Lab 4

Problem 1
- Factorial

```
(defun factorial (num)
    (if (= 0 num)
        1
        (* num (factorial (- num 1)))
    )
)
```

Problem 2
- Fibonacci

```
(defun fibonacci (num)
    (cond
        ((= num 0) 0)
        ((= num 1) 1)
        (t (+ (fibonacci (- num 1)) (fibonacci (- num 2))))
    )
)
```

Problem 3
- Is Member

```
(defun is_member (elem lst)
    (cond
        ((null lst) nil)
        ((eql elem (car lst)) lst)
        (t (is_member elem (cdr lst)))
    )
)
```

Problem 4
- Trim Head

```
(defun trim_head (lst n)
    (cond
        ((null lst) nil)
        ((zerop n) lst)
        (t (trim_head (cdr lst) (- n 1)))
    )
)
```

```
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))
  )

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)
              )
          )
      )
  )
```

Problem 7
  - Add

```
(defun add (num1 num2)
    (if (= 0 num2)
        num1
        (add (1+ num1) (1- num2))
    )
)
```

Problem 8
  - Reverse

```
(defun my_reverse (lst)
    (cond
        ((null lst)
            nil
        )
        (t
            (append (my_reverse (cdr lst)) (list (car lst)))
        )
    )
)
```

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))
            )
        )
    )
)
```

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)
            )
        )
    )
)
```