Peter Padawitz / Hubert Wagner Dortmund, den 12.04.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Vorbemerkung

Die Übungen beginnen am 19.04.05 und finden jeweils dienstags in der Zeit von 8.30 - 10.00 Uhr im SR 318/GB4 statt.
Um einen Leistungsnachweis zu erhalten, sind folgende Kriterien zu erfüllen:

- Aktive und regelmäßige Teilnahme an den Übungen (höchstens zweimaliges unentschuldigtes Fehlen erlaubt)
- Erreichen von 50% der Punkte, die bei den Aufgaben insgesamt erzielt werden können.

Gemeinsame Abgaben von Gruppen bis zu 3 Personen sind zugelassen.

Aufgabenblatt 0

Keine Abgabe - nur Präsenzaufgaben

Aufgabe 0.1
Geben Sie jeweils eine `Haskell`-Funktion an, die zu einer einer Liste von reellen Zahlen

1. das maximale und das minimale Element dieser Liste sowie
2. den Listendurchschnitt

ermittelt.

Aufgabe 0.2
Schreiben Sie `Haskell`-Funktionen *dual*2*int* und *int*2*dual*, die zu einer Dualzahl die zugehörige ganze Zahl und zu einer positiven ganzen Zahl die zugehörige Dualzahl berechnen.

Aufgabe 0.3
In `Haskell98` gibt es den (Daten-) Typ `Ratio Integer` für die Menge der rationalen Zahlen.

1. 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.)
2. Schreiben Sie eine `Haskell`-Funktion, die bei Eingabe der positiven ganzen Zahl n die Summe der in der 1. Teilaufgabe ausgegeben rationalen Zahlen berechnet.

Aufgabe 0.4
Eine positive ganze Zahl n heißt perfekt, falls sie gleich der Summe aller ihrer Faktoren ist, die kleiner als die Zahl n selbst sind.
Schreiben Sie eine `Haskell`-Funktion, die bei Eingabe n die Liste aller perfekten Zahlen $\leq n$ ausgibt.

Peter Padawitz / Hubert Wagner Dortmund, den 19.04.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 1

Abgabe: bis 25.04.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 1.1
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 1.2
Schreiben Sie eine `Haskell`-Funktion, die bei Eingabe einer Liste von Strings in jedem String dieser Liste das erste Symbol, wenn es ein Buchstabe ist, in den entsprechenden Großbuchstaben umwandelt.

Aufgabe 1.3
Wir betrachten Binärbäume, deren Knoten mit Elementen eines Typs a, der eine lineare Ordnung besitzen soll, markiert sind.
Geben Sie eine Haskell-Funktion an, die bei Eingabe eines endlichen Binärbaumes die Tiefe des (bzw. eines) Knoten berechnet, der im Baum mit dem größten Element markiert ist.

Aufgabe 1.4

1. Geben Sie eine Neudefinition von `filter p` unter Verwendung von `foldr`.
2. Schreiben Sie unter Verwendung von `foldl` eine Haskell-Funktion, die in einer Liste Duplikate eliminiert.

Peter Padawitz / Hubert Wagner Dortmund, den 26.04.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 2

Abgabe: bis 02.05.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 2.1
Das Sieb des Eratosthenes beschreibt ein Verfahren, um alle Primzahlen unterhalb einer Schranke n zu bestimmen:

1. Man schreibe alle natürlichen Zahlen von 2 bis n auf.
2. Man markiere die Zahl 2 und streiche dann jede zweite Zahl.
3. Ist k die erste nicht-gestrichene und nicht-markierte Zahl, so markiere man k und streiche dann jede nachfolgende k-te Zahl (gleichwertig: jede Zahl, die durch k teilbar ist).

Übrig bleiben dann genau die Primzahlen $\leq n$. (In dem Verfahren, das beschrieben wurde, kann natürlich k beschränkt werden durch $k \leq \sqrt(n)$.)

Schreiben Sie eine Haskell-Funktion, die mit Hilfe des Siebes von Eratosthenes die unendliche Liste der Primzahlen erzeugt. Verwenden Sie diese Funktion, um damit zu testen, ob eine Zahl eine Primzahl ist.

