechnet.

Aufgabe 5.4 (Stable Marriage Problem) (30 Punkte)

Betrachten Sie das folgende Problem der stabilen Heirat:
Gegeben sind n Männer $m_1, \ldots, m_n$ und n Frauen $f_1, \ldots, f_n$. Für jeden Mann existiert eine Liste, in der er alle Frauen nach absteigender Sympathie geordnet hat. Ebenso existiert für jede Frau eine Liste, in der sie alle Männer nach absteigender Sympathie gruppiert hat. Wir suchen nun eine stabile Zuordnung $[(m_1, f_{k_1}), \ldots, (m_n, f_{k_n})]$ aller Männer und Frauen zu Paaren ($k_1, \ldots, k_n$ ist eine Permutation von $1, \ldots, n$). Dabei heißt eine Zuordnung stabil genau dann, wenn folgendes nicht gilt:
es gibt einen Mann m_i und eine Frau f_j, so dass m_i die Frau f_j sympathischer findet als seine zugewiesene Partnerin f_{k_i} und gleichzeitig f_j den Mann m_i sympathischer findet als ihren zugewiesenen Partner.
Schreiben Sie ein Haskell-Programm, mit dem das Problem der stabilen Heirat gelöst werden kann.

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 6

Abgabedatum: 30.11.2007 16.00 Uhr

Aufgabe 6.1 Einfache String-Funktionen (30 Punkte)

1. Schreiben sie eine Haskell-Funktion `isWordChar :: Char -> Bool`, die für ein Zeichen `True` liefert, wenn es sich dabei um ein Wortzeichen handelt, d.h. das Zeichen ist aus der Menge $\{a, \ldots, z, A, \ldots, Z, -\}$ ist. (5 Punkte)

2. Entwickeln sie eine Haskell-Funktion `replace`, die in einem String einen gegebenen Teilstring durch einen weiteren String ersetzt, d.h. für String-Parameter *str*, *old* und *new* wird in *str* jedes Vorkommnis von *old* durch *new* ersetzt. (10 Punkte)

3. Schreiben sie eine Haskell-Funktion `chopw :: String -> (String,String)`, die für einen String das erste Wort des Strings abschneidet und dieses Wort und den Reststring zurückliefert. Für den String `''Alles wird gut.''` soll die Funktion `chopw` also das Ergebnis `(''Alles'', '' wird gut.'')` zurückliefern. Führende Zeichen, die keine Wortzeichen sind, sollen bei Anwendung der Funktion abgeschnitten werden, d.h. für den String `''... Alles wird gut.''` soll ebenfalls das obige Ergebnis zurückgeliefert werden. (15 Punkte)

Aufgabe 6.2 Zeichenkodierung (30 Punkte)

1. Schreiben sie eine Funktion `chunk::Int -> [a] -> [[a]]` in Haskell, die für eine ganze Zahl n, die gegebene Liste in Listen der Länge n zerlegt. Ein eventuell entstehendes Reststück mit einer Länge kleiner n soll mit zurückgeliefert werden. (10 Punkte)

2. Für einen String s soll seine Bitmusterpräsentation erzeugt werden. Dabei entspricht das Bitmuster eines Strings der Konkatenation der Bitmuster der ASCII-Werte seiner Zeichen. Geben sie eine Haskell-Funktion an, die für einen String das Bitmuster zurückliefert. (10 Punkte)

3. Im folgenden soll aus einem Bitmuster der zugehörige String ermittelt werden. Das Bitmuster soll dabei in je 8 Bit aufgeteilt werden, die für ein ASCII-Zeichen stehen. Schreiben sie eine Haskell-Funktion, die diese Rückwandlung durchführt. (10 Punkte)

Aufgabe 6.3 Dokumentensuche (40 Punkte)

Zur Suche von Dokumenten werden diese häufig in eine Vektordarstellung überführt und der Vektor eines Dokumentes in einer Datenbank gespeichert. Bei der Suche nach einem Dokument wird dann auch die Suchanfrage in einen Vektor überführt und die Ähnlichkeit zu allen Vektoren berechnet. Die Liste der Suchergebnisse enthält dann die Dokumente der ähnlichsten Vektoren.

Im *Vector Space Model* repräsentiert jedes Wort eine Dimension. Der *Wordvector* $\vec{v}$ eines Dokumentes enthält an der i-ten Stelle z.B. die Anzahl der Vorkommnisse des Wortes der i-ten Dimension. Da der Vektor eines Dokumentes in der Regel dünn besetzt ist[1], wird er im folgenden als Liste `[(String,Int)]` angenommen, d.h. für alle nicht vorhandenen Worte ist der Wert 0.

