Ali Ghaleb

11/27/2021

COSC 341

Assignment 3

**Any comments regarding the assignment which you want to address**

* + **Number 10 is hard.**
  + **I couldn’t find any code on number 13, so I just got answers from stack overflow.**
  + **Extra credit question looked easy.**
  + **Overall, some of the questions were straightforward and some were really hard to get the answers for I spent so many hours trying to figure them out. Like 11 and 10.**

**Source code as appendix and Sample Runs**

1. **Function Name: dispnth.**

**Function Description: a function to display the n-th element of the list, you may assume that input list is always longer than n.**

**Source Code:**

(defun dispnthelper(L n helper)

(if(= n helper) (first L)

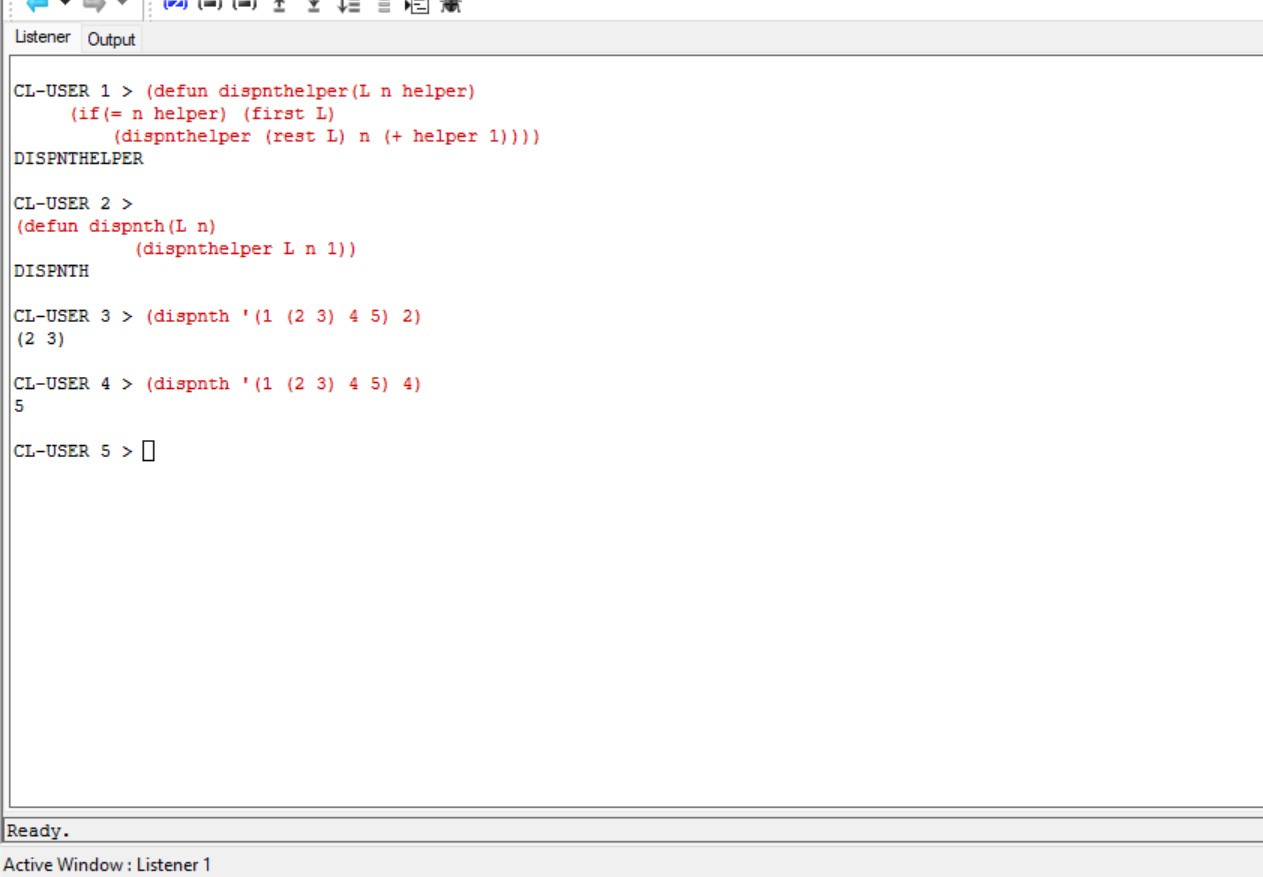
(dispnthelper (rest L) n (+ helper 1))))

(defun dispnth (L n)

(dispnthelper L n 1))

**(dispnth '(1 (2 3) 4 5) 2)**

**Sample Run:**

****

1. **Function Name: delth.**

**Function Description: a function to delete the n-th element of the list, you may assume that input list is always longer than n.**

**Source Code:**

(defun delnthelper(L n helper)

(if(null L) nil

(if(= n helper) (delnthelper (rest L) n (+ helper 1))