Aufgabe 2.2
Wir wollen Polynome in einer Variablen x und mit reellwertigen Koeffizienten über unendliche Listen repräsentieren:

```
type Poly = [Float],
```

so dass insbesondere auch Taylorreihen dargestellt werden können.

Schreiben Sie Haskell-Funktionen zur Berechnung von Summe und Produkt von Polynomen.

Aufgabe 2.3
Schreiben Sie eine Haskell-Funktion, die folgendes leistet:

Bei Eingabe einer natürlichen Zahl n und einer Menge V von Knoten in Form einer Liste soll die Funktion mittels eines durch n gesteuerten Zufallsgenerators einen gerichteten Graphen mit V als Knotenmenge erzeugen. Der erzeugte Graph soll dabei als Adjazenzmatrix vorliegen.

Für die Verwendung eines Zufallszahlengenerators greifen Sie auf das Module Random.hs zurück. Z.B. wird mit `{l :: [Int]; l = randomRs (a, b) (mkStdGen n)}` eine Liste von ganzzahligen Zufallszahlen im Intervall von a bis b erzeugt. n beinflusst dabei das Erzeugungsverfahren der Zufallszahlen. Für unterschiedliche n erhalten Sie unterschiedliche Listen von Zufallszahlen.

Aufgabe 2.4
Zu einem gerichteten Graphen G, der über eine Adjazenzmatrix repräsentiert ist, kann seine reflexive und transitive Hülle $G^\star$ über die Multiplikation Boolescher Matrizen berechnet werden.

Schreiben Sie eine Haskellfunktion, die dies leistet.

Peter Padawitz / Hubert Wagner Dortmund, den 02.05.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 3

Abgabe: bis 09.05.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 3.1

1. Geben Sie ein Haskell-Programm an, das die unendliche Liste der Partialsummen der Fibonacci-Zahlen generiert. Das n-te Listenelement ($n \geq 0$) in dieser Liste soll also gleich der Summe der ersten n Fibonacci-Zahlen sein.

2. Die Hamming-Zahlen sind die Zahlen $2^i 3^j 5^k$ für $i, j, k \in \mathbb{N}$

   Schreiben Sie ein Haskell-Programm, das die unendliche Liste der Hamming-Zahlen in aufsteigender Folge generiert.

Aufgabe 3.2

Implementieren Sie in Haskell Addition, Subtraktion, Multiplikation und Division von komplexwertigen Brüchen, d.h. von Zahlen $a + i \cdot b$, wobei a und b rationale Zahlen (Typ: Ratio Integer) sind

Aufgabe 3.3

Ein *2-3 Baum* ist ein Baum, bei dem jeder innere Knoten 2 oder 3 Söhne hat und in dem jedes Blatt die gleiche Tiefe aufweist. Eine linear geordnete Menge kann durch einen solchen 2-3 Baum dadurch repräsentiert werden, dass die Elemente dieser Menge den Blättern des 2-3 Baumes zugeordnet werden.

1. Geben Sie den polymorphen Datentyp `Tree23 a` für 2-3 Bäume an.

2. Wir wollen eine Folge von ganzen Zahlen aus einem String einlesen und diese Zahlen dann in einen 2-3 Baum des entsprechenden Typs einfügen. Z.B. würde der String "12 4 36 110"die Zahlenfolge 12, 4, 36, 110 repräsentieren.

   Schreiben Sie ein Haskell-Programm, das bei Eingabe eines solchen „Zahlenstrings" die darin enthaltenen Zahlen in einen 2-3 Baum einfügt und anschließend diesen 2-3 Baum ausgibt.

3. Geben Sie eine Haskell-Funktion `f :: Enum a => Tree23 a -> Tree23 a` an, die in jedem Blatt des 2-3 Baumes die Markierung durch den Nachfolgerwert ersetzt. Ist z.B. ein Blatt mit dem Character `c` markiert, so soll es anschließend mit `d` markiert sein.

