Übungsaufgabe 2

Ziel

Die zweite Übungsaufgabe zielt darauf ab, sich mit Operationen wie Filtern, Faltung, Mapping, Zipping und der Verkettung von Operationen vertraut zu machen.

Alle Lösungen zu diesen Aufgaben lassen sich leicht durch Recherche auf Stackoverflow finden, dies ist aber nicht Sinn der Übung. Es geht darum zu üben Probleme funktional zu lösen.

Allgemein

In dieser Übungsaufgabe werden zwei einfache Verschlüsselungsalgorithmen implementiert. Beide Algorithmen sollen ausschließlich auf den 26 Buchstaben des Alphabets operieren und hierbei Groß- und Kleinschreibung beachten. Satz- und Sonderzeichen sollen immer ignoriert werden. Leerzeichen sollen aus der Eingabe entfern werden und im Verschlüsselten Text sollen die Zeichen in Fünfergruppen dargestellt werden.

Ein Char in Haskell implementiert die Typklasse Enum. Enum erlaubt es wiederum einen Wert des Typs in einen Int-Wert umzuwandeln. Mit der Funktion fromEnum wird ein Char in einen Int umgewandelt und mit toEnum von einem Int in ein Char.

Erlaubt sind in dieser Aufgabe alle Sprachkonstrukte aus die auch in Übung 1 verwendet wurden und zusätzlich fromEnum, toEnum, map, filter, foldr, foldr, zip, $ und den . Operator.

Cäsar-Chiffre

Beim Cäsar-Chiffre werden die Buchstaben des zu verschlüsselnden Textes durch Buchstaben mit einem Festen abstand ersetzt. Die Schlüssel ist somit eine Ganzzahl. Beispiel: Ist der Schlüssel fünf und der Buchstabe im Text ein „A“, dann wird dieser in ein „F“ geändert. Aus dem „x“ wird in diesem Fall ein „c“. Man verschiebt also das Alphabet.

Implementiere die folgenden beiden Funktionen

```
encryptCaesar :: Int -> [Char] -> [Char]
decryptCaesar :: Int -> [Char] -> [Char]
```

Beim ersten Parameter handelt sich hier um den Schlüssel, beim zweiten Parameter um die zu verschlüsselnde bzw. entschlüsselnde Nachricht.

Teste folgendes Beispiel:

```
> encryptCaesar 23 "Haskell ist gar nicht so schwer!"
"Exphb iifpq dxokf zeqpl pzetb o!"
> decryptCaesar 23 "Exphb iifpq dxokf zeqpl pzetb o!"
"Haskellistgarnichtsoschwer!"
```

Vigenère-Chiffre

Recherchiere wie die Vigenère-Chiffre funktioniert. Implementiere anschließend die folgenden beiden Funktionen:

```
encryptVigenere :: [Char] -> [Char] -> [Char]
decryptVigenere :: [Char] -> [Char] -> [Char]
```

Teste die Implementierung mit folgendem Beispiel:

> encryptVigenere "beuthhochschule" "Haskell ist gar nicht so schwer!"
"Iemdl szkzl ihlym dlnlv zqjdw t!"
> decryptVigenere "beuthhochschule" "Iemdl szkzl ihlym dlnlv zqjdw t!"
"Haskellistgarnichtsoschwer!"

Entschlüssele anschließend mit dem Schlüssel „zitat“ folgenden Text und schicke ihn mir per Mail.

Vqkkh dvgeg mckeb mmduk ymWil sigzb mlbeS tsnny sjeiv jmg,t amkdh qbdox mvxnp hzxig dUxnz daxhx m,pal fmmag vmkdx munsl .-Tlt mBnrb mo