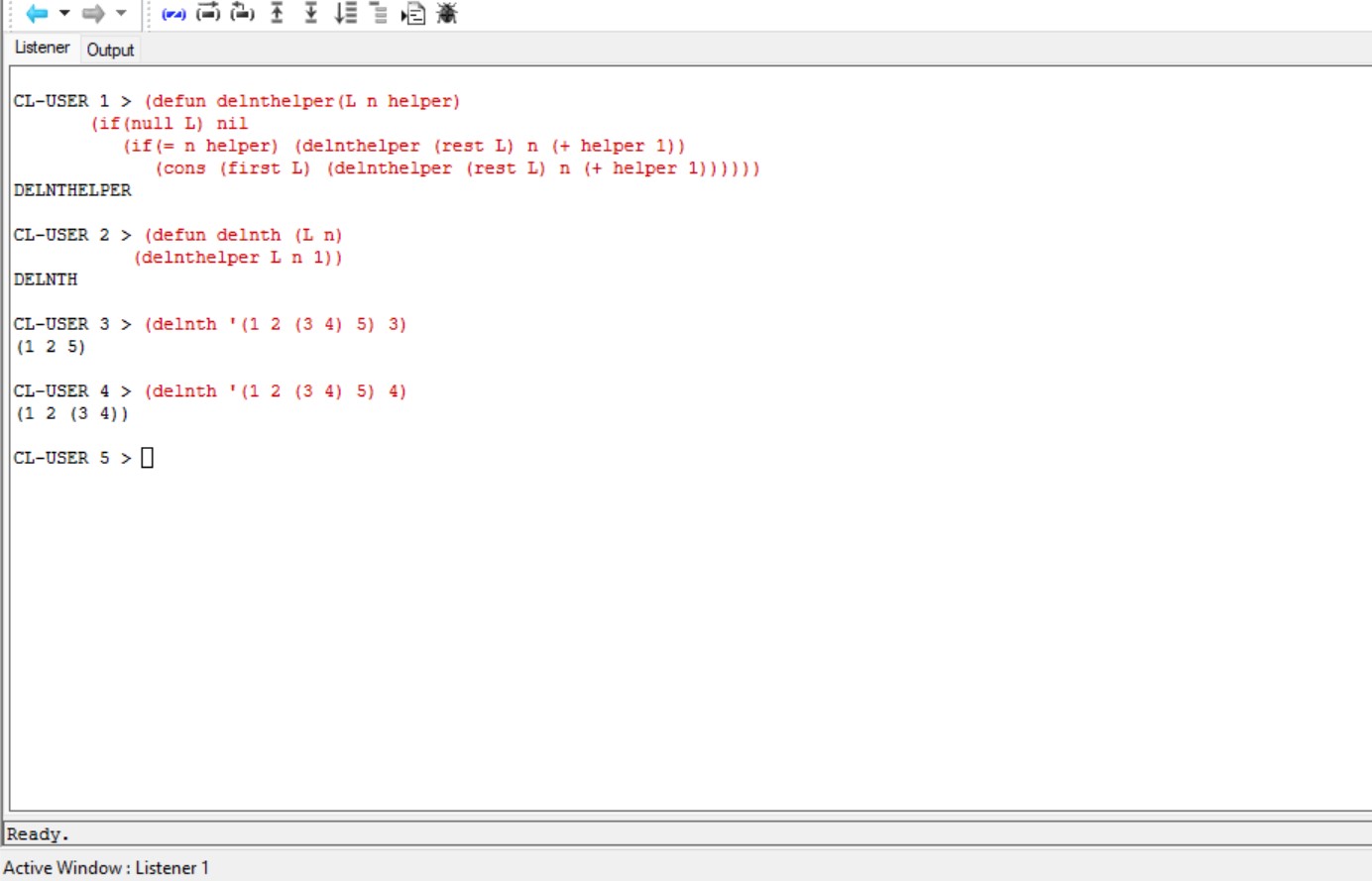
(cons (first L) (delnthelper (rest L) n (+ helper 1))))))

(defun delnth (L n)

(delnthelper L n 1))

**(delnth '(1 2 (3 4) 5) 3)**

**Sample Run:**



1. **Function Name: remv.**

**Function Description: a function remv to remove elements from a list (including all multiple appearance)**

**Source Code:**

(defun remv(a L)

