Exercises, Week 39 (22-28 Sep, 2025)

DM580: Functional Programming, SDU

Learning Objectives

- Write simple functions in Haskell.
- Debug simple functions in Haskell.
- Use library functions from Haskell's `Prelude`.
- Prove simple properties of Haskell programs.

Product Function (1.7.3 from the Book)

Define a function `product` that produces the product of a list of numbers.

Using your definition, prove that `product [2,3,4] = 24`.

Reversing the QuickSort Function (1.7.4 from the Book)

Consider the QuickSort function `qsort` given in chapter 1.5 of the book.

How should the definition of the function `qsort` be modified so that it produces a reverse sorted version of a list?

Modifying the QuickSort Function (1.7.5 from the Book)

What would be the effect of replacing $\leqslant$ by $<$ in the original definition of `qsort`? Hint: consider the example `qsort [2,2,3,1,1]`.

Fixing Errors (2.7.3 from the Book)

The script below contains three syntactic errors. Correct these errors and then check that your script works properly using GHCi.

```
N = a `div' length xs
  where
    a = 10
   xs = [1,2,3,4,5]
```

The Last Function (2.7.4 from the Book)

The library function `last` selects the last element of a non-empty list; for example, `last [1,2,3,4,5] = 5`. Show how the function `last` could be defined in terms of the other library functions introduced in Chapter 2. Can you think of another possible definition?

Leap Year Calculation

Write a function `isLeapYear` that takes a number (`Int`) as input and returns a Boolean (`Bool`) as output; i.e.:

```
isLeapYear :: Int -> Bool
```

The function should return `True` if the input number is divisible by 400, or by 4 but not 100. Otherwise, it should return `False`.

`https://en.wikipedia.org/wiki/Leap_year`

Days/Hours/Minutes Normalization

Write a function `normalizeDHM` which takes as input a triple of integers, representing (from left-to-right) days, hours, and minutes.

```
normalizeDHM :: (Int, Int, Int) -> (Int, Int, Int)
```

The input triple may contain integers may have a number of minutes that is >60, and a number of days that is >24. It should return as output a triple where minutes <60, and hours <24. The output triple should correspond to the same amount of time as the input triple.

For example `normalizeDHM (0,0,60) = (0,1,0)` and `normalizeDHM (0,24,0) = (1,0,0)`.

Debugging Insertion Sort

Consider the intentionally buggy *insertion sort* function in Listing 1.

Any function `f :: [Int] → [Int]` that implements a sorting function, must satisfy the proposition that, for any list `xs`, `(isSorted (f xs) && isPermutationOf xs (f xs)) = True`, where

```
isSorted []       = True
isSorted [_]      = True
isSorted (x:y:xs) = x ⩽ y && isSorted (y:xs)

isPermutationOf xs ys = null (xs \\ ys) && null (ys \\ xs)
  -- `\\` is list difference from `Data.List`.
  -- https://hackage.haskell.org/package/base/docs/Data-List.html#v:-92--92-
```

Prove that `almostSort` does *not* implement a sorting function. (Hint: use a counter-example to disprove the proposition.)

```haskell
almostSort :: [Int] -> [Int]
almostSort []     = []
almostSort (x:y:xs)
  | x > y = y : x : almostSort xs
almostSort (x:xs) = almostInsert x (almostSort xs)

almostInsert :: Int -> [Int] -> [Int]
almostInsert x [] = [x]
almostInsert x (y:ys)
  | x ⩽ y    = x : y : ys
  | otherwise = y : almostInsert x ys
```

Listing 1: Intentionally buggy *insertion sort* function