Programmierparadigmen und Compilerbau
3. Assignment Sheet

Sommersemester 2024
Prof. Visvanathan Ramesh
Gamze Akyol
Alperen Kantarci

Delivery date: 12.6.2024

Exercises for Assignment Sheet 3

Read the following information carefully!

- Group assignments are **not allowed!** However, the tasks are welcome to be solved in groups, only the tasks should be written down **individually**. Plagiarism will not be tolerated.
- Put all the files you want to submit into a single .zip file on Moodle.
- The file name should correspond to the pattern `YourName_YourGroupNumber_AssignmentSheetNumber.zip`.
- You should edit todo parts or add some code depending on the exercise for each file and keep the rest of the file as it is.

This exercise sheet is based on the Haskell MOOC course from University of Helsinki: "https://github.com/moocfi/haskell-mooc/".

Two files `Set5a.hs`, `Set5b.hs` are included in our Moodle page.

- Set5a.hs has exercise sections on implementing different mathematical functions. You are required to solve correctly at least 6 out of the 12 requested exercises to get full credit. Note that some programming tasks may have dependency on other tasks before and doing all the exercises is ideally recommended.
- Set5b.hs has exercise sections on lists and functional programming. You are required to solve correctly at least 5 out of the 10 requested exercises to get full credit. Note that some programming tasks may have dependency on other tasks before and doing all the exercises is ideally recommended.

In order to test your solutions, you need the run the following commands from your Docker environment:

```
stack build
stack runhaskell Set5aTest.hs
```

```
stack runhaskell Set5bTest.hs
```

You can find an example below to see how to edit and test the exercises:

Example:

```
-- Exercise set 5a
--
-- * defining algebraic datatypes
-- * recursive datatypes

module Set5a where

import Mooc.Todo

--
    ---------------------------------------------------------------------------

-- Ex 1: Define the type Vehicle that has four constructors:
     Bike,
-- Bus, Tram and Train.
--
-- The constructors don't need any fields.
-- You will write your code to empty lines below


--
    ---------------------------------------------------------------------------

-- Ex 3: Here's the definition for a datatype ShoppingEntry
    that
-- represents an entry in a shopping basket. It has an item
    name (a
-- String), an item price (a Double) and a count (an Int).
    You'll also
-- find two examples of ShoppingEntry values.
--
-- Implement the functions totalPrice and buyOneMore below.

totalPrice :: ShoppingEntry -> Double
totalPrice = todo
```

You will ONLY change the todo parts or write definitions to empty parts like in Set5a.hs Exercise 1 and then test your solution with test files: `stack runhaskell Set5aTest.hs`, and `stack runhaskell Set5bTest.hs`