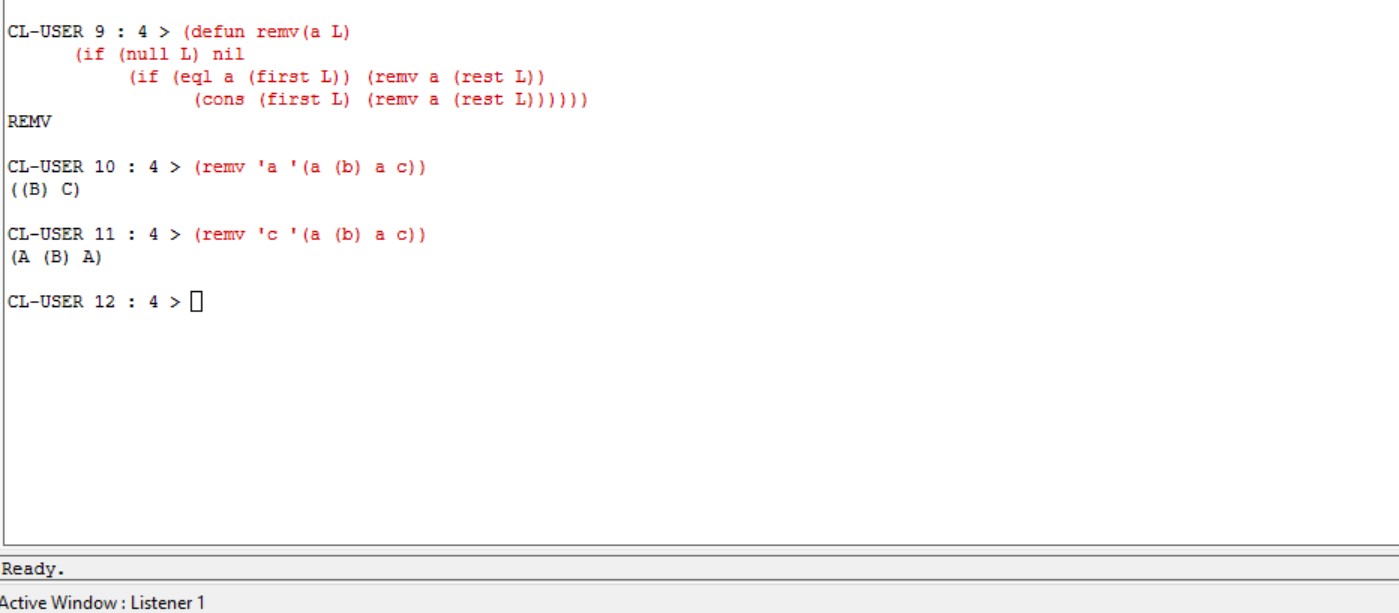
(if (null L) nil

(if (eql a (first L)) (remv a (rest L))

(cons (first L) (remv a (rest L))))))

**(remv 'a '(a (b) a c))**

**Sample Run:**

****

1. **Function Name: remv2.**

**Function Description: a function remv2 to remove give list elements from a list (including multiple appearance)**

**Source Code:**

(defun remv2(a L)

(if (null L) nil

(if (equal a (first L)) (remv2 a (rest L))

(cons (first L) (remv2 a (rest L))))))

**(remv2 '(a b) '(a b (a b) c))**

**Sample Run:**

**Graphical user interface, text, application

Description automatically generated**

1. **Function Name: remvdub.**

**Function Description: a function remvdub to remove duplicate elements from a list.**

**Source Code:**

(defun remv(a L)

(cond ( (null L) nil )

( (eql a (first L)) (remv a (rest L)))

( t (cons (first L) (remv a (rest L))) )

)

)

(defun remvdub(L)

(cond ( (null L) nil)

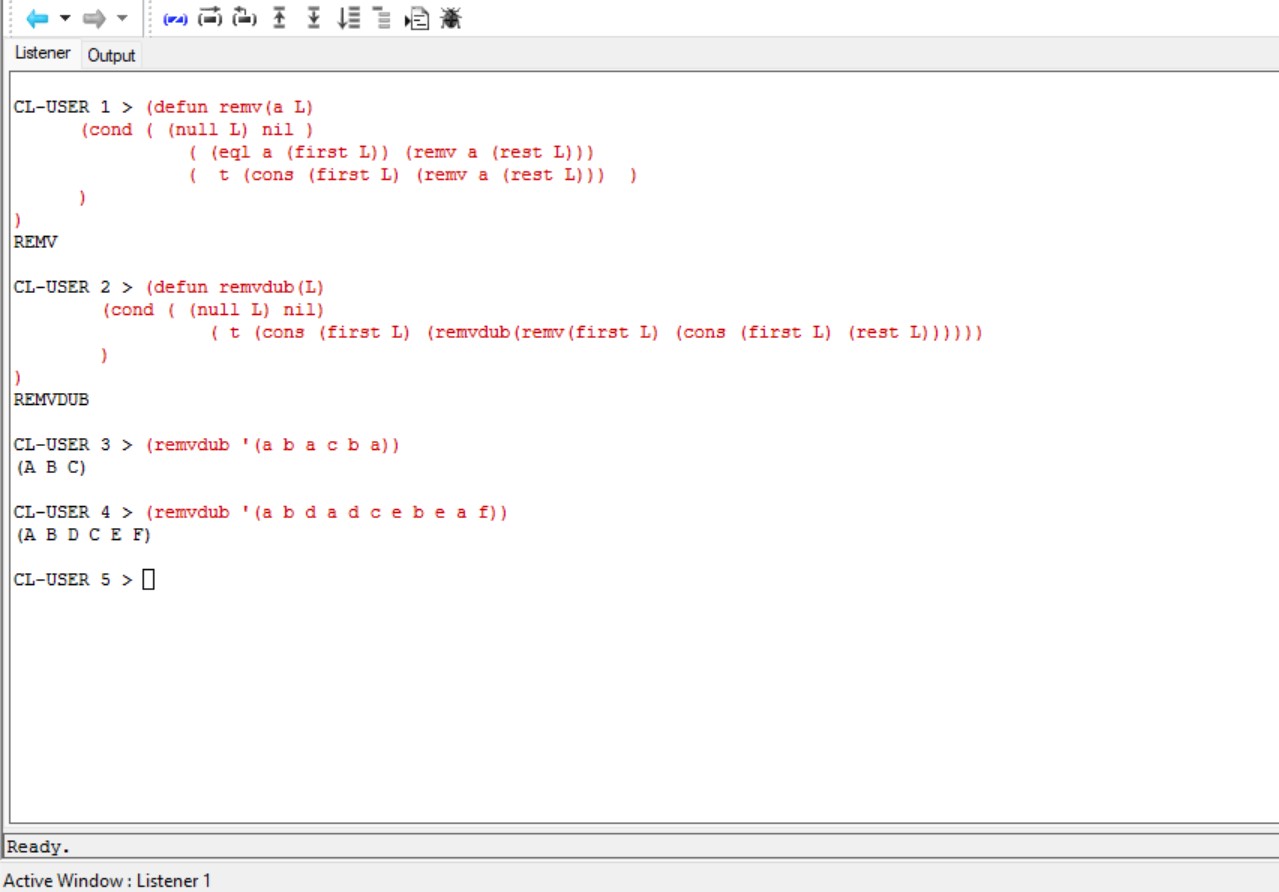
( t (cons (first L) (remvdub(remv(first L) (cons (first L) (rest L))))))

