Vorgabe für den funktionalen Teil:

Scala-Funktionen sind immer gemäß der Definition im Kapitel „Typen und Funktionen", also rein-funktional zu implementieren, es sei denn diese Vorgabe ist in der Aufgabenstellung aufgehoben.

Aufgabe 1 (10 Punkte)

a) Wann nennt man eine Funktion tail-rekursiv?
b) Welcher Vorteil liegt bei tail-rekursiven Funktionen vor?
c) Geben Sie eine tail-rekursive Scala-Funktion zur Implementierung der Summe der ersten n natürlichen Zahlen an.
d) Begründen Sie, warum Ihre Implementierung aus c) tail-rekursiv ist.

Aufgabe 2 (10 Punkte)

a) Beschreiben Sie, wie sich die Call-By-Value und Call-By-Name-Auswertungsstrategien unterscheiden.

Gegeben seien folgende Funktionsdefinitionen in Scala:

```
def f1(i: Int) : Int = i - 1
def f2(i: Int) : Int = f1(i-1) * i
def f3(i: Int) : Int = f2(i-1) * f3(i)
```

Führen Sie die ersten 4 Schritte der Auswertung des Ausdrucks f3(f1(3)) durch und unterstreichen Sie bei jedem Schritt den Teilausdruck, der als nächstes ausgewertet wird. Nutzen Sie als Auswertungsstrategie

b) Call-by-Value
c) Call-By-Name

Aufgabe 3 (10 Punkte)

Seien `map` und `filter` die aus der Vorlesung bekannten Funktionen auf Integer-Listen in Scala-Notation.

a) Geben Sie die Typ-Definition der Funktion `map` auf Integer-Listen in Scala-Notation an.
b) Implementieren Sie die Funktion `filter` auf Integer-Listen als Scala-Funktion.
c) Gegeben sei die Eingabeliste liste0. Konstruieren Sie aus Aufrufen von `filter` und `map` einen Ausdruck, der aus der Eingabeliste eine Liste generiert, für die Folgendes gilt: Die neue Liste enthält die Hälfte jeder geraden Zahl aus der Eingabeliste (Die ungeraden Zahlen werden verworfen) Bsp.: Eingabe: `[1,2,3,4,7,8]`, Ergebnis: `[1,2,4]`