Übungsblatt 3

Aufgabe 1

a) Definieren Sie die rekursive Funktion `rangeProduct` sodass

$$\text{rangeProduct}(m, n) = m \cdot (m+1) \cdot \ldots \cdot n$$

b) `fac` kann mittels `rangeProduct` definiert werden. Geben Sie eine entsprechende Definition.

c) Definiere rangeProduct mittels `fac`.

d) Welche Definition von `rangeProduct` ist die effizientere, angenommen `fac` habe folgende Definition?

```
fac :: Int -> Int
fac 0 = 1
fac n = n * fac (n-1)
```

Aufgabe$^\diamond$ 2

a) Geben SIe eine Definition der Funktion `maxOccurs :: Int -> Int -> (Int, Int)`, welche die größere von zwei Zahlen und ihre Häufigkeit (1 oder 2) ausgibt.

b) Definieren Sie eine Funktion `maxThreeOccurs`, welche sich verhält wie `maxOccurs`, jedoch drei Zahlen als Eingabe nimmt, und die Funktion `maxOccurs` aufruft. Vermutlich möchten Sie `where` benutzen.

Aufgabe$^\diamond$ 3

a) Definieren Sie die Funktion `matches :: Int -> [Int] -> [Int]` welche alle Vorkommnisse einer Zahl in einer Liste herausfiltriert. Zum Beispiel gilt:

```
matches 1 [1,2,1,4,5,1] = [1,1,1]
matches 1 [2,3,4,6]     = []
```

b) Definieren Sie eine Funktion `elem :: Int -> [Int] -> Bool` welche `matches` benutzt um herauszufinden ob eine Zahl in einer Liste enthalten ist. So gilt zum Beispiel:

```
elmt 1 [1,2,1,4,5,1] = True
elmt 1 [2,3,4,6]     = False
```

Aufgabe$^\diamond$ 4

Schreiben Sie eine Funktion die dezimale Zahlen in römische Zahlen umwandelt. Es genügt Zahlen die kleiner sind als 5000 zu akzeptieren. Fangen Sie dabei unerlaubte Eingaben in allen Ihren Funktionen ab und geben Sie entsprechende Fehlermeldungen aus.