Wir spielen Common-Lisp Interpreter (6 Punkte)

Was berechnen die folgenden Funktionen; wozu sind sie gut? Erkläre ihre Funktionsweise an (mindestens) je einem Beispiel.

a)
```
(defun ! (!)
    (append ! (reverse !)))
```

b)
```
(defun ? (?)
    (nth (- (length ?) 2) ?))
```

Listen als Mengen (4 Punkte)

Schreibe eine eigene (rekursive) zweistellige Funktion `mein-subsetp`, die sich so verhält wie `subsetp`. `mein-subsetp` gibt `T` zurück, genau dann wenn jedes Element der ersten Liste Element der zweiten Liste ist, sonst `NIL`, z.B.

```
(mein-subsetp '(a b) '(b l a c k)) --> T
(mein-subsetp (list 'a 'b) (list 'b 'c 'd)) --> NIL
(mein-subsetp NIL '(1 a ())) --> T
(mein-subsetp '( NIL ) '(a b c)) --> NIL
```

Kontrollstrukturen und Typprädikate (5 Punkte)

Schreibe eine einstellige Funktion, die alle Typen die wir bis jetzt erwähnt haben erkennt und verbalisiert. Wenn der Typ unbekannt ist wird das Symbol `WEISS-NICHT` zurückgegeben.

```
? (verb-typ nil) --> LEERE-LISTE
? (verb-typ 42) --> ZAHL
? (verb-typ (list 42)) --> LISTE
[...]
? (verb-typ #\c) --> WEISS-NICHT
```

Rekursion (5 Punkte)

Schreibe eine Funktion, die als Argument eine Liste nimmt und die Summe aller Elemente der Liste berechnet.

```
? (meine-summe '(1 2 3 4 5 6)) -->  21
? (meine-summe nil) --> 0
```

Schmök-Rekursion (3 Punkte)

Schreibe eine Funktion, die als Argument eine Liste von Listen mit Zahlen nimmt, und das Produkt aller Summen berechnet. *Hinweis* Die Funktion in der obigen Aufgabe, läßt sich gut dabei verwenden.

```
? (mein-sp '((1 2 3) (4 5) (6 7))) --> 2268
? (mein-sp (list nil)) --> 0    ;; !
? (mein-sp nil) --> 1           ;; !!
```

Abgabetermin ist der 11. Nov. 1999