# Versuchen Sie immer, Ihren Code zu kommentieren!

## Aufgabe 1

Schreibe eine rekursive Funktion zur Berechnung der Summe der ersten n natürlichen Zahlen. Die Funktion muss den Basisfall n == 0 behandeln. Implementiere die Funktion einmal linear und einmal endrekursiv.

## Aufgabe 2

Schreiben Sie ein Programm, das mittels Rekursion die ersten 30 Fibonacci-Zahlen berechnet und anzeigt. Die ersten beiden Fibonacci-Zahlen sind 0 und 1. Jede weitere Fibonacci-Zahl ist die Summe der beiden Vorgänger. Das ergibt die Zahlenfolge: 0,1,1,2,3,5,8,13,21,...,514229. Die Funktion muss den Basisfall n <= 1 beachten.

## Aufgabe 3

Der größte gemeinsame Teiler (ggT) zweier Zahlen a und b ist definiert als die größte ganze Zahl durch die sowohl a als auch b ohne Rest teilbar ist. Zum Beispiel ist der ggT( 42, 70 ) = 14. Eine Möglichkeit den ggT zu bestimmen besteht darin eine Primfaktorzerlegung der Zahlen durchzuführen. In diesem Fall ist das Produkt der gemeinsamen Primfaktoren der ggT:

70 = 2 * 5 * 7
42 = 2 * 3 * 7
ggT = 2 * 7 = 14

Es gibt jedoch einen wesentlich effizienteren Algorithmus (Euklidischen Algorithmus). Er gründet auf die Idee, dass der gemeinsame Teiler für a und b der gleiche ist, wie der von b und r, wobei r der Rest bei der Division von a durch b ist:
ggT(a,b) = ggT(b,r) mit r = a modulo b.
ggT( 70, 42 ) mit dem Algorithmus:

70 : 42 = 1 Rest 28
42 : 28 = 1 Rest 14
28 : 14 = 2 Rest 0
14 : 0 Ende

Im letzten Fall (Divisor 0) ist der Dividend das gesuchte Ergebnis. Schreiben Sie ein Programm, das diesen Algorithmus mittels einer rekursiven Funktion implementiert (Endrekursion).

## Aufgabe 4

Implementiere eine endrekursive Funktion zur Berechnung der Potenz einer Zahl x hoch n. Die Funktion sollte den Basisfall n == 0 berücksichtigen.

## Aufgabe 5

Entwickle eine Funktion, um die Anzahl der möglichen Wege auf einer Treppe mit n Stufen zu berechnen. Du kannst 1 oder 2 Stufen auf einmal nehmen. Die Funktion sollte die Baumrekursion verwenden. Überlege, welcher Basisfall sinnvoll ist.

## Aufgabe 6

Schreibe eine rekursive Funktion zur Berechnung der Summe aller Elemente in einem gegebenen Array von Ganzzahlen.

Beispiel:

```
int[] array = { 1, 2, 3, 4, 5 };
Die Summe aller Elemente im Array ist 15
```

## Aufgabe 7

Schreibe eine rekursive Funktion, um die Quersumme einer gegebenen Zahl zu berechnen. Die Quersumme ist die Summe der Ziffern der Zahl. Verwende den Basisfall, dass die Quersumme einer einstelligen Zahl gleich der Zahl selbst ist.

## Aufgabe 8

Schreibe eine rekursive Binärsuche, die ein bestimmtes Element in einem sortierten Array sucht. Die Funktion sollte den Index des Elements zurückgeben, wenn es gefunden wurde, andernfalls -1.

## Aufgabe 9

Schreibe einen rekursiven Bubblesort-Algorithmus. Die Funktion soll ein Array von ganzen Zahlen sortieren. Tausche benachbarte Elemente, wenn sie in der falschen Reihenfolge sind, und sortiere den Rest der Liste rekursiv. Die Parameter sollen das Array int[] arr sowie die Länge int n sein.

```
RecursiveBubbleSort(int[] arr, int n)
```