Peter Padawitz / Hubert Wagner Dortmund, den 09.05.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 4

Abgabe: bis 16.05.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 4.1
In dieser Aufgabe betrachten wir determinierte, endliche Baumautomaten mit Eingabealphabet $\{0, 1\}$. Dies sind endliche Automaten, die als Eingaben endliche Binärbäume erhalten, deren Knoten mit 0 oder 1 markiert sind. Hierbei arbeiten diese Baumautomaten in der folgenden Weise:

Der Automat startet im Startzustand q_0. Befindet sich der Automat an dem inneren Knoten i des Baumes im Zustand q, ist der Knoten i mit dem Eingabesymbol a markiert und gilt für die Übergangsfunktion δ des Automaten $\delta(q, a) = (q_1, q_2)$, so geht der Automat für den linken Sohn in den Zustand q_1 und für den rechten Sohn in den Zustand q_2 über, anschließend arbeitet er parallel in den beiden Söhnen weiter. Der Automat akzeptiert den Binärbaum, wenn er in allen Blättern einen Endzustand aus F annimmt.

Formal ist der determinierte, endliche Baumautomat ein Tupel $A = (\{0, 1\}, Q, \delta, q_0, F)$, wobei

- $\{0, 1\}$ das Eingabealphabet
- Q die endliche Zustandsmenge
- q_0 der Startzustand
- $\delta : Q \times \{0, 1\} \to Q \times Q$ die Übergangsfunktion und
- $F \subseteq Q$ die Menge der Endzustände ist.

Schreiben Sie ein Haskell-Programm, das solche Baumautomaten über Instanziierungen der Funktor-Klasse realisiert, d.h. über eine geeignete Definition von `fmap`. Verwenden Sie hierzu den Datentyp `Int` für die Zustände.

Aufgabe 4.2
Wir betrachten die Situation, dass ein Vermieter ein Haus mit 4 Mietwohnungen hat, die allesamt vermietet sind. Die Größe der Wohnungen betrage 80 qm, 100 qm, 50 qm sowie 40 qm. Einmal im Jahr muss der Vermieter eine Nebenkostenabrechnung durchführen. Diese betreffen die Stromkosten, die gleichmäßig auf die 4 Wohnungen aufgeteilt werden, die Kosten für den Wasserverbrauch, die sich über die Anzahl der Personen pro Mietwohnung errechnen, sowie die anteilige Grundsteuer, die nach der Wohnfläche der Mietwohnung berechnet wird.

Schreiben Sie ein Haskell-Programm, das eine solche Nebenkostenabrechnung ausführt. Stromkosten, Gesamtwasserverbrauch sowie Grundsteuer sollen dabei interaktiv abgefragt werden.

Peter Padawitz / Hubert Wagner Dortmund, den 17.05.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 5

Abgabe: bis 23.05.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 5.1
Schreiben Sie ein Haskell-Programm, das folgendes leistet:

Nach Aufruf des Programms wird eine Zeile eingelesen. Anschließend wird getestet, ob der Zeileninhalt ein Palindrom darstellt und es wird das Ergebnis mitgeteilt. Danach wird wieder eine Zeile eingelesen, der Test auf die Palindromeigenschaft ausgeführt usw. . Das Programm endet, falls eine Leerzeile eingegeben wird.

Aufgabe 5.2
Ein *2-3 Baum* ist ein Baum, bei dem jeder innere Knoten 2 oder 3 Söhne hat und in dem jedes Blatt die gleiche Tiefe aufweist. Eine linear geordnete Menge kann durch einen solchen 2-3 Baum dadurch repräsentiert werden, dass die Elemente dieser Menge den Blättern des 2-3 Baumes zugeordnet werden.

Schreiben Sie ein Haskell-Programm, das mit Hilfe von 2-3 Bäumen ein Wörterbuch (dictionary) realisiert, das menügesteuert oder per Abfrage die folgenden Funktionen unterstützt:

- ein neues Wörterbuch anlegen
- ein vorhandenes Wörterbuch öffnen
- ein Element in ein geöffnetes Wörterbuch einfügen
- ein Element in einem geöffneten Wörterbuch löschen
- Testen, ob ein Element in einem geöffneten Wörterbuch vorkommt
- Ausgabe des Wörterbuchs

Peter Padawitz / Hubert Wagner Dortmund, den 24.05.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 6

Abgabe: bis 31.05.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 6.1
In Aufgabe 4.1 wurden endliche deterministische Baumautomaten betrachtet.
Schreiben Sie ein Haskell-Programm, das bei Eingabe eines nichtdeterministischen endlichen Baumautomaten sowie eines Baumes als Eingabe für den Baumautomaten feststellt, ob der Baumautomat diesen Baum akzeptiert. Verwenden Sie dazu eine Monaden-Konstruktion.

