Übungen Funktionale Programmierung (in Clojure) Serie 6

1. Komposition von Funktionen

(a) Schreiben Sie eine Funktion, die von einer Zahl 2 abzieht, ohne dass das Zeichen – im Code auftaucht

(b) Gegeben sei die Funktion `(defn sq[x] (* x x))`. Schreiben Sie eine Funktion sqplus, die eine Zahl um `1` erhöht und dann quadriert. Im Code darf + nicht vorkommen.

2. Eine Funktion n-mal anwenden

Wenn f eine numerische Funktion und n eine positive ganze Zahl ist, dann können wir die n-fach wiederholte Anwendung von f bilden, die als Funktion von x mit dem Wert $f(f(\ldots(f(x))\ldots))$ definiert ist.

Wenn z.B. $f(x) = x + 1$ ist, dann ist die n-fach wiederholte Anwendung von f die Funktion $g(x) = x + n$.

Wenn z.B. f die Quadrierung ist, dann ist g die 2^n-te Potenzierung.

Schreiben Sie eine Funktion `n-fach` mit einer Funktion `f` und einer positiven ganzen Zahl n als Parameter, die sich z.B. folgendermaßen verwenden läßt:

```
; ((n-fach quadrat 3) 5)
; => 390625
```

Hinweis: Denken Sie an Rekursion und `comp`.

Gibt es auch elegantere Lösungen?

3. Eine merkwürdige Funktion

[Aufgabe 1.34 in SICP]

Gegeben sei die Funktion

```
(defn f [g] (g 2))
```

(a) Was ergeben folgende Beispiele?

```
(f #(* % %))

(f (fn [z] (* z (inc z))))
```

(b) Was passiert, wenn wir (f f) auswerten lassen? Genaue Erklärung?