)

)

(remvdub '(a b a c b a))

**Sample Run:**

****

1. **Function Name: remvdub2.**

**Function Description: a function remvdub2 to remove duplicate elements (single elements or lists) from a list.**

**Source Code:**

(defun remv(a L)

(cond ( (null L) nil )

( (equal a (first L)) (remv a (rest L)))

( t (cons (first L) (remv a (rest L))) )

)

)

(defun remvdub2(L)

(cond ( (null L) nil)

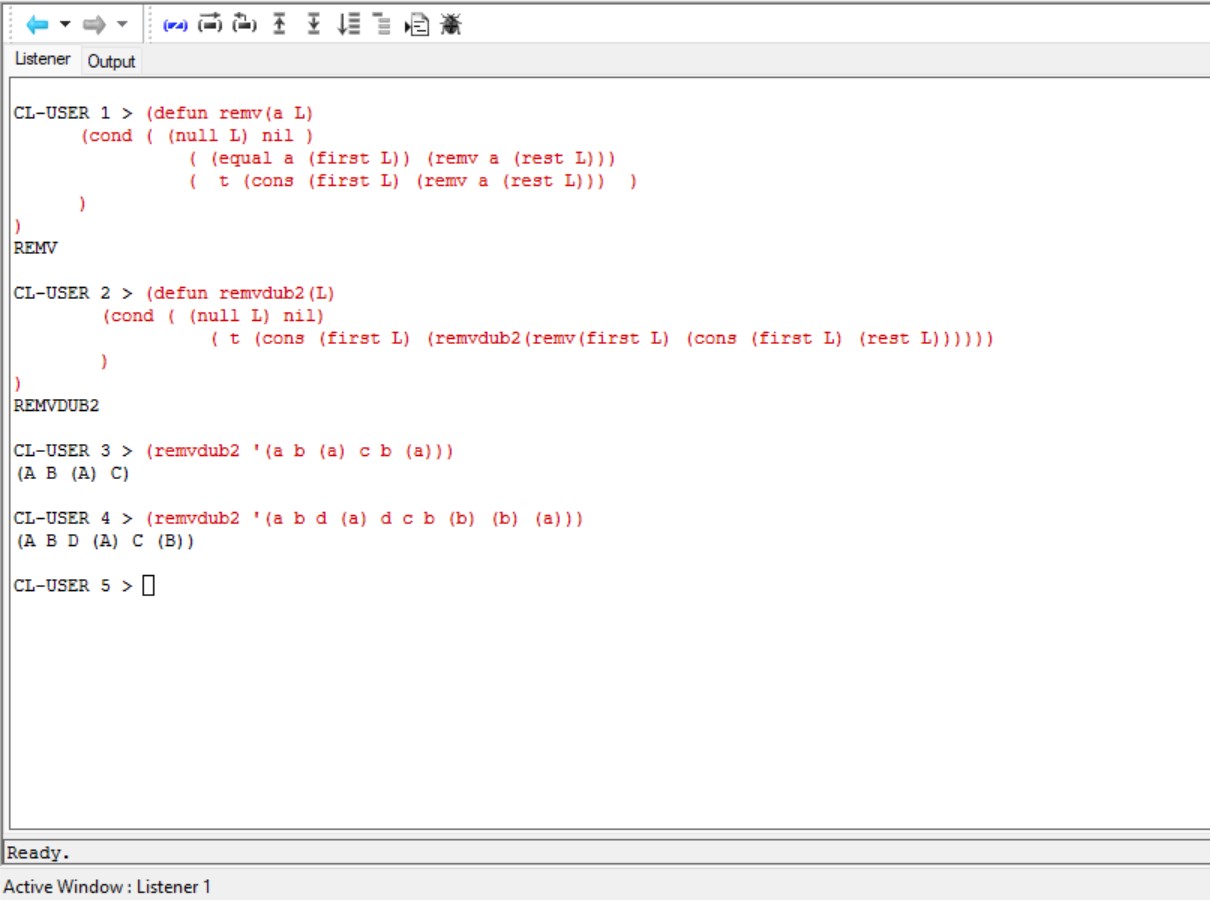
( t (cons (first L) (remvdub2(remv(first L) (cons (first L) (rest L))))))

)

)

(remvdub2 '(a b (a) c b (a)))

**Sample Run:**

****

1. **Function Name: min2.**

**Function Description: a function min2 to compute the second smallest of number of an integer list. You may assume the list has at least 2 numbers and all numbers are distinct.**

**Source Code:**

(defun minL(L)

(if(null(rest L)) (first L)

(if(> (first L) (first(rest L))) (minL (rest L))

(minL(cons (first L) (rest(rest L)))))))

(defun remminL(L value)

