**Practice Questions (Part 1)**

Q1 ) Evalute the value of following expression:

a) (/ ( - 7 1) ( - 4 2))

b) ( l i s t 'my (+ 2 1) "Sons")

c) ( l i s t '( + 2 1) ( + 2 1))

d) (cons 'a '(b c d))

e) (car '( a b c))

f) (car (cdr (cdr '( a b c d ) ) ) )

g) (listp '( a b c))

Q2) what is the output of this conditional expression:

 (if ( listp '(a b c)) (+ 1 2) (+ 5 6))

Q3) Let a function be defined as (defun our-third (x) (car (cdr (cdr x ) ) ) ) . Now what will be the output if we call the function as:
(out-third '(a b c d e))

Q4) Let a function be defined as (defun sum-greater (x y z) (> (+ x y) z)). What will be the output if we call function as:

(sum-greater 1 4 3)

Q5) Consider the following program:

(defun show-squares ( start end)

 (do ( ( i start (+ i 1 ) ) )

 ((> i end) 'done)

 (format t "~A ~k~%" i (\* i i ) ) ) )

Now we call the function as:

(show-squares 2 5)

Q6) WAP to calculate the power of a given number. e.g. 2^3 = 8 ; 3^5 = 81

Q7) Write a function called pick that squares a number if it is negative, and adds 2 to it otherwise. Try it on -3 and +2.

(pick -3) should evaluate as 9, while (pick 2) should evaluate as 4.

Q8) (a) Write a function call zero-it, which, with argument x, evaluates as zero if x is negative, and as x otherwise. So (zero-it -2) evaluates as 0,

while (zero-it 3) evaluates as 3

 (b) Now use mapcar together with zero-it to provide a function which replaces all the negative numbers in a list by zero.

Try it on the list ( 3 -2 4 -6 8)

Q9) Write a function that takes in a list and returns the 3rd to last element IF the list has at least five elements, otherwise return NIL (e.g. given (7 6 4 2 1 9) return 2, and given (9 1 2 4) return NIL).

Q10) Given this function: (defun my\_length (x) (rec\_length x 0)). Write rec\_length recursively so that my\_length returns the length of x.

Q11) Create an array A with 10 elements, and set its successive members to

 2 4 6 . . . 20

Q12) Write a Lisp function average that averages a list of arguments, first checking to see whether they are all numbers, and returning nil if they aren't, e.g.

> (average '(7 6 13))

8.66

> (average '(7 6 apple))

NIL

Q13) Write a Lisp function duplicate-entries that takes a list and returns true if it contains multiple copies of some entry, and nil otherwise.Be sure to handle nested lists properly, e.g.

> (duplicate-entries '(a b a c d))

T

> (duplicate-entries '(a b (a) c d))

NIL

> (duplicate-entries '((a b) b c (a b))

T

Q14) Now write a Lisp function REMOVE-DUPS that takes a list and returns the list that is identical except for having all duplicate entries removed; leave in only the first copy of each entry, e.g.,

> (remove-dups '(a b (a) c d))

(A B (A) C D)

> (remove-dups '((a b) b c (a b) (a b))

((A B) B C)

Q15) Perform some operations like, searching, removing, counting or filtering specific elements without writing explicit loops.

(write (count 7 '(1 5 6 7 8 9 2 7 3 4 5)))

(terpri)

(write (remove 5 '(1 5 6 7 8 9 2 7 3 4 5)))

(terpri)

(write (delete 5 '(1 5 6 7 8 9 2 7 3 4 5)))

(terpri)

(write (substitute 10 7 '(1 5 6 7 8 9 2 7 3 4 5)))

(terpri)

(write (find 7 '(1 5 6 7 8 9 2 7 3 4 5)))

(terpri)

(write (position 5 '(1 5 6 7 8 9 2 7 3 4 5)))