Prof. Dr. Ralf Hinze
M.Sc. Sebastian Schweizer

TU Kaiserslautern
Fachbereich Informatik
AG Programmiersprachen

Functional Programming: Exercise 9

Sheet published: Thursday, June 27th
Submission deadline: Wednesday, July 3rd, 12:00 noon

Exercise 9.1 (Skeleton: `TypeQuiz.hs`)**.** Define a *total* function for each of the following types. You may assume that functions passed as arguments are total.

a) $(Functor\ f) \Rightarrow f\ (a, b) \to f\ (b, a)$

b) $(Functor\ f) \Rightarrow f\ (f\ (a, b)) \to f\ (f\ (a, b, b, a))$

c) $(Functor\ f, Functor\ g) \Rightarrow f\ a \to g\ b \to f\ (g\ (b, a))$

d) $(Applicative\ f) \Rightarrow f\ a \to f\ b \to f\ (a, b)$

e) $(Applicative\ f) \Rightarrow (a \to b \to c \to d) \to (f\ a \to f\ b \to f\ c \to f\ d)$

f) $(Applicative\ f, Applicative\ g) \Rightarrow a \to f\ (g\ a)$

g) $(Monad\ m) \Rightarrow a \to m\ (m\ a)$

h) $(Monad\ m) \Rightarrow m\ a \to m\ (m\ a)$

i) $(Monad\ m) \Rightarrow m\ (m\ (m\ a)) \to m\ a$

Exercise 9.2 (Skeleton: `Distribution.hs`)**.** Due to the current heat wave, Harry Hackers computer started showing unexpected behavior. An analysis yielded that the CPU sometimes fails adding two numbers: with a probability of 0.0001% the result of adding two numbers is 0, in the remaining cases it is the expected result. We can use the distribution monad from the lecture to represent multiple results with their probability.

a) Write a function $faultyAdd :: Integer \to Integer \to Dist\ Integer$ that implements the described faulty addition.

b) Consider the function to sum up the numbers from 0 to n:

$$\begin{aligned} &gauss :: Integer \to Integer \\ &gauss\ 0 = 0 \\ &gauss\ n = gauss\ (n-1) + n \end{aligned}$$

Transform this function such that is has the type $Integer \to Dist\ Integer$ and uses the faulty addition from (a).

c) Do the same for this Fibonacci function:

$$\begin{aligned} &fib :: Integer \to Integer \\ &fib\ 0 = 0 \\ &fib\ 1 = 1 \\ &fib\ n = fib\ (n-2) + fib\ (n-1) \end{aligned}$$

Exercise 9.3 (Skeleton: `Zeroless.hs`)**.** Reconsider the *Binary* type from the lecture (slide 464). The representation is not unique: we can add arbitrary many leading zeros. A unique representation is possible using zeroless binary numbers:

```
data Binary = Nil | One Binary | Two Binary

natural :: Binary → Integer
natural Nil       = 0
natural (One n)   = 1 + 2 * natural n
natural (Two n)   = 2 + 2 * natural n
```

The numbers $0..5$ are *Nil*, *One Nil*, *Two Nil*, *One* (*One Nil*), *Two* (*One Nil*), *One* (*Two Nil*). We can use this type for random-access lists as well:

```
data Sequ elem
  = Nil
  | One elem       (Sequ (Pair elem))
  | Two elem elem (Sequ (Pair elem))
```

a) Adapt the function $size :: Sequ\ elem \to Integer$ to zeroless binary numbers.

b) Adapt the function $cons :: elem \to Sequ\ elem \to Sequ\ elem$ to zeroless binary numbers.

c) Write a function $head :: Sequ\ elem \to elem$ that returns the first element from the given random-access list. What is the benefit of using zeroless binary numbers instead of the normal binary numbers from the lecture?

Exercise 9.4 (Lecture Evaluation - Not graded)**.** Please contribute to the lecture evaluation. Your feedback helps us to improve this course. If you have a `@cs.uni-kl.de` email address, then you already have received an email with access details. If you do not have such an email account, please go to `https://vlu.informatik.uni-kl.de/teilnahme/` and enter your RHRK email address (`username@rhrk.uni-kl.de`). You then will receive an email with access details for the lecture evaluation system.

Access the system using the received URL and then select your lectures. Our lecture is *INF-36-51-V-6: "Funktionale Programmierung"*. Afterwards, you can start answering the questions and give us comments. The evaluation is of course anonymous.

Thanks for taking your time!