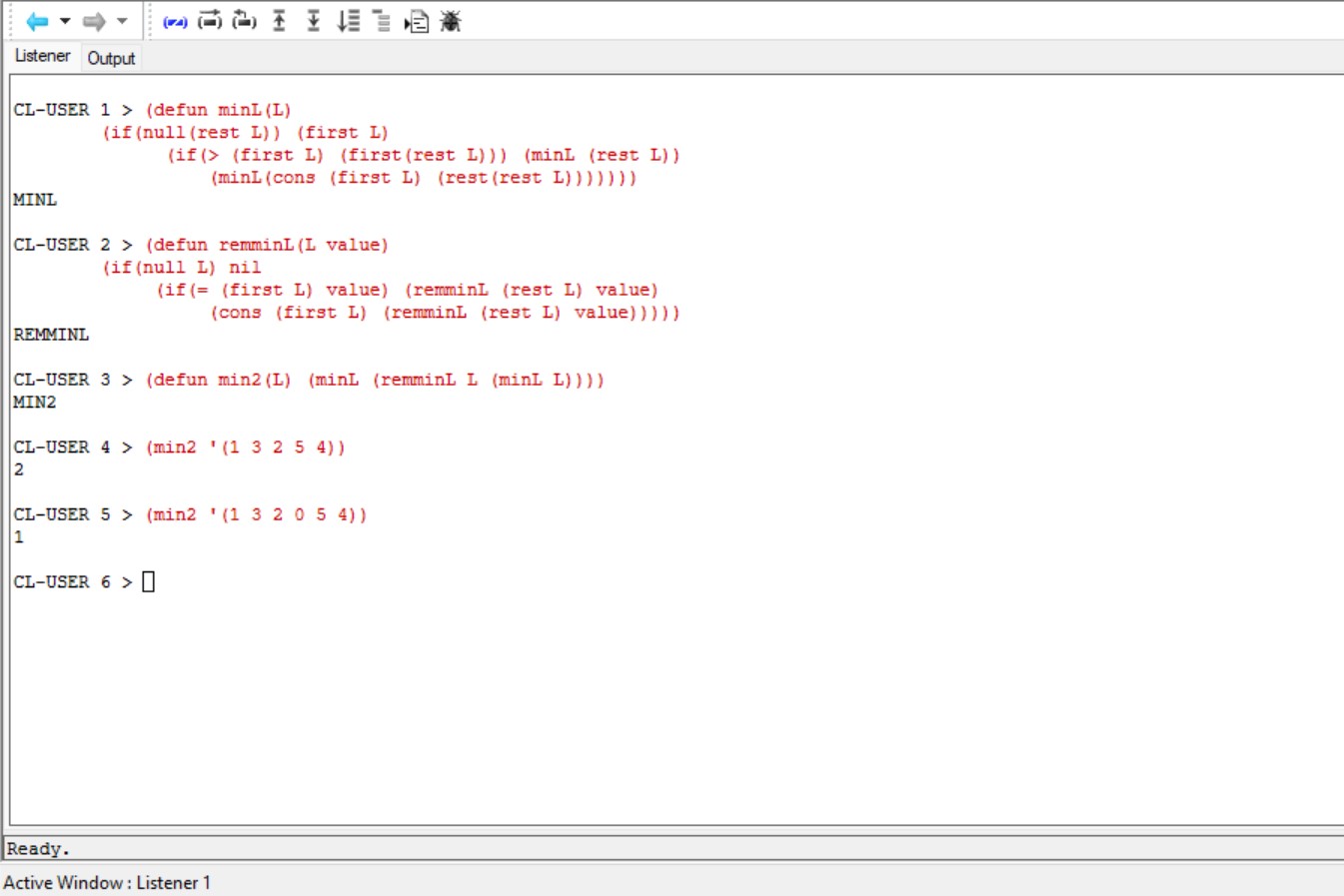
(if(null L) nil

(if(= (first L) value) (remminL (rest L) value)

(cons (first L) (remminL (rest L) value)))))

(defun min2(L) (minL (remminL L (minL L))))

**Sample Run:**

****

1. **Function Name: inde.**

**Function Description: a function inde which returns the index(start from 1) of the occurrence of a given value.**

**Source Code:**

(defun rea(n L index)

