Andrew Morris
CSCI 330
2-15-16

Part 1:

```
(defun foo(s)
        (cond ((null s) 1)
        ((atom s) 0)
        (t (max (+ (foo (first s)) 1) (foo (rest s))))))
```

1)
foo counts the maximum amount of list indirection in the parameter s.
So, if s is a list of lists where each sub list contains atoms, then foo will evaluate to 1. If s is a list of lists where an element of the list is a list, then foo evaluates to two.

```
(defun myfunction(n l)
(if (null l) '() (if (= n 1) (car l) (myfunction (- n 1) (cdr l)))))
```

2)
Myfunction returns the nth element of a list. It recursively calls on n-1 until n==1, then it returns the first element of that list, which is the nth element.

```
(defun choose(n k)
(if (or (= k 0) (= n k)) 1 (+ (choose (- n 1) k) (choose (- n 1) (- k 1)))))
```

3)
Choose returns the possible combinations of k items from n items.

Part II:

1)

```
(defun power(n m)
        (if (= m 0) 1
        (* n (power n (- m 1)))))
```

2)

```
(defun replace_elmt(a_list list_elmt new_elmt)
        (cons (if (eq (first a_list) list_elmt) new_elmt (first a_list))
                (if (eq (rest a_list) nil) '()
                        (replace_elmt (rest a_list) list_elmt new_elmt))))
```

3)

```
(defun add1(my_list)
        (if (= (mod (length my_list) 2) 0) my_list
                (cons (car my_list)
                (if (= (length my_list) 1) (+ (car my_list) 1)
                        (add1 cdr (reverse ( cdr (reverse my_list)))))
                        (car (reverse my_list))))))
```

4)

```
(defun add1(my_list)
	(if (= (mod (length my_list) 2) 0) my_list
		(if (= (length my_list) 1) (+ (car my_list) 1)
			(cons (cons (car my_list)
				(add1 (cdr (reverse ( cdr (reverse my_list))))))
				(car (reverse my_list))))))
```

5)

```
(defun compare(list1 list2)
	(if (and (eq list1 '()) (eq list2 '()))
		0
		(if (eq (car list1) '())
			1
			(if (eq (car list2) '())
				-1
				(compare (cdr list1) (cdr list2))
			)
		)
	))
```