Programmierparadigmen und Compilerbau
2. Assignment Sheet

Sommersemester 2024
Prof. Visvanathan Ramesh
Gamze Akyol
Alperen Kantarci

Delivery date: 29.5.2024

Exercises for Assignment Sheet 2

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 only todo parts of 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/".

Five files `Set2b.hs`, `Set3a.hs`, `Set3b.hs`, `Set4a.hs` and `Set4b.hs` are included in our Moodle page.

- Set2b.hs has exercise sections on implementing different mathematical functions. You are required to solve correctly at least 4 out of the 8 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.
- Set3a.hs has exercise sections on lists and functional programming. You are required to solve correctly at least 7 out of the 14 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.
- Set3b.hs is a special exercise set. The exercises are about implementing list functions using recursion and pattern matching, without using any standard library functions. For this reason, you'll be working in a limited

environment where almost none of the standard library is available. 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.

- Set4a.hs has exercise sections on using type classes and working with lists. 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.

- Set4b.hs has exercise sections on folds. You are required to solve correctly at least 4 out of the 7 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 Set2bTest.hs
stack runhaskell Set3aTest.hs
stack runhaskell Set3bTest.hs
stack runhaskell Set4aTest.hs
stack runhaskell Set4bTest.hs
```

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

Example:

```
module Set2b where

import Mooc.Todo

-- Some imports you'll need. Don't add other imports :)
import Data.List

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

-- Ex 1: compute binomial coefficients using recursion.
    Binomial
-- coefficients are defined by the following equations:
--
--   B(n,k) = B(n-1,k) + B(n-1,k-1)
--   B(n,0) = 1
--   B(0,k) = 0, when k>0
--
-- Hint! pattern matching is your friend.

binomial :: Integer -> Integer -> Integer
binomial = todo
```

You will ONLY change the todo part and then test your solution with test files: `stack runhaskell Set2bTest.hs`, `stack runhaskell Set3aTest.hs`, `stack runhaskell Set3bTest.hs`, `stack runhaskell Set4aTest.hs` and `stack runhaskell Set4bTest.hs`.