

Programmierung

Der iFSR sucht noch helfende Hände für die Erstsemestereinführung (ESE) in der Woche vom 1.10. bis 5.10.2018. Wendet euch bei Interesse oder Fragen bitte an esetutor@ifsr.de.

Aufgabe 1 (AGS 12.1.5)

- (a) Schreiben Sie eine Funktion `fac :: Int -> Int`, so dass `fac n` die Fakultät von n , also $n! = \prod_{i=1}^n i$, berechnet.
- (b) Schreiben Sie nun eine Funktion `sumFacs :: Int -> Int -> Int`, so dass `sumFacs n m` den Wert $\sum_{i=n}^m i!$ berechnet.

Aufgabe 2 (AGS 12.1.6)

Die Folge der Fibonacci-Zahlen f_0, f_1, \dots , ist definiert durch $f_0 = 1, f_1 = 1$, und $f_{i+2} = f_i + f_{i+1}$ für jedes $i \in \mathbb{N}$. Implementieren Sie eine Funktion, die für die Eingabe i die Zahl f_i berechnet.

Aufgabe 3 (AGS 12.1.7)

Schreiben Sie die folgenden Haskell-Funktionen:

- (a) `prod :: [Int] -> Int`, welches die Zahlen in einer Liste aufmultipliziert.
- (b) `rev :: [Int] -> [Int]`, welches eine Liste umkehrt.
- (c) `rem :: Int -> [Int] -> [Int]`, so dass `rem x xs` die Liste ist, die aus `xs` hervorgeht, indem alle Vorkommen von `x` gelöscht werden.
- (d) `isOrd :: [Int] -> Bool`, welches für eine Liste prüft, ob sie aufsteigend sortiert ist.
- (e) `merge :: [Int] -> [Int] -> [Int]`, welches zwei aufsteigend sortierte Listen zu einer aufsteigend sortierten Liste vereinigt.

Zusatzaufgabe 1 (AGS 12.1.8)

Implementieren Sie die (unendliche) Liste `fibs :: [Int]` der Fibonacci-Zahlen f_0, f_1, \dots

Zusatzaufgabe 2 (AGS 12.1.9)

In einem vollen Binärbaum ist jeder Knoten entweder ein Blatt, oder er hat zwei Kindknoten. Implementieren Sie eine Haskell-Funktion, welche für $n \in \mathbb{N}$ die Anzahl der vollen Binärbäume mit Knotenzahl n berechnet.

Zusatzaufgabe 3 (AGS 12.1.10)

Machen Sie sich mit `ghc(i)` (dem Glasgow Haskell Compiler, <https://www.haskell.org/ghc>) vertraut, insbesondere mit den Befehlen `:type`, `:info`, `:browse` und `?:`, und mit der Suchmaschine <https://haskell.org/hoogle>.