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))

)

)

))