Funktionale Programmierung

`http://proglang.informatik.uni-freiburg.de/teaching/functional-programming/2013/`

Übungsblatt 2 (Rekursion, List-Comprehensions, Datentypen)

Di, 2013-10-29

Hinweise

- Lösungen sollen als Haskell Quellcode in das persönliche Subversion (svn) Repository hochgeladen werden. Die Adresse des Repositories wird per Email mitgeteilt.
- **Alle** Aufgaben müssen bearbeitet und pünktlich abgegeben werden. Falls das sinnvolle Bearbeiten einer Aufgaben nicht möglich ist, kann eine stattdessen eine Begründung abgegeben werden.
- Wenn die Abgabe korrigiert ist, wird das Feedback in das Repository hochgeladen. Die Feedback-Dateinamen haben die Form `Feedback-<user>-ex<XX>.txt`.
- Allgemeinen Fragen zum Übungsblatt können im Forum (`http://proglang.informatik.uni-freiburg.de/forum/viewforum.php?f=38`) geklärt werden.

Abgabe: Di, 2013-10-05

Aufgabe 1 (Fib, oder auch „Hello World! der Funktionalen Programmierung")
Definieren Sie die Funktion, welche für eine natürliche Zahl n, die n-te Fibunacci Zahl berechnet:

$$\text{fib}(n) = \begin{cases} 0 & \text{für } n = 0 \\ 1 & \text{für } n = 1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{sonst} \end{cases}$$

Die naive Implementierung kommt nicht weit; auf dem Computer des Assistenten braucht fib(30) schon mehrere Sekunden. Definieren Sie eine optimierte Variante, die mindestens fähig ist, die 20000ste Fibunacci Zahl in akzeptabler Zeit zu berechnen.

Testen Sie die optimierte Variante anhand der naiven Spezifikation. (Sie müssen die Testgröße einschränken, siehe Vorlesung).

Aufgabe 2 (Undup)
Definieren und testen Sie eine Funktion undup, die mehrfach vorkommende Elemente aus einer Liste entfernt. (Es gibt bereits eine Bibliotheksfunktion Data.**List**.**nub** die diese Funktionalität implementiert... die sollen Sie natürlich nicht verwenden)

Hinweise: Falls Sie beim Testen von Properties mit QuickCheck Typfehler bekommen, versuchen Sie zuerst den eine Typsignatur an die Properties zu schreiben.

Aufgabe 3 (Kleinster Faktor)
Eine natürliche Zahl $k \geq 2$ ist *Faktor* einer Zahl n, falls ein m existiert, so dass $n = k \cdot m$. Definieren Sie eine Funktion, die zu einem gegebenen **Integer** den kleinsten Faktor berechnet.

Versuchen Sie die Aufgaben auf zwei verschiedene Arten zu lösen und testen Sie diese gegeneinander.

Aufgabe 4 (Media-Bibliothek)
Viele Musikwiedergabeprogramme, wie iTunes, Windows Media Player oder Amarok, haben eine integrierte Media-Bibliothek, in der die zur Verfügung stehenden Musikstücke organisiert und mit Meta-Daten, wie Beliebtheit oder Anzahl der Abspielungen, in Verbindung gebracht werden. In dieser Aufgaben sollen Sie das Datenmodell für eine Mini-Media-Bibliothek entwerfen und für einfache Anfragen verwenden.

Unsere Bibliothek speichert zu jedem Musikstück den Titel, den Interpreten und die Dauer ab (in Sekunden). Weiterhin können Musikstücke zu Alben zusammengefasst sein. Es gibt außerdem noch mehrere Benutzer der Bibliothek, die die Stücke für sich persönlich bewerten können (als „gut" oder „schlecht").

1. Definieren Sie geeignete Datentypen und Typaliase für die Media-Bibliothek.
2. Definieren Sie die folgenden Operationen auf der Media-Bibliothek:
   - addAlbum : fügt ein Stück einem Album hinzu. (Ein möglicher Typ wäre: addAlbum :: Track −> Album −> MediaBib −> MediaBib)
   - rateTrack : bewertet einen Track für einen bestimmten Benutzer. (Ein möglicher Typ wäre: rateTrack :: User −> Track −> Rating −> MediaBib −> MediaBib)
3. Definieren Sie die folgenden Anfragen auf der Bibliothek:
   - Gebe alle Alben und ihre Dauer zurück.
   - Gebe für einen bestimmten Benutzer alle Alben zurück, bei denen >50% der Stücke positiv bewertet sind.

Um das Testen zu erleichtern finden Sie auf der Vorlesungshomepage eine Datei `Tracks.hs`, die Listen von Musik-Stücken im Format (Title, Artist, Duration) enthält. Diese können Sie dann in Ihre Datenmodell übersetzen.

Aufgabe 5 (Tic-Tac-Toe)
Tic-Tac-Toe (`http://en.wikipedia.org/wiki/Tic-tac-toe`) ist ein in der Informatik sehr beliebtes Spiel, vor allem wegen seiner Einfachheit (in jeglicher Hinsicht).

1. Entwerfen Sie ein Datenmodell für Tic-Tac-Toe.
2. Definieren Sie Funktionen, die feststellen, ob ein Spiel im Gange, gewonnen, oder ungültig ist. Wenn ein Spiel gewonnen ist, dann von wem?
3. Definieren Sie eine Funktion, die von einem gegebenen Spielstand ausgehend die möglichen Gewinnspielstände ausgibt.