**Lab 8**

Problem 1

* Define:

(defmacro **define** (func &rest body)

    (let (

        (name (car func))

        (params (cdr func))

    )

        `(defun **,name** ,params

            ,@body

        )

    )

)

* Test Cases:

(define (add a b c)

    (+ a b c)

)

(print (add 1 2 3))

*; 6*

(define (hello\_world)

    (print 'hello\_world)

    (print 'goodbye\_world)

)

(hello\_world)

*; HELLO\_WORLD*

*; GOODBYE\_WORLD*

Problem 2

* Do Times:

(defmacro **do\_times** (head &rest body)

    (let (

        (var (car head))

        (cnt (cadr head))

        (res (caddr head))

    )

        `(do (

            (,var 0 (1+ ,var))

        )(

            (>= ,var ,cnt)

            ,res

        )

            ,@body

        )

    )

)

* Test Cases:

(format t "~%return: ~a~%" (do\_times (num 5 10)

    (format t "~a " num)

))

*; 0 1 2 3 4*

*; return: 10*

(format t "~%return: ~a~%"  (do\_times (num 2)

    (format t "~a " num)

))

*; 0 1*

*; return: nil*

(format t "~%return: ~a~%"  (do\_times (num 3 num)

    (format t "~a " num)

))

*; 0 1 2*

*; return: 3*

Problem 3

* Reverse:

(defmacro **rev** (lst)

    (defun **rev\_helper** (lst)

        (if (null lst)

            nil

            (nconc (rev\_helper (cdr lst)) (rplacd lst nil))

        )

    )

    `(setf ,lst (rev\_helper ,lst))

)

* Test Cases:

(defvar lst1 '(1 2 3 4 5))

(rev lst1)

(print lst1)

*; (5 4 3 2 1)*

(defvar lst2 '(one 2 three))

(rev lst2)

(print lst2)

*; (three 2 one)*

(defvar lst3 nil)

(rev lst3)

(print lst3)

*; nil*