Lehrstuhl für Softwaretechnik und Programmiersprachen
Bachelor-Seminar: Programmiersprachen
John Witulski

Heinrich Heine Universität Düsseldorf

Übungsaufgabe zu XXX

Aufgabe 1.1

Installieren Sie zunächst den Nix-Package-Manager.
Gehen Sie dafür auf die Seite `https://nixos.org/download.html#nix-install-linux` und folgen Sie den Anweisungen für eine "Multi-user installation".

Geben sie nun folgende Befehle in die Konsole ein:

```
wget https://raw.githubusercontent.com/Quoteme/\
bachelor_seminar_haskell/master/aufgaben/shell.nix

nix-shell
```

Aufgabe 1.2

Schreiben Sie eine Funktion `myMap :: (a -> b) -> [a] -> [b]`, die eine Funktion und eine Liste als Argumente nimmt und die Funktion auf jedes Element der Liste anwendet.
Verwenden Sie `myMap` um die Liste `[1,2,3,4,5]` in die Liste `[1,4,9,16,25]` zu transformieren.
Sie dürfen hierfür nicht die Funktion `map` verwenden.

Aufgabe 1.3

John arbeitet in einem Kleidungsladen. Er hat einen großen Stapel an Socken und möchte wissen, wie viele Paare er hat. Gegeben ist eine Liste von Integern, welche die Farbe der Socken darstellt, finden Sie die Anzahl der Paare.
Schreiben Sie eine Funktion `sockPairs :: [Int] -> Int`, die eine Liste von n Socken als Argument nimmt und die Anzahl der Paare zurückgibt.

Beispiel für $n = 7$ Socken

```
sockPairs [1,2,1,2,1,3,2] = 2
```

Hinweis: Importieren sie `Data.List` und verwenden Sie die Funktion `group` nach `sort` um die Liste zu gruppieren.

Hinweis: Die Funktionen `length`, `sum` und `div` könnten Ihnen hier weiterhelfen.

Aufgabe 1.4

1. Teilaufgabe 1
   Erstellen Sie einen neuen Datentypen `Vector3`, welcher aus drei Werten vom Typen `Integer` besteht und eine Instanz der Klasse `Show` und `Read` ist.
   Hinweis: Verwenden Sie `data` und `deriving` um die Instanzen zu erstellen.

2. Teilaufgabe 2
   Erstellen Sie eine Funktion `add :: (Vector3, Vector3) -> Vector3`, welche ein Tupel von Vektoren addiert.

3. Teilaufgabe 3
   Erstellen Sie eine Funktion `getVector3s :: IO ((Vector3, Vector3))`, welche einen Vector3 aus dem stdin einliest
   Hinweis: Verwenden Sie `read :: Read a => String -> a` aus der Klasse `Read`
   Hinweis: Sie können Funktionen `f :: Read a => b -> a` auf Intanzen `I` der Klasse `Read` durch `\x -> f x :: I` einschränken
   Hinweis: Verwenden Sie `(<$>) :: Functor f => (a -> b) -> f a -> f b` um die Funktionen `getVector3` mit `read`, zu verketten. **Hinweis:** Verwenden Sie `getLine` um eine Zeile aus dem stdin zu lesen.

4. Teilaufgabe 4
   Erstellen Sie eine Funktion `main :: IO ()`, welche zwei Vektoren einliest und die Summe ausgibt.