1 Generalisierte Variablen (1 + 3 + 3 + 3 Punkte)

a) Lies die (relevanten Teile der) Dokumentation zu den neu eingeführten Common-Lisp Funktionen (siehe ‘`http://www.dfki.de/~becker/algorithmen/sitzung2.html`’).

b) Erkläre die Auswirkungen (inklusive möglicher Seiteneffekte) der folgenden Anweisungen; am besten unter Verwendung der Kästchenschreibe für `cons()`-Zellen:

```
? (defparameter *foo* (list 1 2 3))
→ *FOO*
? (setf (first (rest *foo*)) 42)
→ 42
```

c) Schreibe eine zweistellige, rekursive Funktion `append2()`, die ihre zwei Eingabelisten ähnlich wie das vordefinierte `append()` konkateniert; allerdings soll `append2()` völlig ohne Aufruf von `cons()` oder anderer Funktionen, die neue `cons()`-Zellen erzeugen, auskommen, e.g.

```
? (append2 '(1 2 3) '(4 5 6))
→ (1 2 3 4 5 6)
```

Hinweis: Die Konkatenation soll durch destruktive Veränderung (mit `setf()`) der ersten Eingabeliste realisiert werden.

d) Diskutiere den Effekt von `append2()` anhand der Kästchendarstellung des obigen Beispiels. Gib geeignete Beispielaufrufe an, die zeigen, daß die Veränderung der ersten Eingabeliste dauerhaft ist.

2 Verbmorphologie (4 + 3 + 1 + 2 Punkte)

a) Modifiziere die Funktion `token2code()` so, daß sie als Alphabet alle alphanumerischen Zeichen (aus dem Bereich `char-code()` $>= 0$ und < 256) auf Zahlen von 2 aufwärts abbildet (die ersten beiden Spalten der Übergangstabelle sind reserviert). Anders als in der Kodierung mit `case()` soll `token2code()` nunmehr konstante Komplexität haben. Führe weiterhin eine globale Variable `*code-maximum*` ein, die den größtmöglichen Rückgabewert für `token2code()` enthält (i.e. den größten Spaltenindex).

b) Schreibe eine einstellige Funktion `string2table()`, die aus einer Zeichenkette einen Automaten (mit dem Alphabet aus Aufgabenteil (a)) erzeugt, der die übergebene Zeichenkette und weiter nichts erkennt, e.g.

```
? (traverse (string2table "denk") "denk")
→ (4)
```

c) Extrahiere aus dem Automaten aus Aufgabe (2) des ersten Übungsblattes den Teil, der die Suffixe der deutschen Verbmorphologie (im Präsens, Indikativ, Aktiv) erkennt; erweitere den Automaten für das in (a) angegebene Alphabet und ändere die Funktion `traverse()` dahingehend, daß als Rückgabewert nicht nur ‘`t`’ oder ‘`nil`’ sondern ein beliebiger Wert geliefert wird, der mit dem erreichten Endzustand assoziiert ist, e.g.

```
? (traverse *praesens* "st")
→ (2sg)
```

d) Zeige, wie durch Kombination des Automaten aus Aufgabenteil (c) mit `string2table()` und `concatenate-automata()` (‘`http://www.dfki.de/~becker/algorithmen/concatenate.lsp`’) die Präsensformen der Verben *denken*, *lachen* und *trinken* analysiert werden können.

Abgabetermin ist der 7-MAI-1999