(FP-Aufgaben 06)

Aufgabe 1: Verwenden Sie die Funktion foldr, um die Funktion map' zu definieren:

```
map' :: (a -> b) -> [a] -> [b]
map' _ [ ] = [ ]
map' f (x:xs) = f x : map' f xs
```

Aufgabe 2:

a) Implementieren Sie eine rekursive Funktion take_while, die ein Prädikat und eine Liste als Parameter hat und das längste Anfangsstück der Liste zurückgibt, dessen Elemente alle das Prädikat erfüllen.
   Zum Beispiel soll der Aufruf take_While (\x -> x < 3) [1,2,1,1,4,2,5] zu [1,2,1,1] ausgewertet werden.
b) Schreiben Sie das Programm mittels foldr um.

Aufgabe 3: Schreiben Sie ein Programm, das alle Leerzeichen aus einem Text löscht und den ersten Buchstaben jedes Wortes in einen Großbuchstaben umwandelt (bzw. einen Kameltext erstellen).
z.B. wäre der Text „Das ist ein TEXT“, soll die Ausgabe der Funktion „DasIstEinText“ sein.

Hinweis:

- Eine Zeichenkette vom Typ String ist eine Liste von Zeichen (Also [Char]).
  z.B. “abc“ und [‘a‘,‘b‘,‘c‘] sind äquivalent.
- Importieren Sie *Data.Char* in Ihrem Programm. Für die Manipulation von Zeichen stehen die folgenden Funktionen in der Bibliothek *Data.Char* zur Verfügung:
  isLower, isUpper, toLower, toUpper, …
- Sie können die Funktion words verwenden:
  words :: String -> [String]
- Üben Sie mit “map” und “foldr”.

Aufgabe 4: Schreiben Sie ein Programm, das die Klammern-Strukturen in einem Text kontrolliert.
z. B. Wäre der Text=“(summe (x – 1) (y + 4)) (x-1)“ , dann muss Ihre Funktion ein True liefern. Wäre aber der Text=“summe (x – 1) (y + 4))) (x-1“, dann sind die Klammern nicht korrekt und soll die Funktion ein False ausgeben.

Hinweis:

- Üben Sie mit Akkumulator-Technik.