CS3723 Pgm3 Lisp (30 points) – tentative due date: 2018-11-04

© 2018 Larry W. Clark, this document may not be copied to any other website.

Code the macros listed below and use the specified test cases.

Notes:

* You can only use the functions/macros we discussed in the LISP notes. You may also use += in your iterate.
* Your code must be executed on a **fox** server using the specified test cases. To load your code, use (load "**p3Lisp.txt**" :echo T :print T).
* To run the test cases, use (load "p3LispRun.txt" :echo T :print T)
* Turn in a zip file named LastNameFirstName.zip (no spaces) containing:
  + Your source LISP code.
  + Your log of the session (see the setup instructions). This should be a **p3Out.txt**.
  + Do not have any directories within your zip file.
* Your code must follow my LISP programming standards.

1. Code the macro, **+=**, which is passed a variable which it increments and assigns the new value. The function value returned by += should be the new value of *numericVariable.*

(**+=** *numericVariable incrementValue*)

Example:

> (setf x 1)

1

> (+= x 5)

6

CLISP sometimes gives an error like the following when you LOAD a file with that macro definition:

#<PACKAGE COMMON-LISP> is locked

if you continue (by typing 'continue'): Ignore the lock and proceed

To ignore that message, simply type

CONTINUE

2. Code the macro, **iterate**, which is based on the following:

(**iterate** *controlVariable beginValueExpr endValueExpr incrExpr bodyexpr1 bodyexpr2 ... bodyexprN*)

* **iterate** is passed a *controlVariable* which is used to count from *beginValueExpr* to *endValueExpr* (inclusive) by the specified increment.
* For each iteration, it evaluates each of the one or more body expressions.
* Since *beginValueExpr,* *endValueExpr*,and *incrExpr* are expressions, they must be **evaluated**.
* The *endValueExpr* and *incrExpr* are evaluated before processing the rest of the macro. This means the code within the user's use of the macro cannot alter the termination condition nor the increment; however, it can change the value of the *controlVariable.*
* The functional value of iterate will be T.
* You can create an intermediate variable named **endValue** for the *endValueExpr.* You can create an intermediate variable named **incValue** for the *incrExpr.* For 2 points bonus, use **gensym** to generate the name of those two variables.

Examples:

1. > (iterate i 1 5 1

(print (list 'one i))

)

(one 1)

(one 2)

(one 3)

(one 4)

(one 5)

T

2. > (setf n 5)

5

> (iterate i 1 n 1

(print (list 'two i n))

(+= i 1)

)

(two 1 5)

(two 3 5)

(two 5 5)

T

3. > (setf n 5)

5

> (iterate i 1 n 1

(print (list 'three i n))

(+= n 1)

)

(three 1 5)

(three 2 6)

(three 3 7)

(three 4 8)

(three 5 9)

T

4. > (setf n 5)

5

> (setf inc 2)

2

> (iterate i 1 n inc

(print (list 'three i n inc))

(+= inc 1)

)

(three 1 5 2)

(three 3 5 3)

(three 5 5 4)

T

Test cases:

;; test +=

(setf x 10)

(+= x 5)

(print x)

;; iterate

(iterate i 1 5 1

(print (list 'one i) )

)

(setf n 5)

(iterate i 1 n 1

(print (list 'two i n))

(+= i 1)

)

(setf n 5)

(iterate i 1 n 1

(print (list 'three i n))

(+= n 1)

)

(setf n 5)

(setf inc 2)

(iterate i 1 n inc

(print (list 'three i n inc))

(+= inc 1)

)

(setf n 5)

(setf inc 2)

(iterate i 1 (+ n 2) inc

(print (list 'four i n inc))

(+= n 1)

(+= i 1)

(+= inc 1)

)