universität innsbruck

Functional Programming | SS 2020 | LVA 703025

Exercise Sheet 14, 10 points | Deadline: **Friday, June 26, 2020, 12 noon**

- Please write all the Haskell code into a single .hs-file and upload it in OLAT.
- You can use the template .hs-files that are provided on the proseminar page[1].
- Your .hs-files should be compilable with ghci.
- Don't forget to mark your completed exercises in OLAT.
- Feel free to import functions from the Haskell standard library.

Exercise 14.1 *Partitioning of a list* **2 p.**

Write a function `parts :: [a] -> [[[a]]]` that returns all partitions of a list. A list `ys` is a partition of a list `xs` if all elements of `ys` are non-empty and the following property holds: `concat ys = xs`.
Examples:

```
parts [] = [[]]
parts [1] = [[[1]]]
parts [1,2] = [[[1,2]],[[1],[2]]]
parts "abc" = [["a","b","c"],["a","bc"],["ab","c"],["abc"]]
```

The order of the output does not matter.
Hint: If you are given all the partitions of `"c"`, how do you get all the partitions of `"bc"`? If you are given all the partitions of `"bc"`, how do you get all the partitions of `"abc"`?

Exercise 14.2 *Marking students* **8 p.**

You are given two lists, one with student names and student ids and one with student ids and the points students achieved on all exercise sheets.
Example list 1:

```
HASLBECK      101370
THIEMANN      202404
MUSTERMANN    444789
MUELLER       310131
```

Example list 2:

```
101370     10
202404      9
444789      7
310131     10
101370      8
202404     10
444789      2
310131      8
202404      3
```

[1] http://cl-informatik.uibk.ac.at/teaching/ss20/fp/index.php#exercises

List 1 contains a student name and their id on each line seperated by whitespaces. You can assume every name appears only once and each student has a unique id. List 2 contains a student id and points on each line separated by white space. Student ids can appear multiple times in this list.
For the exercise we define the following type synonyms in Haskell:

```
type Name = String
type Id = Integer
type Points = Integer
type Rank = Integer
```

In the following exercises you have to define functions to parse and analyze the two lists and produce pretty printed output. Each of the following exercises can be solved independently. You can assume that you only get well formed inputs for all functions. So you do not have to check for well formed input and can simply fail or produce incorrect results in that case.

1. Write a function `parseIds :: String -> [(Name, Id)]` which parses the list with the names and ids and returns a list of tuples with the names and ids.

   Expected output from example:

   ```
   [("HASLBECK", 101370), ("THIEMANN", 202404), ("MUSTERMANN", 444789), ("MUELLER", 310131)]
   ```

   (2 points)

2. Write a function `parsePoints :: String -> [(Id, Points)]` which parses the list with the ids and points and returns a list of tuples with the ids and the sum of all points for each id.

   Expected output from example:

   ```
   [(101370, 18), (202404, 22), (444789, 9), (310131, 18)]
   ```

   (2 points)

3. Write a function `rankNames :: [(Name, Id)] -> [(Id, Points)] -> [(Name, Points, Rank)]`. Each entry in its output should contain the name of one student, the sum of all of their points and their rank. A student has rank `n` if there are `n-1` students who have more points then them. So there can be multiple students who have the same rank.

   Expected output from example:

   ```
   [("HASLBECK", 18, 2), ("THIEMANN", 22, 1), ("MUSTERMANN", 9, 4), ("MUELLER", 18, 2)]
   ```

   (2 points)

4. Write a function `showList :: [(Name, Points, Rank)] -> String` that outputs a pretty printed version of the list with names, points and ranks.

   Expected output from example:

   ```
   Name          Points  Rank
   HASLBECK          18     2
   MUELLER           18     2
   MUSTERMANN         9     4
   THIEMANN          22     1
   ```

   The output should start with the heading. The column for the names should be 12 spaces wide, the column for the points 6 spaces and the column for the rank 4 spaces. Between each column should be one space. So the whole list is 24 spaces wide. The names should be left-aligned, the points and ranks right-aligned. Use `putStrLn` to print a string with newlines in `ghci`. `putStrLn` actually prints the newlines instead of outputting `\n`.

   (2 points)