Programming Languages

Spring 2019 Lisp Practice

Dr. Gurka

Goal: understand how the various built-in functions work.

No submission for this assignment. It’s complete when you are comfortable using Lisp basics. make notes on the quick reference sheet for all the basic functions, to be used on larger projects and the Lisp quiz.

-----

Work with all functions listed on the handout “in the small” (not part of a larger function, just by themselves). Look for when your Lisp dialect does and does not need a quote. Learn required parameter types (list or atom or either) by trying various combinations. Try the empty list wherever possible, using ( ) [empty parentheses] and NIL; these are interchangeable.

expressions to try (predict before you run them), some are valid, others are not:

(car ‘(a b c))

(car (a b c))

(car a)

(car ‘a)

(car ( (a b) (c d) ))

(car NIL)

(car ())

(cdr ‘(a b c))

etc.

(listp NIL)

(listp ())

(listp ‘())

(listp cat)

(listp ‘(cat))

etc.

(a b c) (unknown function a)

‘(a b c)

a

‘a

Work with append, list, and cons so you know their parameters – what they must have, what they might have, what will be an error – and what the results will be.

Examples:

(append ‘(a b c) ‘(1 2 3)) (appending 2 lists?)

(append a ‘(1 2 3)) (appending an atom to a list?)

(append a (1 2 3)) (appending an atom to a non-quoted list?)

(append ‘a ‘(1 2 3)) (appending a quoted atom to a quoted list?)

(append () ‘(1 2 3)) (appending with the empty list?)

etc.

Then do cons and list on the same test data.

Work with the two forms of equality test: =, equal. Which ones work on which kind of data?

-----

Some functions to write (defun):

(add-list2 lst) (may have nested lists and non-numeric atoms)

(is-pal lst) (is a palindrome, find 2 or 3 ways to do this)

(count1 x lst) (how many times does x, an atom, appear in lst)

(count2 lst1 lst2) (how many times is lst1 embedded in lst2, harder)

(remove x lst) (remove all instances of x, an atom, return the pruned list)

(replace x1 x2 lst) (replace all instances of x1 with x2, return the new list)

(is-sorted lst) (obvious)

(has-dups1 lst) (does a sorted list have duplicates)

(has-dups2 lst) (does a list with unknown ordering have duplicates)

(remove-dups lst) (remove duplicates)

Write simple versions that take a list with no nesting (no embedded lists), and then more robust ones that will dive into nested lists as needed. Test thoroughly.

And for more fun, write a sort …

---