Aufgabe 6.2
Gegeben sei ein $n \times n$-Schachbrett ($n \geq 3$).
Schreiben Sie ein Haskell-Programm, das für die Springerfigur eine Zugfolge bestimmt (sofern es eine gibt), bei dem der Springer jedes Schachfeld genau einmal besucht.

Zur Erinnerung: ein Springer bewegt sich entweder um 2 Felder in der vertikalen Richtung und um ein Feld in der horizontalen Richtung oder aber um 2 Felder in der Horizontalen und um ein Feld in der Vertikalen.

Peter Padawitz / Hubert Wagner Dortmund, den 31.05.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 7

Abgabe: bis 06.06.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 7.1
Schreiben Sie eine Haskell-Funktion

```
accumulate :: [IO a] -> IO [a]
```

die eine Folge von Interaktionen ausführt und am Ende die Resultate in einer Liste ausgibt.

Aufgabe 7.2
Ein $n \times n$-Schiebepuzzle besteht aus einem Quadrat mit n^2 Feldern. Bis auf ein Feld enthält jedes dieser Felder ein Plättchen, das mit einer Zahl zwischen 1 und $n^2 - 1$ markiert ist. Keine Zahl kommt dabei mehrfach als Markierung vor. Ein Plättchen darf nur zu einem benachbarten freien Feld verschoben werden.
Die Aufgabenstellung des Schiebepuzzles besteht nun darin, ausgehend von einer zufälligen Verteilung der Plättchen die Plättchen so zu verschieben, dass am Ende die Plättchen in aufsteigender Nummerierung vorliegen, und zwar mit der linken oberen Ecke beginnend und dann zeilenweise fortfahrend. Am Ende ist somit das Feld n^2 frei.

Implementieren Sie das Schiebepuzzle in Haskell.

Aufgabe 7.3
Schreiben Sie ein Haskell-Programm, das den Inhalt einer Textdatei einliest und ihn im Blocksatz formatiert auf dem Bildschirm ausgibt. (Die Anzahl der Zeichen je Zeile können Sie selbst festlegen. Es ist dabei unterstellt, dass die Wortlänge die Zeilenlänge nicht überschreitet.)

Peter Padawitz / Hubert Wagner Dortmund, den 07.06.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 8

Abgabe: bis 13.06.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 8.1
Eine Registermaschine (kurz: RM) besteht aus einer zentralen Recheneinheit, einem Speicher und aus einem Programm.
Die zentrale Recheneinheit ihrerseits besteht aus zwei Registern: dem Befehlszähler und dem Akkumulator.
Der Speicher enthält unendlich viele Register: $R_1, R_2, \ldots$ mit den Adressen $1, 2, \ldots$. Der Akkumulator hat die Adresse 0.
Ein RM-Programm besteht aus einer endlichen Folge von RM-Befehlen, die mit 1 beginnend aufsteigend durchnummeriert sind. Wir legen die folgenden RM-Befehle zugrunde:

1. Ein- und Ausgabebefehle:
   `LOAD` i
   `STORE` i
2. Arithmetische Befehle:
   `ADD` i
   `SUB` i
   `MULT` i
   `DIV` i
3. Arithmetische Befehle mit Konstanten:
   `CLOAD` i
   `CADD` i
   `CSUB` i
   `CMULT` i
   `CDIV` i
4. Sprungbefehle:
   `GOTO` j
   `JZERO` j
   `END`

Es gibt in diesem Fall also keine indirekte Adressierung.

Geben Sie einen geeigneten Datentyp für Registermaschinen-Programme an. Schreiben Sie dann ein Haskell-Programm, mit dem ein RM-Programm, das als Textdatei gespeichert ist, eingelesen und dann anschließend simuliert wird. Das Programm soll insbesondere auch eine Simulation in Einzelschritten ermöglichen, so dass die Arbeitsweise der RM Schritt für Schritt nachvollzogen werden kann.

