INDENTATION AND SPACING GUIDELINES FOR LISP ASSIGNMENTS 3, 4, and 5

1. When a list is written on more than one line, the first character of the 2nd and any subsequent lines in that list should be FURTHER TO THE RIGHT than the list's opening parenthesis.

2. Do NOT put a space immediately after an opening parenthesis or immediately before a closing parenthesis.
   BAD: (this is bad) GOOD: (this is right)

3. When two consecutive elements of a list are on the same line, they should be separated by a space:
   BAD: ((this is)(a bad)example) GOOD: ((this is) (a good) example)

4. Do NOT put closing parentheses on their own lines. For example, you should NOT write:

```
(defun f (x y z)
   (+ x (* y z))
)
```

   Instead, you can write:

```
(defun f (x y z)
   (+ x (* y z)))
```

   REASON: Putting closing parentheses on separate lines wastes screen space for no good reason and increases the amount of scrolling that is needed to read your code!

5. There should be a blank line after each function definition.

6. For each function call that is not written on a single line, make all the arguments of the call line up, as in:

```
(myfunction argument1
            argument2
            argument3)
```

   or

```
(myfunction
  argument1
  argument2
  argument3)
```

   In the second case the arguments should line up with one of the first few letters of the function name. If argument1, argument2, and argument3 are (func x y), (+ 2 w), and (anotherfunc w (+ y 1)), then you might write the call as follows:

```
(myfunction
  (func x y)
  (+ 2 w)
  (anotherfunc w (+ y 1)))
```

   Another way to write the call would be:

```
(myfunction (func x y)
            (+ 2 w)
            (anotherfunc w
                         (+ y 1)))
```

7. For each IF form that is not written on a single line, make the opening parentheses of the three arguments line up, as in:

```
     (if (evenp n)
         (func
            w
            (+ y 1))
         (- n 3))
```

   It is also OK to put the first two arguments on the same line as the symbol IF and put the third argument on the next line. In that case the third argument should line up with the first argument, as in the following example from p. 115 of Touretzky:

```
     (if (symbolp x) (list 'yes x 'is 'a 'symbol)
         (list 'no x 'is 'not 'a 'symbol))
```

8. For each COND form, make the opening parentheses of the clauses line up, as in:

```
     (cond ((eql e 0) '(integer-zero 0))
           ((numberp e) (list 'num-but-not-0 e))
           ((consp e) (list 'nonempty-list e))
           (t (list 'non-numeric-atom e)))
```

   This COND form could also be formatted as follows:

```
     (cond ((eql e 0)
            '(integer-zero 0))
           ((numberp e)
            (list 'num-but-not-0 e))
           ((consp e)
            (list 'nonempty-list e))
           (t (list 'non-numeric-atom e)))
```

   or

```
     (cond
        ((eql e 0)
           '(integer-zero 0))
        ((numberp e)
           (list 'num-but-not-0 e))
        ((consp e)
           (list 'nonempty-list e))
        (t (list 'non-numeric-atom e)))
```

9. For each LET or LET* form, put the first local variable initialization form on the same line as the symbol LET or LET*. If more than one local variable is initialized, make all the initialization forms line up. Indent the body of the LET or LET* form by two spaces. Here is an example (from p. 144 of Touretzky):

```
     (let* ((diff (- new old))
            (proportion (/ diff old))
            (percentage (* proportion 100.0)))
       (list 'widgets 'changed 'by percentage 'percent))
```

I would also like to remind everyone that when writing Lisp code you should use an editor that matches parentheses for you. Some such editors were mentioned on the last page of the Assignment 2 document.