Übung 2 – Aufgabenblatt

Aufgabe 1: Implementieren Sie eine Funktion or, die ein logisches oder repräsentiert. Sie sollzwei Parameter x und y vom Typ Boolean als Parameter enthalten und ein Boolean zurückgeben. Programmieren Sie die Funktion so, dass nur das erste Element x ausgewertet wird. Wenn x ein true enthält und y einen fehlerhaften z.B. mit einer Exception endenden Ausdruck, so kommt es trotzdem zu einem Ergebnis. Überprüfen Sie dies, in dem Sie der Funktion Ausdrücke übergeben, die in einer Exception/Error enden. (Verwenden Sie bei der Implementierung nicht die Scala oder-Funktion).

Aufgabe 2: Gegeben sei der folgende Programmcode:

```
def myMethod(param:Int):String=

        if param<0 then "kleiner null"
        if param>0 then "größer null"
        else "null"
```

Die Methode soll eine Integerzahl in einen String übersetzen. Dabei wird überprüft, ob der übergebene Parameter größer, kleiner oder gleich 0 ist und ein entsprechender Text zurück gegeben.

Überlegen Sie, ob das Programm das gewünschte Ergebnis liefert und falls nicht, wie Sie den Code ändern müssen, um das Ergebnis zu erhalten.

Aufgabe 3: Gegeben sei der folgende Progammcode:

```
val x=
        val offset= 1;

        {

                val x=2
                val offset=10
                x+offset

        } +
        {
                val x=5
                x+offset
        }
```

Überlegen Sie, erstens womit die Variable x initialisiert wird und zweitens, was passieren würde, wenn Sie das Semikolon in der zweiten Zeile weggelassen wird.

Aufgabe 4: Implementieren Sie eine Funktion squareUnder(x:Double, max:Double):Double, die eine Zahl x so lange quadriert, bis max überschritten ist. Zurückgegeben wird die letzte errechnete Quadratzahl, die darunter liegt.

Aufgabe 5: Schreiben Sie eine Funktion teiler(zahl:Int):Int, die den größten ganzzahligen Teiler der übergebenen Zahl– kleiner als die Zahl selbst – berechnet. Das heißt, wird eine Primzahl übergeben, so gibt die Funktion 1 zurück, ist es keine Primzahl den entsprechenden Teiler.

Implementieren Sie eine Variante, die von oben nach unten zählt und eine Variante, die von unten nach oben zählt.

Aufgabe 6: Schreiben Sie eine Funktion, quersumme mit der folgenden Signatur: def quersumme(zahl:Int):Int. Sie soll die Quersumme der Zahl berechnen, die an die Funktion übergeben wurde.

Aufgabe 7: Die Fibonacci-Folge ist eine unendliche Folge von Zahlen (den Fibonacci-Zahlen), bei der sich die jeweils folgende Zahl durch Addition ihrer beiden vorherigen Zahlen ergibt: 0, 1, 1, 2, 3, 5, 8, 13, … Benannt ist sie nach Leonardo Fibonacci, der damit 1202 das Wachstum einer Kaninchenpopulation beschrieb.

Die Fibonacci-Folge $f_0, f_1, f_2, \cdots$ ist durch das rekursive Bildungsgesetz

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

mit den Anfangswerten

$$f_0 = 0 \text{ und } f_1 = 1$$

definiert. Das bedeutet in Worten:

- Für die beiden ersten Zahlen werden die Werte *null* und *eins* vorgegeben.
- Jede weitere Zahl ist die Summe ihrer beiden Vorgänger.

Schreiben Sie eine Funktion fibo (X), die für eine beliebige Zahl X, die Fibonacci-Zahl berechnet.

Aufgabe 8: Wandeln Sie die Funktion aus Aufgabe 2 so um, dass der Aufruf der Funktion fibo(100) zu einem richtigen Ergebnis kommt.

Aufgabe 9: 2520 ist die kleinste Zahl, die durch jede Zahl von 1-10 ohne Rest geteilt werden kann. Was ist die kleinste positive Zahl, die durch alle Zahlen von 1-20 ohne Rest teilbar ist? (Projekt Euler Aufgabe 5)

Schreiben Sie eine Funktion, die in Abhängigkeit von einer Zahl X berechnet, welches die kleinste Zahl ist, die durch alle Zahlen von 1..X ohne Rest teilbar ist.

(Als kleiner Tipp: Schreiben Sie erst eine Funktion, die testet, ob eine Zahl durch eine Menge von Zahl teilbar ist oder nicht. Dann lassen rufen Sie die Funktion solange auf, bis sie einen entsprechenden Wert gefunden haben.)

Die Zahl ist: 232792560

Aufgabe 10: Die Summe aller Primzahlen der Zahlen bis 10 ist: 2+3+5+7 =17. Schreiben Sie eine Funktion, die die Summe aller Primzahlen unter 2 Millionen bildet (Ergebnis: 12272577818052).

Aufgabe 11: Implementieren Sie die Methode calculatePi, die auf Basis Zufall die Zahl Pi ermittelt (Monte Carlo Algorithmus – Nachlesbar in Wikipedia) Verwenden Sie dabei keine Variablen sondern nur Rekursionen. Zufallszahlen erzeugen Sie mit der Klasse Random, die Funktion nextDouble enthält:

```
import scala.util.Random
val randGen= Random
```

Verwenden Sie für die Lösungen nur Elemente aus der Funktionalen Programmierung, d.h. hier nur unveränderliche Variablen und Rekursionen.