Backquote et. al. (3 Punkte)

Schreibe mit Hilfe von `append, cons` und `list` einen äquivalenten Ausdruck für

`` `(l1 ,l1 (,(rest l2)) ,@l2) ,l2)? ``

unter der Voraussetzung daß:

```
(defparameter l1 '(1))
(defparameter l2 '(2 3))
```

N-times (3 + 2 Punkte)

Schreibe ein Makro, `n-times`, das 3 Argumente nimmt, ein Funktionssymbol, eine S-expr und eine Zahl N, und zu einem Stück Code expandiert, das das Funktionssymbol N-mal auf die S-expr anwendet. Z.B.:

```
> (macroexpand '(n-times foo (* 42 42) 3)) -->

(foo (foo (foo (* 42 42))))
```

Erkläre, warum `n-times` nicht zu `(foo (foo (foo 1764)))` expandieren darf (mit anderen Worten: Warum darf man die S-expr bei der Expansion nicht evaluieren).

Hinweis Denk darüber nach (probiere!) was passiert wenn `foo` Kompiliert wird:

```
(defun foo (n)
   (n-times 1+ 42 n))
```

Lexikon ADT (12 Punkte)

Schreibe einen Abstrakten Datentyp (ADT) für ein einfaches Lexikon. Es soll mindestens die Funktionen `make-lexicon` (Konstruktor, nullstellig) und `clear-lexicon` (einstellig) zum Erzeugen bzw. Löschen eines Lexikons, eine Selektorfunktion `find-in-lexicon` (zweistellig: ein Lexikon und ein Wort (ein String)) und eine Einfügefunktion `store-in-lexicon` (zweistellig: ein Lexikon und ein Lexikoneintrag) geben. Dafür müßt ihr zuerst definieren, was denn ein Lexikoneintrag ist:

Schreibe einen ADT für einen einfachen Lexikoneintrag. Ein Lexikoneintrag besteht aus einem Word (ein String) und der zugehörigen Kategorie (ein Symbol). Es soll mindestens eine Konstruktorfunktion `make-lex-entry` und Selektorfunktionen für `lex-entry-word` und `lex-entry-cat`.

Welche weiteren Funktionen wären nützlich für die ADTs?

Teste die Funktionen, indem Du aus einer Datei ein Lexikon einliest. Dazu könnt ihr folgenden Code verwenden:

```
(defparameter *lexicon* nil)
(with-open-file (lex-stream "/tmp/lex.txt")
     (loop for word = (read lex-stream nil :eof)
           for cat =  (read lex-stream nil :eof)
           until (eq word :eof)
           do (store-in-lexikon '*lexicon* (make-lex-entry word cat))))
```

In der Datei `/tmp/lex.txt` sollten dann die Einträge zeilenweise stehen:

```
"Petra" NOMEN
"liest" VERB
"ein"   ARTIKEL
"Buch"  NOMEN
```

Schmökaufgabe: my-let (7 Schmökpunkte)

Schreibe ein Makro `my-let()`, das sich wie das vordefinierte `let()` verhält und die lokale Bindungsumgebung durch Aufruf einer anonymen Funktion erzeugt.

Hinweis: Der Prototyp des Makros soll sein:

```
(defmacro my-let (bindings &rest body) ... )
```

Noch ein Hinweis: die Expansion des Makros soll beispielsweise den folgenden Ausdruck erzeugen:

```
(macroexpand '(my-let ((foo (+ 4 2)) (bar 2)) (+ foo bar)))
→ (FUNCALL #'(LAMBDA (FOO BAR) (+ FOO BAR)) (+ 4 2) 2)
```