[1] Es werden bei der Bildung des *Wordvectors* üblicherweise nicht alle Worte mit aufgenommen. Meist wird über eine Reihe von sogenannten Stopp-Wörtern und *stem*-Funktionen versucht die Indexwörter zu reduzieren. Stopp-Wörter sind beispielsweise "und", "oder", usw. Die *stem*-Funktionen bilden Worte auf ihren Wortstamm ab, der dann anstelle des eigentlichen Wortes in den Vektor aufgenommen wird. Auf diese Weise werden nur ca. 40-50 % der Worte eines Dokumentes in den Vektor übernommen.

1. Schreiben sie eine Haskell-Funktion `wvector :: String -> [(String,Int)]`, die für ein Dokument (String) eine Liste der Worte und deren Häufigkeit zurückliefert. Das Ergebnis für den Satz `''Alles wird gut.''` sollte dann lauten:

   ```
   [(``Alles'',1), (``wird'',1) , (``gut'',1)]
   ```

   (20 Punkte)

2. Ein häufig verwendetes Ähnlichkeitsmaß für Dokumente ist das einfache Skalarprodukt. Ist das Produkt zweier Vektoren 0, sind sie sich überhaupt nicht ähnlich. Schreiben sie eine Haskell-Funktion `scalar :: [(String,Int)] -> [(String,Int)] -> Int`, die für zwei Wortvektoren das Skalar-Produkt berechnet. (15 Punkte)

3. Schreiben sie eine Haskell-Funktion `search :: String -> [String] -> [(Int,String)]`, die für einen Suchstring s und eine Liste von Dokumenten die Liste der Dokumente, ergänzt mit dem Skalarprodukt mit der Suchanfrage zurückliefert[2]. (5 Punkte)

[2] Schön wäre natürlich eine nach dem Wert des Skalarproduktes absteigend geordnete Liste.

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 7

Abgabedatum: 07.12.2007 16.00 Uhr

Aufgabe 7.1 (20 Punkte)

Das folgende Problem „Türme von Brahma" (bekannt auch als „Türme von Hanoi") geht auf eine hinterindische Sage zurück: In einem im Dschungel verborgenen hinterindischen Tempel Sind Mönche seit Beginn der Zeitrechnung damit beschäftigt, einen Stapel von 50 goldenen Scheiben mit nach oben hin abnehmendem Durchmesser, die durch einen goldenen Pfeiler in der Mitte zusammengehalten werden, durch sukzessive Bewegungen jeweils einer einzigen Scheibe auf einen anderen goldenen Pfeiler umzuschichten. Dabei dürfen sie einen dritten Pfeiler als Hilfspfeiler benutzen, müssen aber darauf achten, dass niemals eine Scheibe mit einem größerem Durchmesser auf eine mit kleinerem Durchmesser zu liegen kommt. Die Sage berichtet, dass das Ende der Welt gekommen ist, wenn die Mönche ihre Aufgabe beendet haben.
Schreiben Sie eine Haskell-Funktion, die bei n Scheiben ausgibt, wie die Scheiben umzuschichten sind. Die einzelnen Schritte sollen dabei durchnummeriert sein.

Aufgabe 7.2 (30 Punkte)

1. Schreiben Sie unter Verwendung von `foldl` eine Haskell-Funktion, die bei Eingabe einer ganzen Zahl n und einer endlichen Liste $[m_1, \ldots, m_k]$ von ganzen Zahlen die kleinste Zahl m_i in dieser Liste mit $m_i > n$ ausgibt. Existiert keine solche Zahl, so soll 0 ausgegeben werden. (10 Punkte)

2. Die Fibonacci-Funktion $fib : \mathbb{N} \to \mathbb{N}$ ist wie folgt definiert:

$$\begin{aligned} fib\ 0 &= 0 \\ fib\ 1 &= 1 \\ fib\ \ (n+2) &= (fib\ n) + (fib\ \ (n+1)) \text{ für } n \geq 0 \end{aligned}$$

   (a) Implementieren Sie die Fibonacci-Funktion in Haskell. (5 Punkte)

   (b) Zeigen Sie, dass für die 2×2-Matrix $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ die folgende Gleichung gilt:

$$\begin{pmatrix} fib\ \ (n+1) \\ fib\ n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} fib\ 1 \\ fib\ 0 \end{pmatrix}$$

   (6 Punkte)

   (c) Verwenden Sie die in der vorigen Teilaufgabe gezeigte Eigenschaft, um die Fibonacci-Funktion effizienter zu implementieren.

