Functional Programming 1 — 1DL330
Assignment 1

Lab: Friday, 13 September

Submission Deadline: 18:00, Wednesday, 18 September, 2019

This assignment should be solved in **groups of two** students. Please join a group in the *Assignment 1* group division on the Student Portal. If you have not found a group partner by the start of the lab for this assignment, please inform the lab assistant. If you would prefer to solve this assignment individually due to special circumstances, please contact the course teacher, Tjark Weber.

Instructions

- Start by reading the General Lab Instructions (on the Student Portal).
- Download the file `Lab1.hs` from the Student Portal. Modify this file as needed to add your solutions. Do *not* rename the file.
- The file that you submit must be a valid Haskell file, so the answers to some of the questions need to be given as comments using −− ... or {− ... −}.
- Remember to follow our Coding Convention (also on the Student Portal), and provide a specification for every function that you write. Also provide a variant for every recursive function.
- Make sure that your solutions pass the tests in `Lab1Tests.hs` before you submit. A submission that does not pass these tests will get grade K. See the *Testing* section near the end of this assignment for details.

1 Reduction, Specification and Variant

Consider the following function declaration:

```haskell
product n =
  if n == 1 then
    1
  else
    n * product (n-1)
```

1. Give a detailed step-by-step evaluation of **product** 2, using the same style as in the lecture slides.

2. Write a specification for the function.
3. Give a variant for the function.

Hint: See the Coding Convention (on the Student Portal) and the lecture slides about specifications and variants.

2 Currying

Consider this function declaration:

```
minus x y = x − y
```

1. Write the function declaration as a value declaration minus = ... Your declaration should be equivalent to the declaration above. (Hint: see the lecture slides on infix operators or anonymous functions.)
2. Enter the declaration **let** foo = minus (5 :: **Integer**) 4 into GHCi. What is the type of foo? What happens when you evaluate foo?
3. Enter the declaration **let** bar = minus (5 :: **Integer**) into GHCi. What is the type of bar? What happens when you evaluate bar?
4. Give a step-by-step evaluation of minus 5 4.

3 Types

Give functions with the following types:

1. **Integer** −> **Integer**
2. **Integer** −> **Integer** −> **Integer**
3. **Integer** −> (**Integer**,**Integer**)
4. (**Integer**,**Integer**) −> **Integer**
5. **Integer** −> **Double** −> **String** −> **String**
6. **Integer** −> **String** −> **String** −> **Integer** −> (**Integer**, **String**)

In each case, try to find a function that is defined for all possible input values, and where the result depends on all parameters. Name your functions funN, where N is replaced by the type number (1–6) in the list above.

4 Sum-Square Difference[1]

The sum of the squares of the first ten natural numbers is

$$1^2 + 2^2 + \ldots + 10^2 = 385$$

The square of the sum of the first ten natural numbers is

$$(1 + 2 + \ldots + 10)^2 = 3025$$

Hence the difference between the square of the sum of the first ten natural numbers and the sum of the squares is $3025 - 385 = 2640$.

Declare a function `sumSquareDiff n` that returns the difference between the square of the sum and the sum of the squares of all natural numbers from 1 to n. (You must decide how to handle the case when $n < 1$.)

Testing

Use the test cases provided in `Lab1Tests.hs` (from the Student Portal) to test your solution.

1. Start GHCi in the same directory where you saved `Lab1.hs` and `Lab1Tests.hs`: e.g., `cd Lab1dir; ghci`
2. Enter `:l Lab1Tests` at the GHCi prompt.
3. Enter `runTests` at the GHCi prompt to run all tests.
4. Check the output for failing tests.
5. Quit GHCi (e.g., by typing Ctrl+D).
6. Fix all failing tests by modifying your `Lab1.hs` file.
7. Repeat steps 1.-6. until all tests succeed.

Good luck!

[1] This exercise was inspired by Problem 6 at Project Euler, a web site that offers hundreds of challenging programming problems.