DM552 Exercises 3

Department of Mathematics and Computer Science
University of Southern Denmark

September 13, 2017

1. In the prelude, there is a function

   $$replicate :: Int \to a \to [a]$$

   which takes a number n and an element x, and returns a list with x repeated n times. E.g. $replicate\ 5\ 4 \equiv [4,4,4,4,4]$. Suggest how such a function can be implemented.

2. Give source code for a function

   $$zipIdx :: [a] \to [(a, Int)]$$

   which returns a new list with pairs of elements and indexes of the original list. E.g. $zipIdx$ `"abc"` $\equiv [('a',0),('b',1),('c',2)]$.

3. Give source code for a function

   $$setIdx :: [a] \to Int \to a \to [a]$$

   which overwrites the value at a specific position of the given list, if the given index exists. E.g.: $setIdx\ [1,3,5]\ 2\ 0 \equiv [1,3,0]$

4. Give source code for a function

   $$modIdx :: [a] \to Int \to (a \to a) \to [a]$$

   which modifies the value at a specific position of the given list by applying a function to the current value, if the given index exists. E.g.: $modIdx\ [1,3,5]\ 2\ (*2) \equiv [1,3,10]$

5. In the module *Data.List* there exists a function to remove duplicates of a list, with the type signature

   $$nub :: Eq\ a \Rightarrow [a] \rightarrow [a]$$

   Suggest source code for such a function, using list comprehensions. What is the time complexity of your function?

   Is it possible to define an alternative version without the type constraint *Eq a*?

   $$nub' :: [a] \rightarrow [a]$$

   What about

   $$nub'' :: Ord\ a \Rightarrow [a] \rightarrow [a]$$

   Ord a is a subclass of *Eq a*, so your definition from earlier can be used. But can you make a new definition with better time complexity, given the new type constraint?

6. This exercise is about making a function which the number of occurrences for each element of a list. Give definitions for

   $$elemCountsEq :: Eq\ a \Rightarrow [a] \rightarrow [(a, Int)]$$
   $$elemCountsOrd :: Ord\ a \Rightarrow [a] \rightarrow [(a, Int)]$$

   with optimal time complexities. Example: *elemCountsEq* `"asdqweasd"` = $[(\texttt{'a'}, 2), (\texttt{'s'}, 2), (\texttt{'d'}, 2), (\texttt{'q'}, 1), (\texttt{'w'}, 1), (\texttt{'e'}, 1)]$

7. Suggest a definition for a function which returns a list with specified element counts:

   $$fromElemCounts :: [(a, Int)] \rightarrow [a]$$

   Example: *fromElemCounts* $[(\texttt{'a'}, 2), (\texttt{'s'}, 2), (\texttt{'d'}, 2), (\texttt{'q'}, 1), (\texttt{'w'}, 1), (\texttt{'e'}, 1)]$ = `"aassddqwe"`

8. **Probability Distributions**

   In this exercise, we introduce a type signature

   $$\textbf{type}\ Dist\ a = [(a, Double)]$$

   to represent a finite, discrete probability distribution.

A fair coin toss would be represented by

$$[(\mathit{False}, 0.5), (\mathit{True}, 0.5)] :: \mathit{Dist}\ \mathit{Bool}$$

and a fair dice roll could be represented by

$$[(1, 0.166), (2, 0.166), (3, 0.166), (4, 0.166), (5, 0.166), (6, 0.166)] :: \mathit{Dist}\ \mathit{Int}$$

(a) Give definition of a function which assigns the same probability to each element of a list

$$\mathit{uniformly} :: [a] \to \mathit{Dist}\ a$$

Example: $\mathit{uniformly}\ [1, 1, 2, 3] = [(1, 0.25), (1, 0.25), (2, 0.25), (3, 0.25)]$

(b) Give definition of a similar function, which only mentions each occurrence of an element once

$$\mathit{uniformlyEq} :: \mathit{Eq}\ a \Rightarrow [a] \to \mathit{Dist}\ a$$
$$\mathit{uniformlyOrd} :: \mathit{Ord}\ a \Rightarrow [a] \to \mathit{Dist}\ a$$

Example: $\mathit{uniformlyEq}\ [1, 1, 2, 3] = [(1, 0.5), (2, 0.25), (3, 0.25)]$

(c) Give definition of functions

$$\mathit{joinDists} :: \mathit{Dist}\ a \to \mathit{Dist}\ b \to \mathit{Dist}\ (a, b)$$
$$\mathit{flattenDist} :: \mathit{Dist}\ (\mathit{Dist}\ a) \to \mathit{Dist}\ a$$

which return valid probability distributions (sum of probability weights are equal to 1).

Explain what your functions do, and what they can be used for.