   Eine 2×2-Matrix $\begin{pmatrix} u & v \\ w & x \end{pmatrix}$ kann dabei einfach als 4-Tupel (u, v, w, x) repräsentiert werden. (Wer möchte, kann auch auf den Datentyp der Arrays zurückgreifen, der aber im `prelude.hs` nicht enthalten ist und somit importiert werden muss.)
   Anmerkung: Implementiert man die Potenz einer Matrix über eine „divide-and-conquer"-Strategie, so ist die Berechnung der Fibonacci-Funktion besonders effizient. (10 Punkte)

Aufgabe 7.3 (20 Punkte)

Auf einer Menge M lässt sich eine Ordnung durch den paarweisen Vergleich der Elemente definieren. Dieser Vergleich ist eine einfache Funktion `cmp :: a -> a -> Bool`, die angibt, ob das erste Element in der Ordnung vor dem zweiten liegt. (Ein Beispiel dafür ist die Funktion "<").

1. Schreiben sie eine Haskell-Funktion `lessThan :: String -> String -> Bool`, die für zwei Strings angibt, ob der erste vor dem zweiten liegt. Dabei soll für zwei Strings s_1, s_2 gelten, dass s_1 vor s_2 liegt, falls s_1 kürzer ist als s_2 oder bei gleicher Länge in der lexikographischen Ordnung vor s_2 liegt. (5 Punkte)

2. Erweitern sie die aus der Vorlesung bekannte Funktion Quicksort zu einer Funktion höherer Ordnung, indem als zusätzlicher Parameter eine Vergleichsfunktion übergeben wird. Das Ergebnis des Quicksort-Algorithmus soll dann eine Liste sein, deren Elemente entsprechend der Vergleichsfunktion sortiert sind. (15 Punkte)

Aufgabe 7.4 (30 Punkte)

Gegeben sei eine Menge von Punkten (Knoten) $V = \{1, \ldots, n\}$, die z.B. unterschiedliche Orte/Städte darstellen. Eine Menge von Kanten $E \subseteq \{\,(i,j) \mid i,j \in V, i \neq j\}$ stellt die direkten *gerichteten* Verbindungen zwischen den Punkten dar. Der dadurch definierte Graph $G = (V, E)$ lässt sich in Form einer *Adjazenzmatrix* A beschreiben, wobei für die Einträge $a_{i,j}$ von A gilt

$$a_{i,j} = \begin{cases} 1, & \text{falls } (i,j) \in E \\ 0, & \text{sonst} \end{cases}$$

Über Potenzbildung der Adjazenzmatrix A lassen sich erweiterte Wege in G finden. So enthält z.B. die Matrix

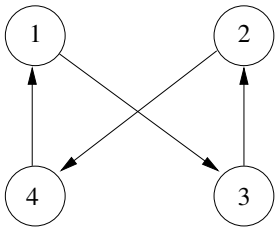
$$A^k = \underbrace{A \cdot A \cdot \ldots \cdot A}_{k\text{-mal}}$$

an der Stelle $a_{i,j}$ die Anzahl der Wege vom Knoten i zum Knoten j, deren Länge genau k beträgt. Eine Matrix lässt sich in Haskell kompakt als Liste `[(Int,Int,Num)]` beschreiben. Das Element an Zeile i und Spalte j ist dann $(i, j, a_{i,j})$, bzw. $(i, j, 0)$, falls die Liste keinen Eintrag $(i, j, \_)$ enthält.

1. Geben sie eine konstante Haskell-Funktion an, die die Adjazenzmatrix des in Abbildung (a) dargestellten Graphen zurückliefert. (5 Punkte)

2. Schreiben sie eine Haskell-Funktion, die für eine Matrix A in genannter Form die n-te Potenz der Matrix berechnet. (20 Punkte)

3. Geben sie eine Haskell-Funktion an, die anhand der Adjazenzmatrix feststellt, ob ein Graph einen Kreis der Länge n enthält. (5 Punkte)



(a) Beispiel-Graph

$$A = A^1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

(b) Zugehörige Adjazenzmatrix

Anmerkung: Um Tipparbeit zu sparen, läßt sich der Typ `[(Int,Int,Num)]` als eigenständiger Typ definieren:

```
type Matrix = [(Int,Int,Num)]
```

