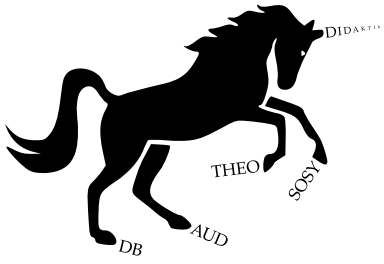
Aufgabe

(*QuickSort*)

Stichwörter: Funktionale Programmierung mit Haskell

Implementiere ebenfalls in der Datei Sortierverfahren.hs die Funktion quickSort, die den Quicksort-Algorithmus umsetzt. Sie erhält eine unsortierte Liste mit Elementen (auf denen eine Ordnung definiert ist), sortiert diese in aufsteigender Reihenfolge und gibt die sortierte Liste zurück. Die Signatur der Sortierfunktion lautet: `quickSort :: (Ord a) => [a] -> [a]`

Falls eine leere Liste übergeben wird, ist das Ergebnis die leere Liste. Ansonsten wird das erste Element der Liste als Pivot-Element gewählt. Es soll nun für zwei Teillisten (die mittels Listengenerator erzeugt werden sollen und eine Liste alle Elemente kleiner gleich dem Pivot-Element und die andere Liste alle Elemente größer dem Pivot-Element enthält) die Funktion `quickSort` (Rekursion!) aufgerufen werden. Dabei sollen die beiden Teillisten mit dem Pivot-Element in der Mitte konkateniert werden. (Tipp: Diese Funktion kann (ausgenommen von der Signatur und der Abbruchbedingung der Rekursion) als Einzeiler programmiert werden.)



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Module/60_FUMUP/30_Funktionale-Programmierung/Aufgabe_QuickSort.tex`