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:

```lisp
(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:

```lisp
(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
```