;;Jacob Steckel

;;CSC345

;;Homework 1

;;sum function

(defun sum (n1 n2)

"adds two non negative integers"

(if (zerop n1) n2

(sum (1- n1) (1+ n2))))

;;replace

(defun my-replace (e1 e2 L)

"Returns a new version of L where e1 is replaced with e2"

(cond

((null L) nil)

((listp (first L))

(if (equal e1 (first L))

(cons e2 (my-replace e1 e2 (rest L))) ;cons e2 onto recursive call on rest L

(if (member e1 (first L) :test #'equal)

(if (equal e1 (first (first L)))

(cons (cons e2 (my-replace e1 e2 (rest (first L)))) ;cons e1 and call my-replace

(my-replace e1 e2 (rest L))) ;on rest of first L. cons that to rest L

(cons (cons (first (first L)) (my-replace e1 e2 (rest (first L))))

(my-replace e1 e2 (rest L))))

(cons (first L) (my-replace e1 e2 (rest L))))))

((equal e1 (first L))

(cons e2 (my-replace e1 e2 (rest L))))

(t (cons (first L) (my-replace e1 e2 (rest L))))))

;;fibonacci

(defun fibonacci (n)

"Returns the n(th) value in the Fibonacci sequence"

(cond

((eq n 1) 0)

((eq n 2) 1)

(t (+ (fibonacci (- n 1)) (fibonacci (- n 2))))))

;;fibonacci tail recursion

(defun fibonacci-TR (n)

(labels ((calc-fib (n a b)

(if (= n 0)

a

(calc-fib (- n 1) b (+ a b)))))

(calc-fib n 0 1)))