Übung: Erste Schritte und Repl

- Arbeiten im "Read Evaluate Print Loop (REPL)".
- Arbeiten mit Listen.
- Einfache Funktionsdefinitionen und einfache Funktionen höherer Ordnung.

1 Aufgabe

Implementieren Sie eine Funktion

```
flatSort :: [[Integer]] -> [Integer]
```

mit folgenden Eigenschaften:

- `flatSort xs` enthält die gleichen Elemente wie `concat xs`.
- Die Elemente innerhalb eines Elementes vom Argument werden sortiert.
- Die Elemente von verschiedenen Elementen des Arguments werden nicht gemischt/sortiert.

Beispiele:

```
flatSort [[1,2,3],[1,3,2]] = [1,2,3,1,2,3]
flatSort [[1],[2]] = [1,2]
flatSort [[1],[1,2], [1,2,3]] = [1,1,2,1,2,3]
```

Verwenden Sie für diese Aufgabe die Zeile

```
import Data.List (sort)
```

zum Importieren von `Data.List`. Verwenden Sie userdem die Funktionen `map` und `concat`.

2 Aufgabe

Geben Sie eine Liste `xs::[[Integer]]` an, so dass `concat xs` verschieden von `flatSort xs` ist.

3 Aufgabe

Implementieren Sie die Funktion `flatSort` mit Rekursion und `++`.

4 Aufgabe

Implementieren Sie rekursiv eine Funktion

```
initial :: String -> String -> Bool
```

die entscheidet ob das erste Agrument ein Anfangssegment des zweiten Argumentes ist. Beispiele:

```
initial "abc" "abcd" == True
initial "abc" "xabcd" == False
```

5 Aufgabe

Implementieren Sie mithilfe Ihrer Funktion `initial` und der Funktion `tail`, eine Funktion

```
substring :: String -> String -> Bool
```

die entscheidet ob das erste Agrument ein Teilstring des zweiten Argumentes ist. Beispiele:

```
substring "abc" "xadbcd" == False
substring "abc" "xadbabccd" == True
```