Functional Programming WS 2022 LVA 703025

Exercise Sheet 2, 10 points Deadline: Wednesday, October 19, 2022, 6am

- Mark your completed exercises in the OLAT course of the PS.
- You can use a template .hs file that is provided on the proseminar page.
- Upload your modified .hs file for Exercise 3 in OLAT.
- Your .hs file should be compilable with ghci.

Exercise 1 *Parsing expressions* 3 p.

Draw the abstract syntax trees of the following expressions:

1. `(59 + 25) / (9 - 7)` (1 point)
2. `(x && (a > (x + 3))) || (y >= 7)` (1 point)
3. `square velocity * (density * volume / 2)` (1 point)

Remarks:

- Mathematical operators bind in the same way as in typical mathematical notation. Operators associate to the left, except for exponentiation. For example, `1 - 2 + 3` means `(1 - 2) + 3`.
- Function applications (e.g. `square x`) bind stronger than operator applications (e.g. `3 * 4`).

Exercise 2 *Types* 4 p.

Give a suitable type for each of the following expressions. Justify your answers.
For example, the expression `1 + 2` can have the type `Integer`.

1. `3.14` (0.5 points)
2. `True` (0.5 points)
3. `3 - 4` (0.5 points)
4. `2 * 0.5` (0.5 points)
5. `head "Hello World"` (1 point)
6. `half` (1 point)

Remark: Here `half` is defined as `half x = x / 2`.

Exercise 3 *Datatype definition* **3 p.**

In this exercise you should model data relevant for a university course. You can use the Haskell template provided on the course website for this exercise.
Hint: To test your datatype definitions, it might be helpful to add some example values into your Haskell file.

1. A course needs to take place in a room. The properties of a room we are interested in are its name, the number of students which fit in the room, and whether the room is equipped with computers for students. Define a datatype in Haskell called `Room` to represent rooms. (1 point)

2. Courses also have a teacher and students. To model these, we can reuse the datatype `Person` from the lecture. We want to be able to add an arbitrary amount of students to a course. In order to achieve this, define a `PersonList` which can store any number of persons. (1 point)

3. Our courses are either a lecture or a lab. Define a datatype called `Course` in Haskell. It should have two constructors, one called `Lecture` and one called `Lab`. Both course types have a name, take place in a room and have a teacher and students. A lecture additionally has an exam date. Labs should store the number of exercise sheets. (1 point)