universität innsbruck

Functional Programming	SS 2020	LVA 703025
Exercise Sheet 7, 10 points		Deadline: Monday, May 4, 2020, 10am

- Watch the lecture of week 7 and study part 5 of the slides up to slide no. 29.[1].
- Please write all the Haskell code into a single .hs-file and upload it in OLAT.
- You can use the template .hs-file that is provided on the proseminar page[2].
- Your .hs-file should be compilable with ghci.
- Don't forget to mark your completed exercises in OLAT.

Exercise 7.1 *Lists* **6 p.**

For this exercise you mainly use slides part 4 and part 5 up to slide 24 (e.g. `filter`, `map`, `reverse`, `sum`). Add type signatures to each function that are as general as possible (e.g. in exercises 7.1.4 - 7.1.6 the duration can be of any numeric type).

You are given a dataset of website visits. Here is an example of how such a dataset might look:

```
webData :: [String]
webData = ["Youtube", "Google", "Facebook", "Youtube", "Facebook",
           "Youtube", "Facebook", "Google", "Youtube"]
```

Each entry in the list represents a visit to a website. The goal is to create a ranking based on the number of times a website has been visited. For example, Google has been visited two times.

1. First you have to determine which websites occur in the dataset. Write a function `uniqueWebsites` that takes a dataset of the same form as `webData` and returns a list with the websites that occur in the dataset. For the example dataset a result could be:

   ```
   uniqueWebsites webData = ["Google", "Facebook", "Youtube"]
   ```

   (1 point)

2. The next step is to write a function `count` that takes the name of a website and a dataset and returns how often that website occurs in the dataset. For the example dataset and `website = "Google"` the result is: `count "Google" webData = 2`. (1 point)

3. The functions `uniqueWebsites` and `count` can now be combined to create a function that gives us a ranking of website visits. Write a function `result` that takes the dataset `webData` and returns an *ordered* list of tuples `[(count, website)]`, where the first tuple in the list is the website with the highest count and the last tuple the website with the lowest count (If two websites have equal count the ordering does not matter). For the example dataset the result is:

   ```
   result webData = [(4,"Youtube"),(3,"Facebook"),(2,"Google")]
   ```

   Hint: have a look at the functions `sortBy` and `compare`. (1 point)

[1] http://cl-informatik.uibk.ac.at/teaching/ws19/fp/slides/05x1.pdf
[2] http://cl-informatik.uibk.ac.at/teaching/ss20/fp/index.php#exercises

Now you are given a more detailed dataset that gives besides each website visit also the duration of that visit:

```
webDataDuration :: [(String, Double)]
webDataDuration = [("Youtube", 2.5), ("Google", 23.2), ("Facebook", 23.2),
                   ("Youtube", 3.4), ("Facebook", 4.5), ("Youtube", 34.5),
                   ("Facebook", 33.2), ("Google", 34.3), ("Youtube", 12.4)]
```

Each entry in the list is a tuple representing a visit to a website and the duration of that visit in seconds. This time you have to rank the website based on the total duration of time a website has been visited. For example, Google has been visited for 57.5 seconds.

4. Write a new function `uniqueWebsites2`, similar to `uniqueWebsites`, for a dataset of the same form as `webDataDuration` that returns a list of websites that occur in the dataset. (1 point)

5. Write a new function `count2`, similar to `count`, which takes a `website` and a dataset of the same form as `webDataDuration` as inputs and returns the total time `website` has been visited. (1 point)

6. Write a new function `result2` that is a higher-order function version of the original `result` function. `result2` can be used for both datasets and rank each of them accordingly. The function `result2` will take three inputs, a count function, a getWebsites function and a dataset. For the example datasets the results are:

```
result2 count uniqueWebsites webData =
  [(4,"Youtube"),(3,"Facebook"),(2,"Google")]
result2 count2 uniqueWebsites2 webDataDuration =
  [(60.9,"Facebook"),(57.5,"Google"),(52.8,"Youtube")]
```

(1 point)

Exercise 7.2 *Non-Recursive Data-Types* 4 p.

1. Define two non-recursive datatypes `Polar` and `Cart` for coordinates in the polar coordinate system [3] and the cartesian coordinate system [4]. Think about choosing useful type synonyms for the components of our coordinates.

   Write a function `createPolar` and its type signature, which takes a radius and an angle in degrees and returns a polar coordinate with the angle in radians. (1 point)

2. Implement the functions `cart2Tuple` and `polar2Tuple`, that convert a `Cart` into a tuple and a `Polar` into a tuple. Add the corresponding type signature. (1 point)

3. Define two conversion functions `polar2Cart :: Polar -> Cart` and `cart2Polar :: Cart -> Polar` between the coordinate systems above.

   To convert the polar coordinates (r, φ) to cartesian coordinates (x, y) use:

   $$x = r \cdot \cos\varphi \qquad\qquad y = r \cdot \sin\varphi$$

   The cartesian coordinates (x, y) can be converted into the polar coordinates (r, φ) as follows:

   $$r = \sqrt{x^2 + y^2} \qquad\qquad \varphi = \begin{cases} \arccos(\frac{x}{r}) & \text{if } y \geq 0 \text{ and } r \neq 0 \\ -\arccos(\frac{x}{r}) & \text{if } y < 0 \\ d & \text{if } r = 0 \end{cases}$$

   where you should define a sensible value for d.

   Think about the mathematical definitions above and find suitable implementations in Haskell. For the calculation of φ use guarded equations.

   Some useful functions: `cos`, `acos`, `sin`, `sqrt`, `round`, `pi`.

   Hint: Be aware that all trigonometric functions work with radians. (2 points)

[3] https://en.wikipedia.org/wiki/Polar_coordinate_system
[4] https://en.wikipedia.org/wiki/Cartesian_coordinate_system