Prof. Dr. Ralf Hinze
M.Sc. Sebastian Schweizer

TU Kaiserslautern
Fachbereich Informatik
AG Programmiersprachen

Functional Programming: Exercise 3

Sheet published: Thursday, May 2nd
Submission deadline: Wednesday, May 8th, 12:00 noon

Exercise 3.1 (Skeleton: `Lists.hs`)**.** Solve the following tasks using the *List Design Pattern*, i.e. write recursive functions and do not use any library functions.

a) Write a function $prod :: [Integer] \to Integer$ that computes the product of a list of numbers.

b) Write a function $contains :: Integer \to [Integer] \to Bool$ that checks wether the given number is contained in the list.

c) Write a function $nth :: Integer \to [a] \to Maybe\ a$ that takes an index n and a list of elements and returns the nth element of the list (if the index is valid) or $Nothing$ if the index is invalid.

d) Write a function $remove :: Integer \to [Integer] \to [Integer]$ that takes a number x and a list of numbers and returns the list where all occurences of x are removed.

e) Write a function $suffixes :: [a] \to [[a]]$ that takes a list and returns the list of all suffixes, starting with the list itself and ending with an empty list. For example: $suffixes\ [1,2,3] = [[1,2,3],[2,3],[3],[]]$.

Exercise 3.2 (Skeleton: `Comprehensions.hs`)**.** Solve the following tasks using the *List Comprehensions*, do not use *map*, *filter* or *concat*.

a) Write a function $f :: [(Bool, a)] \to [a]$ that takes a lists of pairs and returns only those elements where the first component is $True$.

b) Write a function $g :: [(a, b)] \to [(b, a)]$ that swaps the all pairs in the given list.

c) Write a function $h :: [(Integer, a)] \to [a]$ that takes a list of components (n, x) and returns a list where each element x is repeated n times. For example: $h\ [(3, \texttt{'a'}), (2, \texttt{'b'}), (4, \texttt{'z'})] = \texttt{"aaabbzzzz"}$. Do not use *replicate*, think of another way to repeat the elements.

Exercise 3.3 (Skeleton: `HigherOrder.hs`)**.** Solve the previous task without list comprehensions, instead use the functions *map*, *filter* and *concat*.