

## Programmierung

---

### Aufgabe 1 (AGS 12.1.55)

- (a) Schreiben Sie eine Funktion `pack :: [Char] -> [[Char]]`, welche in einer Liste aufeinander folgende Wiederholungen des gleichen Werts in einer Teilliste zusammenfasst.  
Z.B.: `pack ['a','a','b','b','b','a'] = [['a','a'], ['b','b','b'], ['a']]`.
- (b) Schreiben Sie eine Funktion `encode :: [Char] -> [(Int, Char)]`, welche eine Liste lauffängerkodiert.  
Z.B.: `encode ['a','a','b','b','b','a'] = [(2, 'a'), (3, 'b'), (1, 'a')]`.
- (c) Schreiben Sie eine Funktion `decode :: [(Int, Char)] -> [Char]`, welche eine lauffängerkodierte Liste wieder dekodiert.  
Z.B.: `decode [(2, 'a'), (3, 'b'), (1, 'a')] = ['a','a','b','b','b','a']`.
- (d) Schreiben Sie eine Funktion `rotate :: [Int] -> Int -> [Int]`, so dass `rotate xs n` die Liste `xs` um `n` nach links rotiert.  
Z.B.: `rotate [1,2,3,4] 1 = [2,3,4,1]` oder `rotate [1,2,3] (-1) = [3,1,2]`.

### Aufgabe 2 (AGS 12.1.53)

- (a) Schreiben Sie eine Funktion `unwords :: [String] -> String`, welche eine Liste von Wörtern aneinanderfügt. Die einzelnen Wörter sollen dabei durch ein Leerzeichen voneinander getrennt werden.
- (b) Schreiben Sie eine Funktion `words :: String -> [String]`, welche den Eingabestring in seine Teilwörter zerlegt. Nehmen Sie dabei zur Vereinfachung an, dass Wörter nur durch Leerzeichen voneinander getrennt sind.

Hinweise:

- `String = [Char]`.
- `words` und `unwords` sind in der Prelude-Library bereits definiert. Schließen Sie also diese Funktionen beim Einbinden der Prelude mittels der Direktive `hiding` aus.

### Aufgabe 3 (AGS 12.1.11)

Gegeben sei eine Liste der Bauart  $[l_1, l_2, \dots, l_n]$  mit  $l_1, l_2, \dots, l_n$  jeweils vom Typ `[Int]`. Es soll von jeder Liste  $l$  dieses Typs die Länge der längsten Liste, also  $\max \{\text{length}(l_i) \mid 1 \leq i \leq n\}$ , berechnet werden.

- (a) Geben Sie ein Beispiel für eine Liste dieses Listentyps an und nennen Sie das zugehörige Ergebnis.
- (b) Schreiben Sie in Haskell eine Funktion `max_length :: [[Int]] -> Int`, die diese Aufgabe erfüllt und werten Sie abschließend einen Funktionsaufruf aus.

*Hilfestellung:* Ermitteln sie zunächst die Liste der Listenlängen.

### Zusatzaufgabe 1 (AGS 12.1.45 b–c)

- (a) Welches Problem fällt Ihnen bei [Int] auf, wenn wir das Laufzeitverhalten beim Einfügen von Elementen am Anfang bzw. am Ende der Datenstruktur betrachten?
- (b) Implementieren Sie nun mit den Listen von Haskell einen *Queue*-Datentyp. Eine Queue enthält eine Folge von Werten (bei uns: Ints) und bietet darüber hinaus Funktionen
- zum Prüfen ob die Queue leer ist:  
`isEmpty :: Queue -> Bool,`
  - zum effizienten Einfügen eines Elements ans *Ende* der Queue:  
`enqueue :: Int -> Queue -> Queue`
  - zum effizienten Zugriff auf das *Anfangselement* der Queue:  
`first :: Queue -> Int`
  - zum Abspalten der Rest-Queue vom ersten Element:  
`rest :: Queue -> Queue`

Sie dürfen dabei die Funktion `reverse` zum Umkehren von Listen verwenden!