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 8

Abgabedatum: 14.12.2007 16.00 Uhr

Aufgabe 8.1 (25 Punkte)

1. Definiere eine Haskell-Funktion `sit::((Integer,Integer)-> Integer)->Integer->Integer`, die zu einer zweistelligen Funktion $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ und zu $x \in \mathbb{N}$ den Wert

$$\texttt{sit}\ f\ x = \begin{cases} 0 & \text{falls } x = 0 \\ 1 & \text{falls } x = 1 \\ \underbrace{f\,(x, f\,(x, f\,(x, \ldots f\,(x, x))))}_{x-1 \text{ malige } f\text{-Anwendung}} & \text{falls } x > 1 \end{cases}$$

   berechnet. So soll z.B. für $x = 4$ der Wert von $f(4, f(4, f(4, 4)))$ berechnet werden. (15 Punkte)

2. Die Catalan-Zahlen, oder Catalansche Zahlen, benannt nach dem belgischen Mathematiker Eugène Charles Catalan (1814–1894), stellen eine Folge natürlicher Zahlen dar, die in vielen Problemen der Kombinatorik auftaucht. Die n-te Catalan-Zahl C_n ist z.B. die Anzahl der verschiedenen Möglichkeiten, ein konvexes $(n + 2)$-Eck durch Diagonalen in Dreiecke zu zerteilen (Triangulation).Für die Catalan-Zahlen gilt die folgende Rekursionsformel

$$\begin{aligned} C_0 &= 0 \\ C_{n+1} &= \sum_{k=0}^{n} C_k \cdot C_{n-k}. \end{aligned}$$

   Geben Sie auf der Basis dieser Rekursionsformel eine rekursive Definition der Catalan-Zahlen in Haskell. (10 Punkte)

Aufgabe 8.2 (25 Punkte)

Ein sehr einfach zu implementierendes Sortierverfahren ist *Bubblesort*, das nach dem folgenden Algorithmus arbeitet: (n sei die Länge der Liste):

- Im j-ten Durchlauf (mit $j = 1$ beginnend):

  Für jedes i von $i = 1$ bis $n - j$: falls das $(i + 1)$ kleiner als das i-te Listenelement ist, vertausche die Positionen dieser beiden Elemente.

Entwerfen Sie ein polymorphes Programm für diesen Algorithmus und sortieren Sie mit diesem Programm folgende Listen bzgl. $\leq$:

- $[3, 5, 4, 1, 2]$
- $[3.2, 5.7, 4.3, 1.8, 2.5]$
- [“ein”, “Kaugummi”, “Dir”, “Bubble”, “Gut”]

Aufgabe 8.3 (20 Punkte)

Geben Sie eine Haskell-Funktion an, die für einen gerichteten Graphen überprüft, ob der Graph topologisch sortiert werden kann. Falls dies der Fall ist, soll eine topologische Sortierung ausgegeben werden.

Aufgabe 8.4 (30 Punkte)

Ein Labyrinth kann als ein ungerichteter Graph aufgefasst werden, in dem 2 Knoten ausgezeichnet sind: der Ausgang aus dem Labyrinth und der Knoten (Punkt), von dem aus der Ausweg aus dem Labyrinth gefunden werden soll.

Schreiben Sie eine Haskell-Funktion, die zu einem Labyrinth einen Ausweg bestimmt.

Zusatzaufgabe 8.5 (Bearbeitung freiwillig, 15 zusätzliche Punkte möglich)

Auf Blatt 7 wurden Wege der Länge k über die k-te Potenz der Adjazenzmatrix gefunden. Beweisen Sie, dass die Adjazenzmatrix in der k-ten Potenz genau an den Stellen $a_{i,j}$ ungleich 0 ist, für die ein Weg der Länge k vom Knoten i zum Knoten j existiert.

