Arbeitsblatt: Higher Order Functions I

In Haskell gibt es keine Schleifen (while, for). In der Lerneinheit 6 haben Sie Rekursion als elegante Alternative kennen gelernt. Sie haben bereits einige rekursive Funktionen selbst definiert. In diesem Arbeitsblatt geht es darum wiederkehrende Rekursionsmuster in wiederverwendbare Funktionen zu giessen.

Sie haben diverse Funktionen geschrieben, die Elemente von Listen transformieren. Als Beispiel diene die Funktion squares, sie quadriert jedes Listenelement:

```
squares :: [Int] -> [Int]
squares []     = []
squares (i:is) = i^2 : squares is
```

Ein weiteres Beispiel ist die Funktion emails. Sie extrahiert aus seiner Liste von Studierenden die Liste deren Emailadressen:

```
data Student = Student { email :: String, grade :: Float }

emails :: [Student] -> [String]
emails []     = []
emails (s:ss) = (email s) : emails ss
```

a) Vergleichen Sie die zwei Funktionen squares und emails. Markieren Sie im Code der beiden Funktionen die Gemeinsamkeiten und in welchen Teilen sie sich unterscheiden.
Hinweis: Die Namen der Funktionen sind nicht relevant, es geht um die Struktur der Funktionen.

b) Definieren Sie die Funktion transform. Die Funktion soll den Teil, der bei squares und emails gleich ist implementieren und den Aspekt worin sich die Funktionen unterscheiden als Argument nehmen.
Hinweis: Schreiben Sie zuerst die Typsignatur der Funktion.

c) Implementieren Sie die Funktionen squares und emails basierend auf Ihrer neuen Funktion transform: