Lab 4

Problem 1

* Factorial

(defun **factorial** (num)

    (if (= 0 num)

        1

        (\* num (factorial (- num 1)))

    )

)

* Test Cases:

(print (factorial 1))

*; 1*

(print (factorial 10))

*; 3628800*

(print (factorial 15))

*; 1307674368000*

Problem 2

* Fibonacci

(defun **fibonacci** (num)

    (defun **fib** (num)

        (cond

            ((= num 0) 0)

            ((= num 1) 1)

            (t (+ (fib (- num 1)) (fib (- num 2))))

        )

    )

    (if (= 0 num)

        nil

        (append (fibonacci (- num 1)) (list (fib num)))

    )

)

* Test Cases:

(print (fibonacci 11))

*; (1 1 2 3 5 8 13 21 34 55 89)*

(print (fibonacci 3))

*; (1 1 2)*

(print (fibonacci 21))

*; (1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946)*

Problem 3

* Is Member

(defun **is\_member** (elem lst)

    (cond

        ((null lst) nil)

        ((eql elem (car lst)) lst)

        (t (is\_member elem (cdr lst)))

    )

)

* Test Cases:

(print (is\_member 4 '(1 (2 3) 4 5 6)))

*; (4 5 6)*

(print (is\_member 4 '(1 (2 3) 4.0 5 6)))

*; NIL*

(print (is\_member '(2 3) '(1 (2 3) 4 5 6)))

*; NIL*

(print (is\_member 7 '(1 (2 3) 4 5 6)))

*; NIL*

Problem 4

* Trim Head

(defun **trim\_head** (lst n)

    (cond

        ((null lst) nil)

        ((zerop n) lst)

        (t (trim\_head (cdr lst) (- n 1)))

    )

)

* Test Cases:

(print (trim\_head '(1 2 3 4 5 6) 3))

*; (4 5 6)*

(print (trim\_head '(1 2 3) 3))

*; NIL*

(print (trim\_head '(1) 3))

*; NIL*

(print (trim\_head nil 3))

*; NIL*

Problem 5

* Trim Tail

(defun **trim\_tail** (lst n)

    (defun **trim\_head** (lst n)

        (cond

            ((null lst) nil)

            ((= 0 n) lst)

            (t (trim\_head (cdr lst) (- n 1)))

        )

    )

    (reverse (trim\_head (reverse lst) n))

)

* Test Cases:

(print (trim\_tail '(1 2 3 4 5 6) 3))

*; (1 2 3)*

(print (trim\_tail '(1 2 3) 3))

*; NIL*

(print (trim\_tail '(1) 3))

*; NIL*

(print (trim\_tail nil 3))

*; NIL*

Problem 6

* Count Atoms

(defun **count\_atoms** (lst)

    (let (

        (head (car lst))

        (tail (cdr lst))

    )

        (if (null lst)

            0

            (+

                (if (atom head)

                    1

                    (count\_atoms head)

                )

                (count\_atoms tail)

            )

        )

    )

)

* Test Cases:

(print (count\_atoms '(1 2 nil (three 4) 5 (6 (seven 8) nine) 10)))

*; 11*

(print (count\_atoms nil))

*; 0*

(print (count\_atoms '(1 2 3)))

*; 3*

Problem 7

* Add

(defun **add** (num1 num2)

    (if (= 0 num2)

        num1

        (add (1+ num1) (1- num2))

    )

)

* Test Cases:

(print (add 5 7))

*; 12*

(print (add 7 5))

*; 12*

Problem 8

* Reverse

(defun **my\_reverse** (lst)

    (cond

        ((null lst)

            nil

        )

        (t

            (append (my\_reverse (cdr lst)) (list (car lst)))

        )

    )

)

* Test Cases:

(print (my\_reverse nil))

*; nil*

(print (my\_reverse '(1 2 3 4 5)))

*; (5 4 3 2 1)*

(print (my\_reverse '(1 2 (3 4) 5)))

*; (5 (3 4) 2 1)*

Problem 9

* Is Member

(defun **is\_present** (elem lst)

    (let (

        (head (car lst))

        (tail (cdr lst))

    )

        (cond

            ((null lst)

                nil

            )

            ((listp head)

                (is\_present elem head)

            )

            ((eql (car lst) elem)

                t

            )

            (t

                (is\_present elem (cdr lst))

            )

        )

    )

)

* Test Cases:

(print (is\_present nil '(1 2 3 4 5)))

*; nil*

(print (is\_present nil '(1 2 () 3 4 5)))

*; T*

(print (is\_present 3 '(1 2 3 4 5)))

*; T*

(print (is\_present 3 '(1 2 (3 4) 5)))

*; T*

(print (is\_present 6 '(1 2 3 4 5)))

*; NIL*

Problem 10

* Squash

(defun **squash** (lst)

    (let (

        (head (car lst))

        (tail (cdr lst))

    )

        (cond

            ((null lst)

                nil

            )

            ((atom head)

                (append (list head) (squash tail))

            )

            ((listp head)

                (append (squash head) (squash tail))

            )

            (t

                (squash tail)

            )

        )

    )

)

* Test Cases:

(print (squash '(1 nil (two 3) 4 (5 (6 7) 8) 9 nil)))

*; (1 nil two 3 4 5 6 7 8 9 nil)*