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 9

Abgabedatum: 21.12.2007 16.00 Uhr

Anmerkung: Das Parsen mit Haskell wurde in der Vorlesung vom 14.12.2007 anhand der Folien des Hutton-Buchs erläutert. Der vorgestellte Ansatz ist eine Vereinfachung des Prinzips monadischer Parser, ohne die z.B. die `do`-Anweisung nicht funktioniert. Die auf den Folien vorgestellten Parser sind in dieser Form nicht lauffähig. Zur Bearbeitung dieses Übungszettels ist daher die Bibliothek `Parsing.lhs` des Hutton-Buches notwendig. Die Bibliothek und Informationen zum Einbinden der Bibliothek finden sich unter

`http://www-ai.cs.uni-dortmund.de/LEHRE/VORLESUNGEN/FP/parsing.html`

Aufgabe 9.1 (25 Punkte)

Boolesche Ausdrücke können über eine kontextfreie Grammatik eingeführt werden, die der kontextfreien Grammatik für arithmetische Ausdrücke sehr ähnelt:

$$\begin{array}{lcl} bexpr & \longrightarrow & bterm\ (\text{"||"}\ bexpr \mid \varepsilon) \\ bterm & \longrightarrow & bfactor\ (\text{"\&\&"}\ bterm \mid \varepsilon) \\ bfactor & \longrightarrow & \text{"not"}\ bfactor \mid batom \mid \text{"("}\ bexpr\ \text{")"} \\ batom & \longrightarrow & \text{"True"} \mid \text{"False"} \end{array}$$

Schreiben Sie einen Parser für Boolesche Ausdrücke, der wie im Falle der arithmetischen Ausdrücke den geparsten Ausdruck direkt auswertet.

Aufgabe 9.2 (35 Punkte)

Auf Blatt 7 wurde für einen Graphen die Repräsentation als Adjazenzmatrix benutzt. Diese Matrix wurde in Haskell als Liste `[(i,j,1)]` dargestellt.

1. Schreiben sie einen Parser, der einen String liest und eine Liste von Kanten $(i, j, 1)$ zurückliefert. Dabei soll der String das folgende Format haben:

   ```
   ‘‘1->2,3->1,1->4’’
   ```

   D.h. die Kanten werden jeweils durch ihre beiden Knoten, getrennt durch `‘‘->’’`, beschrieben und als Komma-separierte Liste aufgezählt. Der obige String beispielsweise würde die folgende Adjazenzmatrix A repräsentieren:

   $$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(22 Punkte)

2. Erweitern sie ihren Parser so, dass durch ``i-j'' auch ungerichtete Kanten angegeben werden können (d.h. in diesem Fall würde sowohl die Kante $(i, j, 1)$ als auch $(j, i, 1)$ erzeugt). (13 Punkte)

Aufgabe 9.3 (40 Punkte)

Im folgenden sein ein einfacher regulärer Ausdruck ein String aus Zeichen eines Alphabets $\Sigma = \{a, \ldots, z\}$ und optionalen Multiplikatoren $*$ und ?. Dabei entspricht $*$ einem Vorkommen einer beliebigen Anzahl des vorangegangenen Zeichens. Im Fall von ? soll das vorangegangene Zeichen garnicht oder maximal einmal vorkommen. Ohne explizite Angabe eines Multiplikators, steht ein Zeichen für *genau ein* Vorkommnis dieses Zeichens (impliziter Multiplikator). Der Ausdruck

```
``a*bc?''
```

beschreibt beispielsweise einen String, der aus beliebig vielen aufeinanderfolgenden Zeichen a besteht, gefolgt von genau einem Zeichen b und optional dem Zeichen c.
Ein einfacher regulärer Ausdruck läßt sich als Liste `[(Char,Char)]` darstellen, wobei jedes Tupel aus einem Zeichen und seinem Multiplikator besteht. Der obige Ausdruck ließe sich dabei als Liste

```
[ ('a', '*'), ('b', '1'), ('c', '?') ]
```

darstellen, wobei der Multiplikator `'1'` der implizite Multiplikator ist.

1. Schreiben sie einen Parser, der einen einfachen regulären Ausdruck liest und eine Liste von Zeichen mit ihren Multiplikatoren zurückliefert. (15 Punkte)

2. Schreiben sie eine Funktion `compile:: [(Char,Char)] -> Parser String`, die für eine Liste in obiger Form, einen Parser zurückliefert, der einen String gemäß der Zeichen und ihren minimalen und maximalen Vorkommnissen parst. (20 Punkte)

3. Schreiben sie eine Funktion `match:: String -> String -> Bool`, die für einen regulären Ausdruck r und einen String s überprüft, ob s dem regulären Ausdruck r entspricht. (5 Punkte)

FB Informatik LS I – Funktionale Programmierung – Prof. Dr. P. Padawitz
Christian Bockermann, Hubert Wagner

Übungen zur Vorlesung

Funktionale Programmierung

Wintersemester 2007/08

Aufgabenblatt 10

Abgabedatum: 11.01.2008 16.00 Uhr

Aufgabe 10.1 (30 Punkte)

