Aufgabe 30

Implementieren Sie folgende Funktionen rekursiv, d.h. die Verwendung von

Schleifen ist verboten!

1. `int getAnzahlZiffern (int zahl)`:
   a. liefert die Anzahl an Ziffern der übergebenen Zahl, z.B. `getAnzahlZiffern (2542)` → 4
2. `int getZiffernWert (int zahl, int stelle)`:
   a. liefert den Wert der Ziffer der übergebenen Zahl an der Stelle; die Stellen werden dabei von rechts nach links angegeben und beginnen bei 0, z.B. `getZiffernWert(27381, 3)` → 7 oder `getZiffernWert (27381, 0)` → 1
3. `int ersetzeZiffer (int zahl, int stelle, int wert)`:
   a. ersetzt die Ziffer der übergebenen Zahl an der Stelle durch den übergebenen Wert und liefer die neue Zahl; die Stellen werden dabei von rechts nach links angegeben und beginnen bei 0, z.B. `ersetzteZiffer (24135, 3, 7)` → 27135

Aufgabe 31

Der größte gemeinsame Teiler (ggT) zweier Zahlen `a` und `b` ist eine natürliche Zahl `t` mit folgenden Eigenschaften:

1. `t` teilt `a`
2. `t` teilt `b`
3. Jede andere Zahl, die `a` und `b` teilt ist auch automatisch Teiler von `t`

Beispiel: `ggT(18, 12) = 6` (`6` teilt `18` und `12`, jede andere Zahl die `18` und `12` teilt – `1`, `2` und `3` – ist Teiler von `6`)

Berechnen Sie rekursiv den `ggT` von zwei ganzen Zahlen.

Hierbei sind zwei Informationen von Vorteil:

```
ggT(0,x) = x
ggT(x,0) = x
```

Aufgabe 32

Wir betrachten einmal die folgende Zahlenfolge:

0, 1, 3, 8, 20, 49, 119, ...

Wir erkennen sofort, dass sich eine Zahl aus der Summe vom verdoppelten Vorgänger sowie den um eins erhöhten Vorvorgänger ergibt.

a) Welche beiden Zahlen folgen der 119?

b) Schreiben Sie eine rekursive Methode, welche die n-te Zahl berechnet.

c) Schreiben Sie eine iterative Methode, welche ebenfalls die n-te Zahl berechnet.

Aufgabe 33

Die „Fibonacci-Reihe“ ist eine Folge von ganzen, positiven Zahlen $f_0, f_1, f_2, ...$ Die ersten beiden Fibonacci-Zahlen sind $f_0 = 0$ und $f_1 = 1$. Jede weitere Zahl ist die Summe der beiden Vorgänger:

$f_n = f_{n-2} + f_{n-1} \forall n \geq 2$

Der Anfang der Fibonacci-Reihe lautet 0,1, 1, 2, 3, 5, 8, ...

Schreiben Sie eine **rekursive** Funktion `fibonacci()`, die eine Zahl größer 0 akzeptiert und die entsprechende Fibonacci-Zahl f_n berechnet. Ignorieren Sie arithmetischen Überlauf.