(if(null L) nil

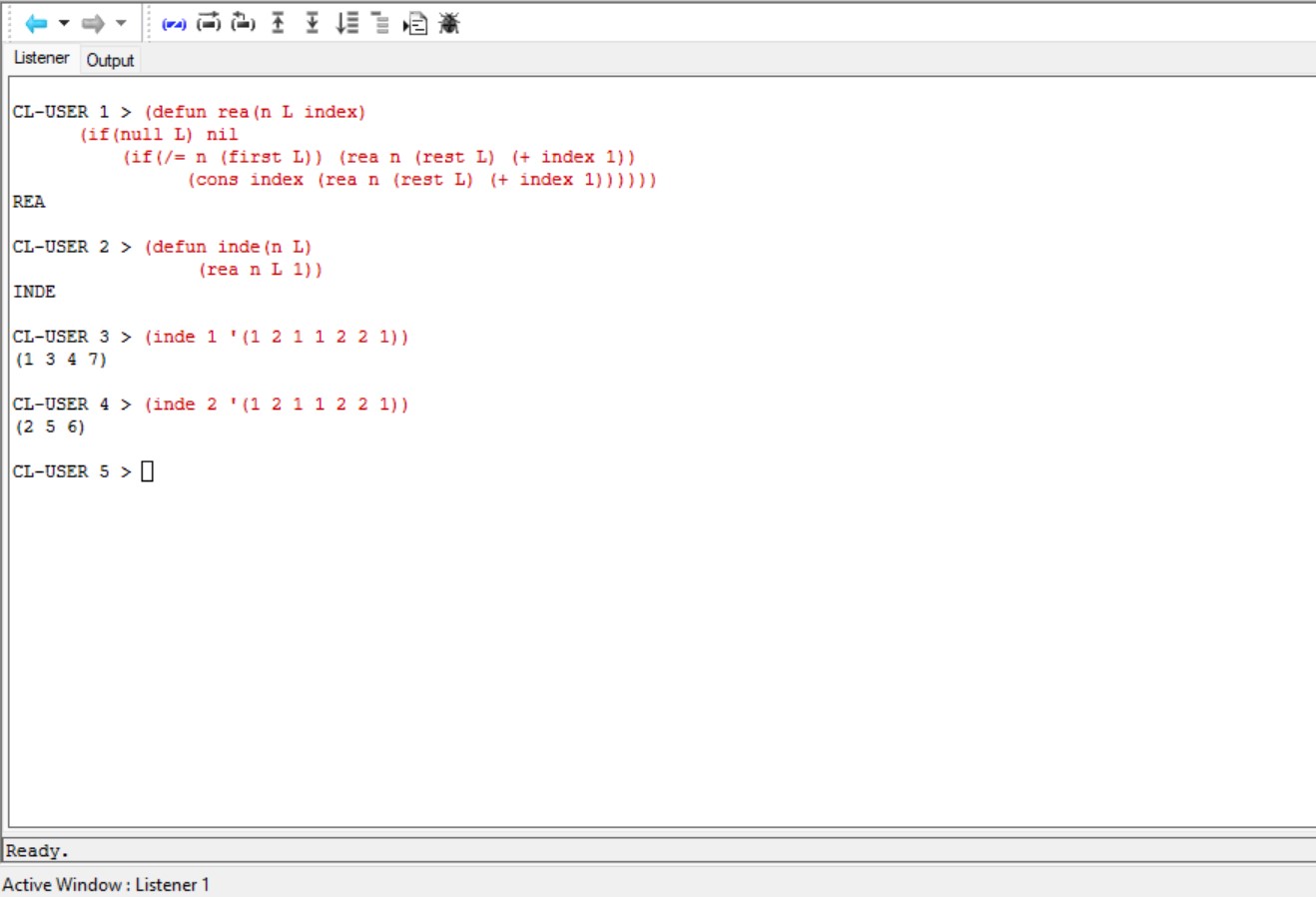
(if(/= n (first L)) (rea n (rest L) (+ index 1))

(cons index (rea n (rest L) (+ index 1))))))

(defun inde(n L)

(rea n L 1))

**Sample Run:**

****

1. **Function Name: nele.**

**Function Description: write a function nele which repeats each element in a list n times.**

**Source Code:**

(defun neleR(L n indexreset)

(if(null L) nil

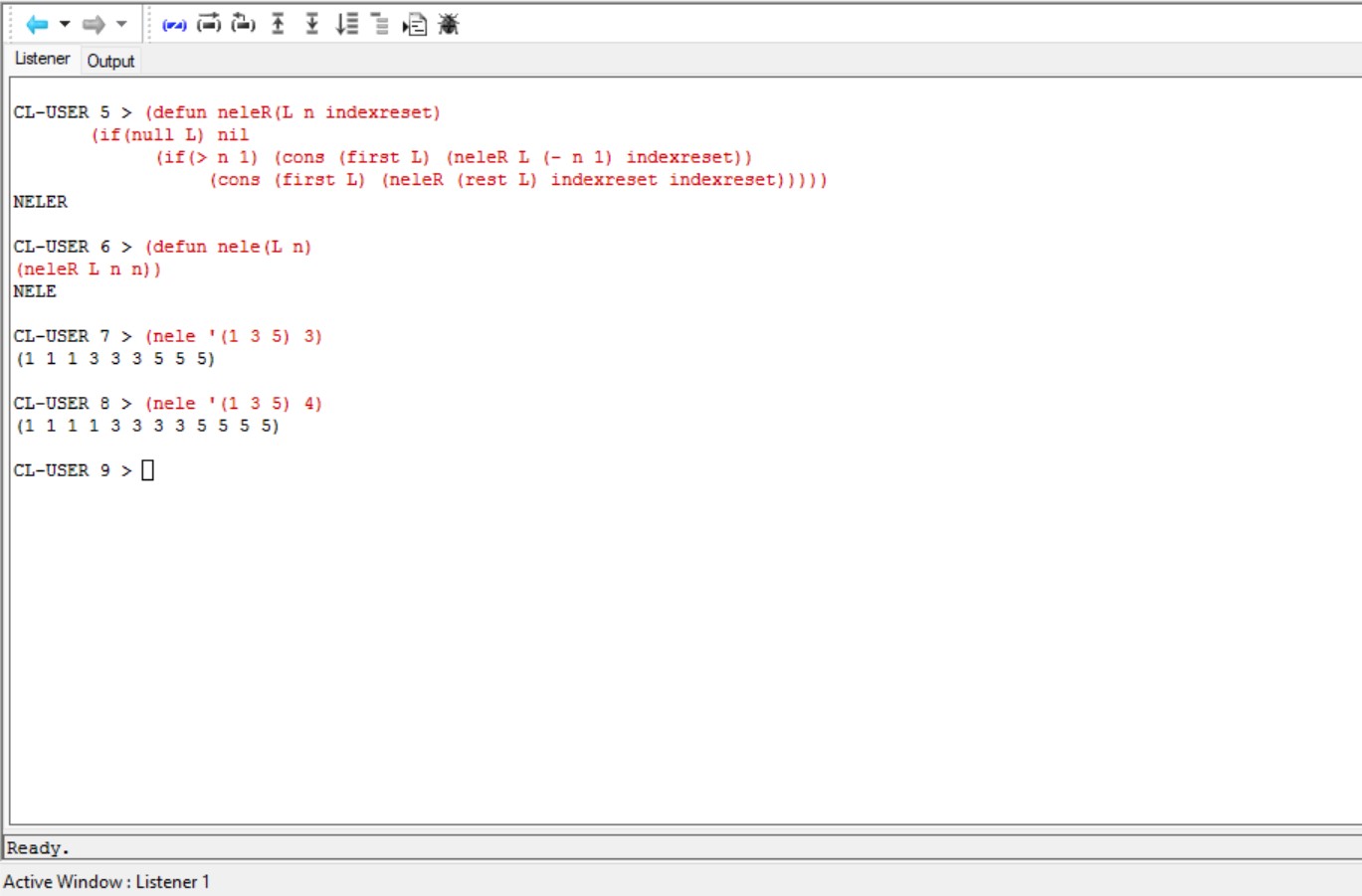
(if(> n 1) (cons (first L) (neleR L (- n 1) indexreset))

