Rekursion

1 Rekursion (3 Punkte)

Doppelrekusion: erweitere die Funktion `sum` aus der letzten Übung so, daß alle Zahlen einer beliebig geschachtelten Liste summiert werden. Ein Pluspunkt wenn auch Listen mit punktierten Pärchen korrekt verarbeitet werden können. Beispiel:

```
(sum '(1 (2 3) (((((31)) 5))))) -> 42
```

2 Mehr Rekursion (3 Punkte)

Schreibe eine Funktion, die eine Liste mit (natürlichen) Zahlen nimmmt und eine Liste gleicher Länge zurückgibt. Dabei wird jede Zahl n durch eine Liste der Länge n ersetzt, deren Elemente eine fallende Sequenz sind. Beispiele:

```
(foo '(1 2 3)) -> ((1) (2 1) (3 2 1))
(foo '(4 2 42 0)) -> ((4 3 2 1) (2 1) (42 41 ... 1) nil)
```

Denk darüber nach wie das Problem zerlegt werden kann!

3 Noch mehr Rekursion (3 + 3 + 3 = 9 Punkte)

Schreibe eine zweistellige rekursive Funktion `vereinigung()`, die aus zwei Mengen in Listendarstellung (jedes Element ist nur einmal enthalten; die Ordnung der Elemente ist irrelevant) die Vereinigungsmenge berechnet. Definiere analoge Funktionen `schnitt()` und `differenz()`, die die Schnitt- beziehungsweise Differenzmengenberechnung ausführen; Hinweis: Ein guter Test ob die Funktionen funktioneren ist es, die Ergebnisse mit `set-equal` zu vergleichen mit denen, die man mit hilfe der vordefinierten Funktionen `union`, `set-difference` und `intersection` bekommt.

Beispiel: Sei `l1` und `l2` an zwei listen gebunden.
Dann soll: `(set-equal (union l1 l2) (vereinigung l1 l2)) -> T`.

4 Geheimnisvolle Rekursion (5 Punkte)

Beschreibe den Zweck der folgenden Funktion:

```
(defun elch (foo)
   (cond ((null foo) foo)
         ((atom foo) (list foo))
         (t (append (elch (first foo))
                    (elch (rest foo))))))
```

Wie genau funktioniert `elch()`? Gehe in der Beschreibung auf jeden der drei Zweige der Fallunterscheidung ein: warum sind sie nötig; was machen sie?

5 Schmökrekursion (2 bzw 3 rekursive Schmökpunkte)

Erweitere die Funktion `sum` aus Aufgabe 1a (2 Punkte) 1b (3 Punkte) so, daß die Liste auch andere Elemente enthalten kann (die allerdings nicht summiert werden sollen).