technische universität dortmund

FLDIT

Lehrstuhl 1
Logik in der Informatik

Prof. Dr. Peter Padawitz

Jos Kusiek (*jos.kusiek@tu-dortmund.de*) Wintersemester 2017/2018

Übungen zu Funktionaler Programmierung
Übungsblatt 12

Ausgabe: 19.2.2017, **Abgabe:** 26.2.2017 – 16:00 Uhr, **Block:** 6

Importieren Sie die Module `Examples`, `Coalg` und `Data.Array`.

Aufgabe 12.1 (3 Punkte) *Zustandsmonade*
Schreiben Sie folgende Funktionen.

a) `getX, getY :: State Point Float` – Liest die Koordinate x bzw. y eines Punktes aus.

b) `setX, setY :: Float -> State Point ()` – Setzt die Koordinate x bzw. y eines Punktes.

c) `rotate :: (Float,Float) -> Float -> State Point ()` – Eine zustandsbasierte Variante der Funktion `rotate` von Folie 29. Der erste Parameter gibt die Koordinaten an, um die gedreht werden soll. Der zweite Parameter gibt den Winkel an.
Beispiel: `runS (rotate (4,5) 180) $ Point 5 8` ⇝ `((),(3.0000002,2.0))`

Aufgabe 12.2 (4 Punkte) *IO-Monade*
Schreiben Sie eine Funktion `main :: IO ()`, welches eine Textdatei einliest und den Inhalt in Groß- oder Kleinbuchstaben in eine andere Datei schreibt. Das Programm soll interaktiv sein. Der Benutzer kann die Eingabedatei, die Ausgabedatei und die Art der Änderung bestimmen. Ein Ablauf soll wie folgt aussehen:

```
Eingabedatei: <Eingabe>
Ausgabedatei: <Eingabe>
Ändere Eingabe in...
1. Großbuchstaben
2. Kleinbuchstaben
<Eingabe>
```

Übersetzen Sie das Programm mit dem Befehl

```
ghc <filename>.hs
```

in eine ausführbare Datei und führen Sie diese aus.

Hinweis: Sie können die Funktionen `toUpper` und `toLower` für die Umwandlung benutzen. Importieren Sie dazu das Modul `Data.Char`.

Aufgabe 12.3 (2 Punkte) *Felder*
Gegeben sei die bekannte Funktion `credit`.

```
type ID = Int
type Bank = [(ID,Account)]
data Account = Account { balance :: Int, owner :: Client } deriving Show
data Client = Client
  { name :: String
  , surname :: String
  , address :: String
  } deriving Show

credit :: Int -> ID -> Bank -> Bank
credit amount id ls
  = updRel ls id entry{ balance = oldBalance + amount} where
    Just entry = lookup id ls
    oldBalance = balance entry
```

Ändern Sie das Typsynonym `Bank` in

```
type Bank = Array ID Account
```

und schreiben Sie die Funktion `credit` entsprechend um.

Aufgabe 12.4 (3 Punkte) *Dynamische Programmierung*
Gegeben sei folgende rekursive Berechnung des Binomialkoeffizienten:

```
bincoeff :: (Int,Int) -> Int
bincoeff(n,k)
  | k == 0 || k == n = 1
  | 0 < k, k < n = bincoeff(n-1,k-1) + bincoeff(n-1,k)
  | otherwise = 0
```

Definieren Sie eine Funktion `bincoeffDyn` vom Typ `(Int,Int) -> Int`, welche das Problem mithilfe der dynamischen Programmierung löst.

Hinweis: Tupel sind Instanzen der Klasse `Ix` und können daher auch zur Indizierung von Feldern genutzt werden.