(cons (first L) (neleR (rest L) indexreset indexreset)))))

(defun nele(L n)

(neleR L n n))

**Sample Run:**



1. **Function Name: occr.**

**Function Description: a function occr to display the occurrence of an element of a list, or nil if the list is empty. Two functions only.**

**Source Code:**

(defun helper(L L1 L2)

(if(null L) (append L1 L2)

(if(null L2) (helper (rest L) nil (cons (first L 1) L2)

(if(=(first L) (first(first L2)) (helper(rest L nil (first (first (first L2))) (append(cons(first(rest(first L3))) l1)L2)))

(helper L first(first (first L2)) (first(rest(first (first L2))) L1 append (rest L2)))))))

1. **Function Name: insea**

**Function Description: a function insea to insert an element to each position of a list. Two functions only (not including possible @ or app)**

**Source Code:**

(defun ad(L1 L2)

(if(null L1) L2

(cons (first L1) (ad(rest L1) L2))))

(defun insertElement(L1 a L2)

(if(null L2) (cons (ad L1 (list a) ) (list nil))

(if(null L1) (cons (ad (list a) L2) (insertElement (list (first L2)) a (rest L2)))

(cons (ad (ad L1 (list a)) L2) (insertElement (ad L1 (list (first L2))) a (rest L2))))))

(defun insea(a L) (insertElement(nil a L)))

1. **Function Name: mergesort.**

**Function Description: implement function mergesort. (let) is optional, not required.**

**Source Code:**

(defun mergesort (list)

(labels ((merge (f s)

(cond

((null f) s)

((null s) f)

((< (first f) (first s)) (list\* (first f) (merge (rest f) s)))

((> (first f) (first s)) (list\* (first s) (merge f (rest s))))

((= (first f) (first s)) (list\* (first f)

(first s)

(merge (rest f) (rest s)))))))

(let ((len (list-length list)))

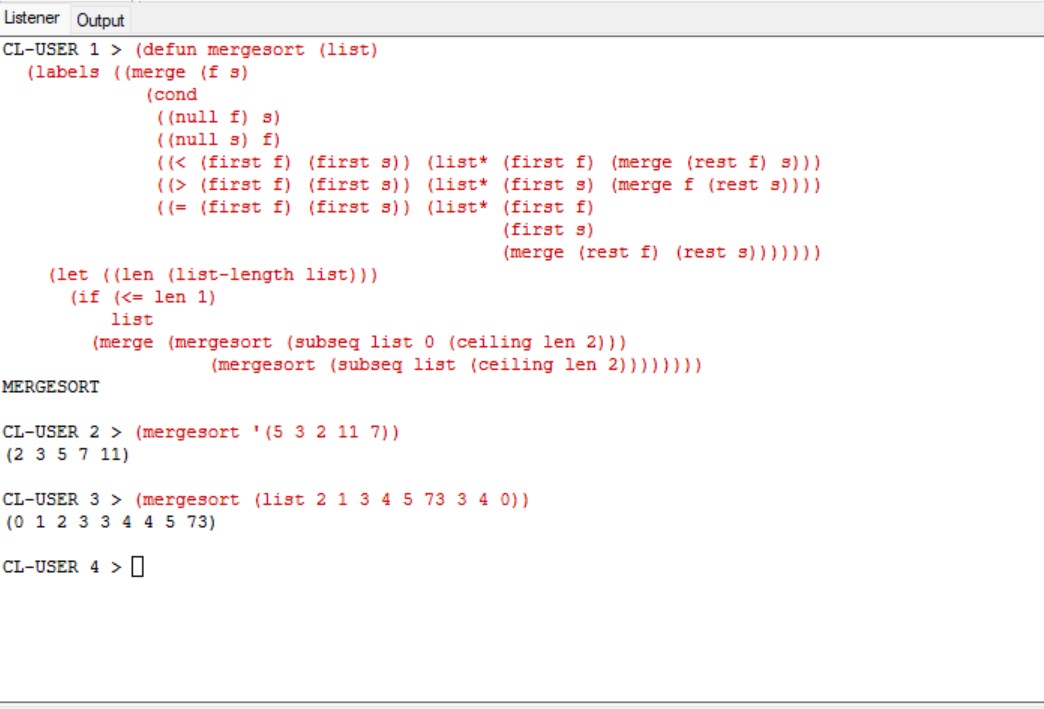
(if (<= len 1)

