Lab Sheet 7

User-defined Types

STS Software Technology Systems

Functional Programming - Winter 2018/2019 - December 6, 2018 - Schupp/Lehmann

How to succeed with the labs and exercises?

Labs and exercise sheets are published every week on the course homepage at StudIP. As described in the first lecture, each successfully completed lab and exercise earns you bonus points towards your final score in this semester's exam. Keep in mind that you only get bonus points if you would pass the exam without them. **Cheating does not help you - but we will!**

How to complete a lab successfully?

In the lab, you will solve the lab sheet with fellow student(s). To find out who your teammates are, please look at the group pdf in StudIP's download area. **During the session, one participant of every group will be selected, who must then explain one of their task solutions to a lab assistant**.

How to complete an exercise successfully?

In order to complete an exercise sheet successfully, you must upload your answers using DOMjudge **before the deadline** printed on the exercise sheet. We will not consider any solutions handed in after the deadline! Furthermore, you must solve and hand in the exercises **individually** and your Haskell code **must compile** and **pass certain amounts of tests** as specified. During the exercise session, we develop possible solutions together. Please participate! We encourage you to ask and answer questions from fellow students.

Technically, Haskell files you submit using DOMjudge must have the format as specified in the task sheets(usually ".hs", ".lhs", or ".txt"). Furthermore, DOMjudge will only consider your last submission. Therefore, if you first submit successfully (your code compiles and tests are passed) and afterwards unsuccessfully (your code does not compile or certain tests fail again), your last submission counts, and - if it does not compile - will therefore be ignored. Make sure your last submission was successfull!

How to get additional information?

We encourage you to discuss past and present exercise sheets with us. Either approach us during the exercise session, or visit us during the weekly office hours. We are also available via e-mail or on the StudIP forum. We try to reply as quick as possible and in general, you should get a reply the next weekday, but we cannot guarantee this.

Exercise 1 Try to load the following haskell code in (Win)GHCi.
You can find it in the file "Errors.hs" on StudIP.

```
type Event a = Lecture | Lab a | exercise deriving Show

type Professor = String
newtype Title = Title String
newtype Course a = Course Professor Title [Event a] deriving Show

type PathToFile = String
data EventDescription = Lab {
     topic :: String
    ,assistants :: [String]
    } deriving Show

isLab :: Event a -> Bool
isLab (Lab _ ) = True

giveMeAllLabs :: Course EventDescription -> [Event EventDescription]
giveMeAllLabs (Course es) = filter isLab es

labs = giveMeAllLabs (Course "Schupp" (Title "Functional␣Programming") [Lecture])
```

Fix all errors. Test that `labs` returns the expected result.

Exercise 2 Represent the following fictitious course description as a value of type `Course EventDescription` from Exercise 1.

Esoteric Programming Languages

In this Lecture, Prof. Brain and assistant Pinky explain programming languages of the future. The course consists of at least a lecture and an exercise. Furthermore, Pinky offers labs on Brainfuck[1] and Taxi[2]

Exercise 3 Consider the type definitions of Exercise 1. Write a function that takes a value of type `Course EventDescription` and returns the list of all lab topics. Test your implementation with your solution from Exercise 2.

Exercise 4 Follow the design recipe for data definitions from the lecture and design a type according to the following specification.

The type represents either an error, parameterized by an error id and an error description, or a non erroneous result of some kind. The error id must be an integral number and the description must be a string.

Exercise 5 * Consider your type definition of an error in exercise 4. Write a function `bindE` that takes a value `v` of the type you defined and a function `f`. `bindE` applies `f` to the wrapped value in `v` if it represents a non erroneous value. Otherwise it returns the original error. Explain how this function would be used.

[1] http://en.wikipedia.org/wiki/Brainfuck
[2] http://www.bigzaphod.org/taxi/