1 Einfaches Mapping (2 + 2 Punkte)

a) Definiere eine einstellige Funktion `get-even`, die als Argument eine Liste mit Ganzzahlen nimmt und durch eine geeignete Mapping-Operation (d.h. eine Kombination aus mapping- und anonymer `lambda`-Funktion) alle ungeraden Zahlen n durch $n+1$ ersetzt.

b) Definiere eine einstellige Funktion `eliminate-odds`, die als Argument eine Liste mit Ganzzahlen nimmt und durch eine geeignete mapping-Operation alle ungeraden Zahlen unterdrückt, z.B.

```
? (get-even '(1 2 3 4 5)) --> (2 2 4 4 6)
? (eliminate-odds '(1 2 3 4 5)) --> (2 4)
```

2 Destruktiv (5 Punkte)

Erkläre das Verhalten der folgenden Funktion:

```
(defun fools-comparative (letter-list)
   (nconc letter-list '(e))
   (nconc letter-list '(r)))

? (setq *print-length* 10)
? (fools-comparative '(q u i c k))  -->   (Q U I C K E R)
? (fools-comparative '(l o n g))    -->   (L O N G E R R R R R ...)
? (fools-comparative '(f a s t))    -->   ...Endlosschleife...
```

Achtung: Wenn ihr das ausprobiert, dann müßt ihr das LISP abbrechen, evtl. sogar neu starten. Ein Hinweis: malt euch die Cons-Kästchen auf! Und noch ein Hinweis:

```
(defun equal-or-eq ()
   '(test))

? (setq a (equal-or-eq))    -->   (TEST)
? (setq b (equal-or-eq))    -->   (TEST)
? (equal a b)               -->   T
? (eq a b)                  -->   ...ausprobieren...
```

3 Iteration (5 Punkte)

In einer CD-Datenbank sind die Informationen über einzelne CDs als Listen abgespeichert. Enthalten sind Titel, Interpret, Komponist und für jedes Stück als Liste der Titel und die Zeit (wieder eine Liste mit Minuten und Sekunden). Zum Beispiel:

```
(setq four-seasons
      '("Vier Jahreszeiten" "Sir Yehudi Menuhin" "Vivaldi"
 ("Fruehling" (10 23)) ("Sommer" (12 23)) ("Herbst" (13 1)) ("Winter"   (9 32)))
```

Definiere eine Funktion `gesamtdauer`, die mittels `dolist` oder mapping für eine CD die Gesamtlänge als Liste mit Minuten und Sekunden berechnet. Hinweis: die Funktionen `floor` und `mod` sind hilfreich. Damit es übersichtlich bleibt, definiert eine Hilfsfunktion für das Addieren von Zeiten.

```
(gesamtdauer four-seasons) --> (45 19)
```

4 Heavy Mapping (3 + 3 Punkte)

Schreibe mithilfe einer geeigneten Mapping-Funktion eine einstellige rekursive Funktion `flatten`, die alle Atome einer (möglicherweise) verschachtelten Eingabeliste als flache Liste zurückgibt, z.B.

```
? (flatten '(a (b (c (d) e)))) --> (A B C D E)
```

Erweitere `flatten` zu `how-deep`, so daß statt des Atoms eine Liste mit Atom und der ursprünglichen Tiefe zurückgegeben wird:

```
? (flatten '(a (b (c (d) e)))) --> ((A 1) (B 2) (C 3) (D 4) (E 3))
```

5 Schmökaufgabe: `my-mapcar` (2 + 2 Schmökpunkte)

Definiere als eigene Version von `mapcar` eine rekursive Funktion `my-mapcar`, die keine `map` Funktion außer natürlich `my-mapcar` verwendet. `apply` oder `funcall` dürfen verwendet werden.

Der Haken: `mapcar` kann auch mehr als eine Liste als Argument nehmen. Dann muß die Funktion, die als erstes Argument übergeben wird, entsprechend viele Argumente verarbeiten. `mapcar` bricht ab, sobald eine der Listen am Ende ist. Zum Beispiel:

```
(mapcar #'+ '(1 2 3) '(42 17) '(1 1 1 1 1 1 1)) --> (44 20)
```

Abgabetermin ist der 9. Dezember 1999