Peter Padawitz / Hubert Wagner Dortmund, den 14.06.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 9

Abgabe: bis 20.06.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 9.1
Schreiben Sie ein Haskell-Programm, das ohne zusätzliche Abfrage nach Eingabe einer natürlichen Zahl bzw. einer römischen Zahl diese in eine römische bzw. in eine natürliche Zahl transformiert.

Aufgabe 9.2
Schreiben Sie eine Haskell-Funktion, mit der die k-te Potenz einer $n \times n$-Matrix effizient nach der divide-and-conquer Strategie berechnet wird.

Peter Padawitz / Hubert Wagner Dortmund, den 21.06.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 10

Abgabe: bis 27.06.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 10.1
Implementieren Sie das Spiel „Vier gewinnt" entweder unter Verwendung der SOE- Graphik-Bibliothek, die mit der neuesten hugs98-Version mitgeliefert wird, oder unter Verwendung von wxhaskell, falls Sie mit ghc arbeiten bzw. arbeiten wollen.

In diesem Spiel soll ein Spieler gegen den Computer spielen, wobei die Spielzüge des Computers über α-β-pruning errechnet werden sollen.

Bemerkung:

wxhaskell bietet sehr viel mehr Möglichkeiten als die SOE-Graphik-Bibliothek, für die Aufgabenstellung ist aber letztere durchaus ausreichend.

Peter Padawitz / Hubert Wagner Dortmund, den 05.07.2005

Übungen
Einführung ins funktionale Programmieren

Sommersemester 2005

Aufgabenblatt 11

Abgabe: bis 11.07.05 10.00 Uhr in Raum 102/GB4 oder per email (Hubert.Wagner@udo.edu)

Aufgabe 11.1
In den zur Verfügung gestellten Folien ist der TDIDT-Algorithmus zum Erlernen von Entscheidungsbäumen angegeben.

Implementieren Sie diesen Algorithmus in Haskell und benutzen Sie ihn dann für die folgende Problemstellung:

Entlassungswelle in einer großen Firma: Aus heiterem Himmel bekommen Angestellte ihre Kündigung oder werden an andere Arbeitsplätze versetzt. Im Betriebsrat grübelt man, welche Empfehlungen die kürzlich angeheuerte Unternehmensberatung der Unternehmensführung gegeben haben mag. Folgende Daten über die Entlassungen wurden bisher gesammelt:

Pers.-Nr.	Abteilung	Firmenzugehörigkeit	Alter	Tätigkeit	**Klassifiz.**
1	EDV	kurz	jung	Sachbearbeiter	+
2	EDV	kurz	jung	Führungsposition	-
3	EDV	kurz	älter	Führungsposition	+
4	EDV	lang	älter	Sachbearbeiter	-
5	Kundenbetreuung	kurz	jung	Führungsposition	+
6	Kundenbetreuung	lang	älter	Führungsposition	+
7	Kundenbetreuung	kurz	jung	Sachbearbeiter	-
8	Marketing	kurz	jung	Sachbearbeiter	-
9	Marketing	lang	jung	Führungsposition	-
10	Marketing	lang	älter	Sachbearbeiter	+

"Kurze" Firmenzugehörigkeit heißt hier "weniger als 5 Jahre", "jung" bedeutet "jünger als 35". Die Klassifizierung "+", bedeutet, dass eine Kündigung oder Versetzung ausgesprochen wurde[1].

Bestimmen Sie einen Entscheidungsbaum zur Klassifizierung der Kündigungen/Versetzungen. Gehen Sie bei der Attributauswahl (Abteilung, Firmenzugehörigkeit, Alter, Tätigkeit) wie folgt vor: Das Attribut, das die meisten Beispiele eindeutig klassifiziert, wird für die nächste Verzweigung ausgewählt. Gibt es kein solches Attribut oder mehrere Attribute mit derselben Anzahl eindeutig klassifizierter Beispiele, gilt die Reihenfolge der Attribute in der Tabelle (von links nach rechts).

[1] Kündigung und Versetzung sind hier gleichermaßen als "mit + klassifiziert" zu betrachten, die Option "Versetzung" soll das Beispiel nur etwas sozial verträglicher machen.