list

(merge (mergesort (subseq list 0 (ceiling len 2)))

(mergesort (subseq list (ceiling len 2))))))))

**Sample Run:**

****

1. **Function Name: qsort.**

**Function Description: a function that quick sorts a list.**

1. **Qs is a higher order function with anonymous function.**
2. **Qs applies a standard (filter P L) function.**
3. **Total two functions, filter and qort**

**Source Code:**

(defun qsort(L)

(cond

((null L) nil)

(t

(append

(qsort (list< (car L) (cdr L)))

(cons (car L) nil)

(qsort (list>= (car L) (cdr L)))))))

(defun list< (a b)

(cond

((or (null a) (null b)) nil)

((< a (car b)) (list< a (cdr b)))

(t (cons (car b) (list< a (cdr b))))))

(defun list>= (a b)

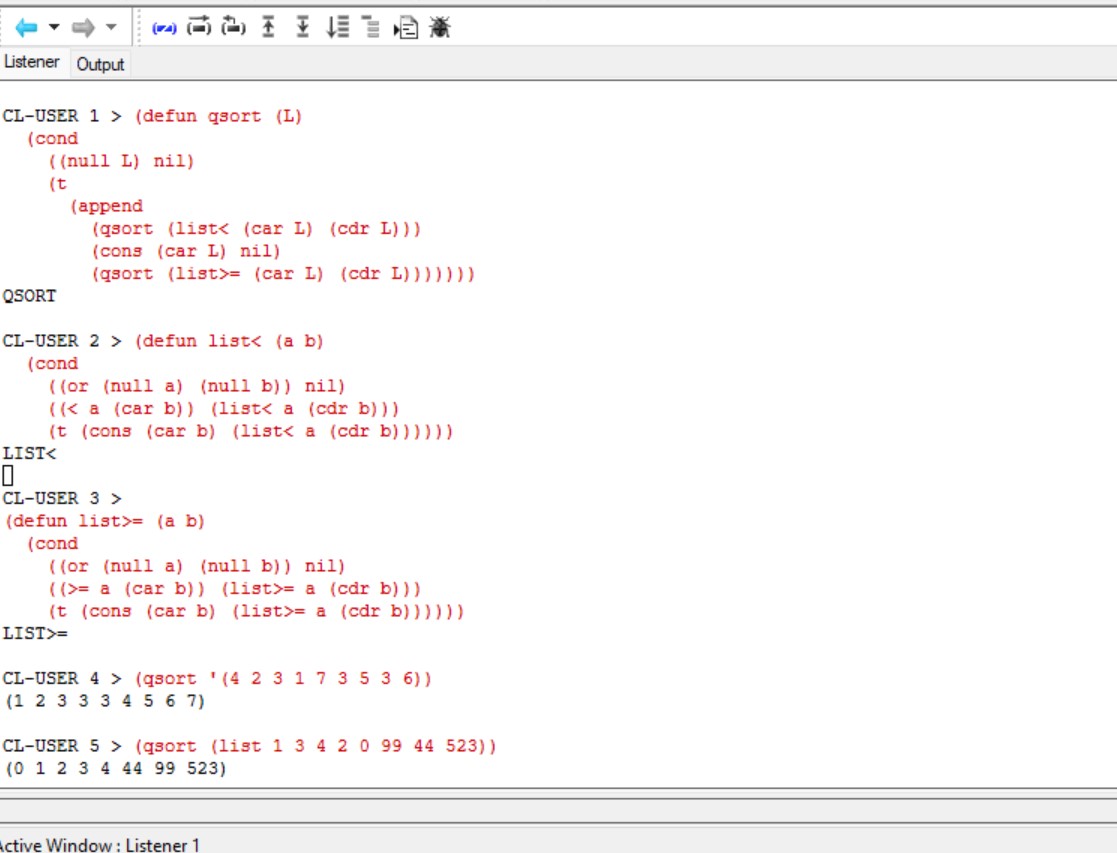
(cond

((or (null a) (null b)) nil)

((>= a (car b)) (list>= a (cdr b)))

(t (cons (car b) (list>= a (cdr b))))))

**Sample Run:**

****

1. **Function Name: primes.**

**Function Description: write a higher order function primes applying standard filter to find all prime numbers from 2 to a given n.**

**Source Code:**

**(defun helper(n,b)**

**(if(n=b) true**

**(if(=(mod n b) 0) false**

**(helper(n (+ b 1))))))**

**(defun isPrime(n)**

**(helper(n 2)))**