Unter einem *Fraktal* versteht man zumeist ein Bild, dass eine hohe Selbstähnlichkeit enthält. Diese ergibt sich z.B. in dem man ein Bild aus einer einfachen geometrischen Form bildet, die immer wieder (ggf. leicht verändert) aneinander gehängt wird. Ein Beispiel für eine derartige Konstruktion ist der Pythagoras-Baum, der sich aus stetig aneinandergesetzten Quadraten ergibt:

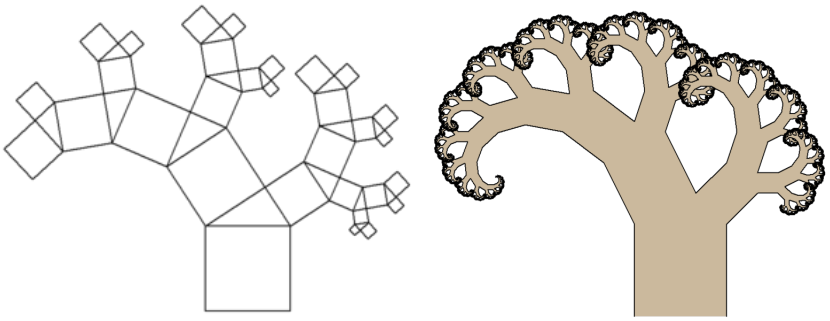


Abbildung 1: Pythagoras-Baum mit unterschiedlich hoher Anzahl von Iterationen

Unter Benutzung der SOE-Graphics-Bibliothek (Graphics.SOE) können derartige Grafiken mit Haskell leicht erstellt werden. Einige Hinweise zu der Bibliothek und eine Code-Vorlage, die die Bearbeitung dieser Aufgabe erleichtert, finden Sie unter

`http://www-ai.cs.uni-dortmund.de/LEHRE/VORLESUNGEN/FP/grafik.html`

1. In dieser Aufgabe soll ein Schneeflocken-Fraktal auf die folgende Weise generiert werden:

   Zunächst wird ein sogenannter Davidsstern gezeichnet. Dieser besteht aus 2 gleich großen, gefärbten gleichseitigen Dreiecken, wobei das zweite Dreieck gegenüber dem ersten um 180° gedreht ist. Beide Dreiecke haben aber den gleichen Schwerpunkt (= Schnittpunkt der Seitenhalbierenden). Das Schneeflocken-Fraktal ergibt sich nun dadurch, dass rekursiv für jeden schon erzeugten Davidsstern in den 6 Spitzen (Ecken) wiederum ein Davidsstern, dann aber in einer anderen Farbe, erzeugt wird, der nur noch ein Drittel so hoch ist. Dies wird so lange durchgeführt, wie der Davidsstern noch sinnvoll dargestellt werden kann.

   Modifizieren Sie den zur Verfügung gestellten Haskell-Code zum Zeichnen eines Davidssterns so, dass mit dem resultierenden Programm ein Schneeflocken-Fraktal gezeichnet wird. Ersetzen Sie dazu die Zeile

```
david w 500 400 200 Blue
```

in geeigneter Weise. (15 Punkte)

2. Schreiben sie eine Haskell-Funktion, die den oben abgebildeten Pythagoras-Baum erstellt. Die Funktion soll dabei als Parameter die Anzahl der Rekursionen erhalten, die ausgeführt werden.

(15 Punkte)

Aufgabe 10.2 (40 Punkte)

Geben Sie eine Funktion `readLine :: IO String` an, die sich wie die Funktion `getLine` verhält. Zusätzlich soll es dabei möglich sein, mit der *Entfernen*-Taste Zeichen in der Eingabe zu löschen.

Hinweis: Die *Entfernen*-Taste erzeugt das Zeichen `'\DEL'`, um den Cursor ein Zeichen zurück zu bewegen wird die Kontrollsequenz "`\ESC[1D`" benutzt.

Aufgabe 10.3 (30 Punkte)

Das "Nimm!"-Spiel wird von zwei Spielern gespielt und besteht aus einem Feld von 5 Reihen. Zu Beginn des Spiels enthält jede Reihe eine Anzahl von Sternen:

```
1:  * * * * *
2:  * * * *
3:  * * *
4:  * *
5:  *
```

Die Spieler nehmen nun abwechselnd einen oder mehrere Sterne aus den Reihen. Der Spieler, der den letzten Stern vom Spielfeld nimmt hat gewonnen.

Implementieren sie das Spiel in Haskell.