

Homework Solutions: Problem Set Homework 2

Due Tuesday September 8, 2009 in class

Be sure to follow the guidelines for Programming Assignments. Since these problems are simple, you may skip the Statement of the Problem, Approach and Algorithm - Program Flow sections and give an overall conclusion for the entire programming assignment. Make sure you give me a listing of each program and at least three test cases for each. You may wish to use the "dribble" option in the file menu of XLISP to capture the screen as you test your functions.

Problems 3-1, 3.3, 3.4 and 3.5 in Chapter 3 of Winston and Horn's LISP (see Below)

Problems 4-1, 4-5 and 4-6 in Chapter 4 of Winston and Horn's LISP (see Below)

3.1 Define EXCHANGE, a procedure that returns the elements of a two-element in reverse order.

```
Lisp-> (exchange '(adam eve))  
(eve adam)
```

```
3-1 (defun EXCHANGE (lis) (cond  
  ((null lis) nil)  
  ((= (length lis) 2) (list (cadr lis) (car lis)))  
  (t 'Error)))
```

3.3 Define ROTATE-LEFT, a procedure that takes a list as its argument and returns a new list in which the former first element becomes the last.

```
Lisp-> (rotate-left '(a b c))  
(b c a)  
Lisp-> (rotate-left (rotate-left '(a b c)))  
(c a b)
```

```
3-3 (defun ROTATE-LEFT (lis) (append (cdr lis) (list (car lis))))
```

3.4 Define ROTATE-RIGHT which is like ROTATE-LEFT except in the opposite direction. You might want to use BUTLAST.

```
3-4 (defun ROTATE-RIGHT (lis) (append (last lis) (butlast lis)))
```

3.5 A palindrome is a list that has the same sequence of elements when read from right-to-left that it does from left-to-right. Define PALINDROME such that it takes a list as its argument and returns a plaindrome that is twice as long.

```
3-5 (defun PALINDROMIZE (lis) (cond  
  ((null lis) nil)  
  (t (append lis (reverse lis)))))
```

4.1 Define DIVISIBLE-BY-THREE, a predicate that determines if an integer is divisible by three. You may want to use REM.

```
Lisp-> (divisible-by-three 3)
T
Lisp-> (divisible-by-three 4)
Nil

> (DEFUN DIVISIBLE-BY-THREE (n) (cond
      ((eq n 0)nil)
      ((eq (rem n 3) 0) t)
      ('else nil)))
DIVISIBLE-BY-THREE
> (mapcar 'divisible-by-three '(1 2 3 4 5 6 7 8 9 10))
(NIL NIL T NIL NIL T NIL NIL T NIL)
> (divisible-by-three 0)
NIL
```

4.5 Write IF forms that are equivalent to (ABS X), (MIN A B), and (MAX A B), where X, A, and B are numbers.

4.6 Compose COND forms that are equivalent to (NOT U), (OR X Y Z), and (AND A B C).

XLISP-PLUS version 3.02  
Portions Copyright (c) 1988, by David Betz.  
Modified by Thomas Almy and others.  
XLISP-STAT Release 3.52.4 (Beta).  
Copyright (c) 1989-1998, by Luke Tierney.

```
>(pprint (function-lambda-expression #'exchange))
(LAMBDA (LIS)
  (COND ((NULL LIS) NIL)
        ((= (LENGTH LIS) 2)
         (LIST (CADR LIS) (CAR LIS)))
        (T (QUOTE ERROR))))
> (exchange '())
NIL
> (exchange '(a))
ERROR
> (exchange '(a b))
(B A)
> (exchange '(a b c))
ERROR

> (pprint (function-lambda-expression #'construct))
(LAMBDA (F R) (CONS F R))
> (construct 'a 'b)
(A . B)
> (construct 'a '())
(A)
> (construct 'a '(b))
(A B)
> (construct nil '(a b))
(NIL A B)
> (construct '(a) '(b c))
((A) B C)

> (pprint (function-lambda-expression #'rotate-left))
(LAMBDA (LIS)
  (APPEND (CDR LIS) (LIST (CAR LIS))))
> (rotate-left '(a b c))
(B C A)
> (rotate-left '())
(NIL)
> (rotate-left '(a b))
(B A)

> (pprint (function-lambda-expression #'rotate-right))
(LAMBDA (LIS)
  (APPEND (LAST LIS) (BUTLAST LIS)))
> (rotate-right '(a b c))
(C A B)
> (rotate-right '(a b))
(B A)
> (rotate-right '(a))
(A)
> (rotate-right nil)
NIL
> (rotate-right '())
NIL
```

```
> (pprint (function-lambda-expression #'palindromize))
(LAMBDA (LIS)
  (COND ((NULL LIS) NIL)
        (T (APPEND LIS (REVERSE LIS)))))
> (palindromize '())
NIL
> (palindromize '(a))
(A A)
> (palindromize '(a b))
(A B B A)

> (pprint (function-lambda-expression #'f-to-c))
(LAMBDA (F) (/ (* 5 (- F 32)) 9))
> (f-to-c 32)
0
> (f-to-c 85)
29.444444444444444
> (f-to-c 212)
100
> (f-to-c 94)
34.444444444444444

> (pprint (function-lambda-expression #'c-to-f))
(LAMBDA (F) (+ (/ (* 9 F) 5) 32))
> (c-to-f 0)
32
> (c-to-f 100)
212
> (c-to-f 34.44)
93.991999999999984

> (pprint (function-lambda-expression #'palindromep))
(LAMBDA (LIS)
  (EQUAL LIS (REVERSE LIS)))
> (palindromep '(a b c b a))
T
> (palindromep '(a b c c b a))
T
> (palindromep '(a b c c c b a))
T
> (palindromep '(a b c c a))
NIL

> (pprint (function-lambda-expression #'rightp))
(LAMBDA (H S1 S2)
  (< (ABS (- (** H 2) (** S1 2) (** S2 2)))
      (* 0.02 (** H 2))))
> (rightp 5 4 3)
T
> (rightp 1.414 1 1)
T
> (rightp 5 3.9 3)
NIL
> (rightp 5 3.94 3)
T
> (rightp 5 3.93 3)
NIL
```

```
> (pprint (function-lambda-expression #'check-temp))
(LAMBDA (TEMP)
  (COND ((< TEMP 0) (QUOTE TOO_COLD))
        (> TEMP 100) (QUOTE TOO_HOT))
        (T (QUOTE OK))))
> (check-temp 0)
OK
> (check-temp 100)
OK
> (check-temp 50)
OK
> (check-temp 101)
TOO_HOT
> (check-temp -1)
TOO_COLD

> (pprint (function-lambda-expression #'mygcd))
(LAMBDA (N1 N2)
  (COND ((AND (INTEGERP N1) (INTEGERP N2))
        (GCD N1 N2))
        (T (QUOTE ERROR))))
> (mygcd 12 4)
4
> (mygcd 12.1 4.1)
ERROR
> (mygcd 'a 4)
ERROR
> (mygcd 12 'b)
ERROR
> (mygcd 12.0 4.0)
ERROR
> (exit)
```