**CS 352 Spring 2018 -- Programming Assignment #4 (20 points)**

***Due date: Tuesday, April 17***

***Do the problems in Lisp or Python (though Lisp is recommended but Python okay – if you work in Python, you may need to do minor adjustment on problem statements such as representing lists using [] etc.)***

**Problem 1*:*** Write a recursive function **COUNT#** which takes a list and returns the number of numbers which are found in the list, at any depth or nesting. For example,

**(count# nil) → 0**

**(count# 3) → 1**

**(count# ’a) → 0**

**(count# ’(5 1 a 4)) → 3**

**(count# ’((1 1 (a)) 7 (b (3)))) → 4**

(defun count# (L)

(cond ((null L) 0)

((atom L) (cond ((numberp L) 1) (T 0)))

(T (+ (count# (car L)) (count# (cdr L))))))

**Problem 2:** Write two functions, one recursive and another none-recursiveto check if all atoms of a list form a Palindrome. You cannot use predefined *reverse* function.

***Required test cases for each function:*** (note: you may name your functions other than ‘Pal’.)

(Pal 'nil) -> nil

(Pal ‘a) -> nil

(Pal '(a)) -> T

(Pal '(a b c b a)) -> T

(Pal '(a b c c b a)) -> T

(Pal '(a b c a)) ->nil

(Pal '(a b d e f f c b d a)) ->nil

(Pal ‘((a b) (b a))) -> nil

(defun Pal (L)

(cond ((null L) nil)

((atom L) nil)

(T (cond ((null (cdr L)) T) ;list of one element

((listp (car L)) nil)

((listp (nth (1- (length L)) L)) nil)

((eq (car L) (nth (1- (length L)) L))

(cond ((= (length L) 2) T)

(T (pal (subseq L 1 (1- (length L)))))))

(T nil)))))

(Pal 'nil)

(Pal 'a)

(Pal '(a))

(Pal '(a b c b a))

(Pal '(a b c c b a))

(Pal '(a b c a))

(Pal ‘((a b) (b a)))

[33]> (load "rpal.lsp" :print T)

;; Loading file rpal.lsp ...

PAL

NIL

NIL

T

T

T

NIL

NIL

NIL

;; Loaded file rpal.lsp

T

(none recursive version omitted)