Abteilung Algorithmik **Winter 2017/18**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Christian Rieck
Arne Schmidt

Algorithmen und Datenstrukturen
Übung 0a vom 30. 10. 2017

Dieses Blatt dient lediglich der persönlichen Vorbereitung. Es wird nicht abgegeben und geht nicht in die Bewertung ein. Die Aufgaben und ihre Lösungen werden in den kleinen Übungen besprochen.

Aufgabe 1 (Euklidischer Algorithmus): Zur Bestimmung des größten gemeinsamen Teilers (ggT) zweier Zahlen eignet sich der sogenannte *Euklidische Algorithmus* (siehe Algorithmus 1). Er bekommt als Input zwei ganze Zahlen a und b und gibt den ggT von a und b zurück. Führe den Euklidischen Algorithmus mit den Zahlen 256 und 106 aus. Gib *nach* jeder Iteration der **while**-Schleife die Werte von a und b aus.

```
function EUKLID(a, b)
    while b ≠ 0 do
        h ← a mod b
        a ← b
        b ← h
    end while
    return a
end function
```

Algorithmus 1: Euklidischer Algorithmus

Aufgabe 2 (Fibonacci-Zahlen): Die Fibonacci-Zahlen sind rekursiv definiert. So berechnet sich eine bestimmte Fibonacci-Zahl aus der Summe der beiden vorherigen Fibonacci-Zahlen. Konkret: $F(n) := F(n-1) + F(n-2)$ wobei $F(0) = 0$ und $F(1) = 1$. Wir wollen nun zwei Algorithmen (siehe Abbildung 1) testen, die $F(n)$ berechnen.

a) Berechne $F(5)$ mit FibonacciRek und gib dabei *alle* Aufrufe der Funktion an. Wie oft wird eine Summe berechnet? Hat FibonacciRek alle Eigenschaften eines Algorithmus?

b) Berechne $F(5)$ mit Fibonacci und gib dabei die Werte aller $f(i)$ aus. Wie viele Summen wurden berechnet? Hat Fibonacci alle Eigenschaften eines Algorithmus?

```
function FibonacciRek(n)
    if n ≤ 1 then
        return n
    end if
    a ← FibonacciRek(n − 1)
    b ← FibonacciRek(n − 2)
    return a + b
end function
```

```
function Fibonacci(n)
    f(0) ← 0
    f(1) ← 1
    for i in 2, ..., n do
        f(i) ← f(i−1) + f(i−2)
    end for
    return f(n)
end function
```

Abbildung 1: Zwei Algorithmen für $F(n)$: Rekursiv (links) und „mit Gedächtnis“(rechts)

Aufgabe 3 (Primfaktorzerlegung): Bei einer Primfaktorzerlegung einer Zahl n werden k nicht notwendigerweise verschiedene Primzahlen $p_1, \ldots, p_k$ gesucht, sodass deren Produkt n ergibt. So ist zum Beispiel 3185 das Produkt der Primzahlen 5, 7, 7 und 13.

a) Gib einen Algorithmus (in Pseudocode, beispielsweise wie in Algorithmus 1 oder 1) an, der die Primfaktorzerlegung der gegebenen Zahl n ausgibt.

   Hinweise:

   (i) Der kleinste Teiler > 1 ist immer eine Primzahl.

   (ii) Eine Zahl a teilt b wenn $b \bmod a = 0$

   (iii) Ist ein Teiler a von b gefunden, sind die restlichen Faktoren von b in $\frac{b}{a}$ enthalten.

b) Benutze Deinen Algorithmus, um die Primfaktorzerlegung von 4680 finden.