HTWK Leipzig, Fakultät IM
Prof. Dr. Sibylle Schwarz sibylle.schwarz@htwk-leipzig.de

5.Übung zur Vorlesung „Fortgeschrittene Programmierung“
Sommersemester 2020 zu lösen bis 13. Mai 2020

Aufgabe 5.1 (Haskell-Typen)

Geben Sie alle Elemente dieser Datentypen an:

a. `Maybe ()`

b. `Maybe (Bool, Maybe ())`

c. `Either (Bool , Bool) (Maybe (Maybe Bool))`

Aufgabe 5.2 (Haskell-Standard-Typen)

Geben Sie zu jedem der folgenden Ausdrücke den Typ an:

a. `[ 'a', 'b', 'c' ]`

b. `( 'a', 'b', 'c' )`

c. `[ ( False, '0' ) , ( True, '1' ) ]`

d. `( [ False, True ], [ '0', '1' ] )`

e. `( "so", True, 2, ( 'a', [ 1 , 3, 41 ] ), 'c' )`

f. `Left ( 2, "foo", Nothing )`

Aufgabe 5.3 (Modellierung mit Haskell-Datentypen)

Ein Kaffeeautomat bietet Kaffe in zwei verschiedenen Größen zu verschiedene Preisen an und kann auf Wunsch noch eine oder mehrere Portionen Milch und / oder Zucker dazugeben. Jede Portion Zucker und jede Portion Milch kostet einen Aufpreis.

a. Geben Sie einen Haskell-Datentyp an, durch den genau alle auf diese Art zusammengestellten Getränke beschrieben sind.
Überprüfen Sie Ihre Antwort, indem Sie alle Elemente des Datentyps angeben.

b. Welches Element repräsentiert einen großen Kaffe mit Zucker und Milch?

c. Geben Sie eine Haskell-Funktion an, welche jedem Getränk seinen Preis zuordnet.

d. Der Automat gibt das gewählte Getränk erst aus, nachdem genügend Geld eingeworfen wurde. Geben Sie eine Haskell-Funktion an, welche zu jedem Getränk und dem gezahlten Betrag das Rückgeld berechnet.

e. Geben Sie einen geeigneten Haskell-Datentyp zur Verwaltung des Vorrates an Zucker- und Milch-Portionen im Automaten an.

Aufgabe 5.4 (Funktionen auf Listen)

Definieren Sie die folgenden Funktionen (Typdeklarationen nicht vergessen):

a. für Listen von Peano-Zahlen:

   (a) `succs`, welche jede eingegebene Liste von Peano-Zahlen auf die Liste der Nachfolger der Zahlen in der Liste abbildet,

   (b) `preds`, welche jede eingegebene Liste von Peano-Zahlen auf die Liste der Vorgänger der Zahlen in der Liste abbildet (dabei ist `Z` der Vorgänger von `Z`),

   (c) `list_sum`, welche jede eingegebene Liste von Peano-Zahlen auf die Summe aller Zahlen in der Liste abbildet,

b. für Int-Listen:

   (a) `squares`, welche jede eingegebene Liste von Zahlen auf die Liste aller Quadrate der Elemente der Eingabeliste abbildet,

   (b) `list_prod`, welche jede eingegebene Liste von Zahlen auf das Produkt aller Zahlen in der Liste abbildet,

   (c) `list_min`, welche jede eingegebene Liste von Zahlen auf die kleinste in der Liste vorkommende Zahl abbildet.

c. für Listen über beliebigem Typ:

   (a) `alle_gleich`, welche feststellt, ob eine Liste nur gleiche Elemente enthält.

   (b) `n_mal`, welches in einer eingegebenen Liste jedes Element durch eine eingegebene Anzahl von Kopien dieses Elementes ersetzt.
   Beispiel: `n_mal [1,3,2] 4 = [1,1,1,1,3,3,3,3,2,2,2,2]`

   (c) `n_kop`, welches zu jeder eingegebenen Liste die Verkettung einer eingegebenen Anzahl von Kopien dieser Liste berechnet.
   Beispiel: `n_kop [1,3,2] 4 = [1,3,2,1,3,2,1,3,2,1,3,2]`