Prof. Dr. Peter Thiemann
Janek Spaderna
janek.spaderna@pluto.uni-freiburg.de

Winter Term 2022/23
Issued 2022-10-28
Discussion on 2022-11-04

Functional Programming

http://proglang.informatik.uni-freiburg.de/teaching/functional-programming/2022/

Exercise Sheet 2

Exercise 1 (List functions II—deduplication)

Define and test a function `undup`, which keeps only the first occurrence of each element in a given list. What is the run time of `undup`?

Notes:

- The run time of `undup` is the reason why you should always try to avoid `Data.List.nub`, which is a predefined version of your `undup`.
- If you get type errors when testing properties with QuickCheck, first try to add a type signature to the properties.

Exercise 2 (Smallest factor)

Given two natural numbers $k \geq 2, n \geq 2$ we call k a *factor* of n iff there exist another natural number m such that $n = k \cdot m$. Define a function that computes the smallest factor of a given `Integer`.

Try to solve the task in two different ways and test them against each other. If your functions are only partially defined you can make use of QuickCheck's implication operator `(==>)`[1]:

```
>>> prop_div1 x = 100 `div` x <= 100
>>> prop_div2 x = (x /= 0) ==> (100 `div` x <= 100)
>>> quickCheck prop_div1
*** Failed! Exception: 'divide by zero' (after 1 test):
0
>>> quickCheck prop_div2
+++ OK, passed 100 tests; 16 discarded.
```

(`div` is integer division.)

Exercise 3 (Media Library)

A major part of music streaming services, such as Spotify, Apple Music, or Tidal, are the huge databases organizing and storing not only the songs itself but also the surrounding metadata—for example popularity, listening preferences, play counts etc. The goal of this exercise is to design the data model for a mini-media library and implement simple queries.

Our library saves the title, the artist and the duration (in seconds) for each song. Songs, or tracks, are uniquely identified by an ID. Furthermore, tracks can be combined into albums identified by their name. Additionally, each user of the library can rate songs either "good" or "bad".

1. Define appropriate data types and type aliases for the media library.

   Note: Consider using custom data types instead of tuples with more than two components. Definitely avoid tuples with more than three components.

[1] https://hackage.haskell.org/package/QuickCheck-2.14.2/docs/Test-QuickCheck.html#v:-61--61--62-

2. Define the following operations on the media library:
   - `addAlbum`: adds a song to an album. One possible type would be:
     `addAlbum :: TrackId -> AlbumName -> MediaBib -> MediaBib`
   - `rateTrack`: rate a song for a particular user. One possible type would be:
     `rateTrack :: User -> TrackId -> Rating -> MediaBib -> MediaBib`
3. To make testing easier, you can find a file `Tracks.hs` containing lists of songs on the lecture home page. Inspect the file and the used data format. Write a function to build a media library from a `TrackList`.
4. Define the following queries on the library and test them against the data from `Track.hs`:
   - Retrieving the titles of all albums and the duration of the contained songs.
   - For a particular user, return all albums where more than 50 % of the contained tracks are rated "good".

Exercise 4 (Tic-Tac-Toe)

Tic-Tac-Toe is a popular game between young children and in computer science, mainly because of its simplicity.

1. Design a data model for Tic-Tac-Toe.
2. Define functions that determine whether a game is in progress, won, or a draw. If a game is won, then by whom?