CS3723 Pgm3 Lisp (20 points)

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 including ones we developed as exercises.
* Use (load "p3Lisp.txt" :echo T :print T) to load your source code.
* Your code must be executed on a **fox** server using the specified test cases. Larry provided a test file named p3Run.txt. Use (load "p3Run.txt" :echo T :print T).
* Turn in a zip file named *LastNameFirstName*.zip (no spaces) containing:
  + Your source LISP code named p3Lisp.txt
  + Your log of the session. This should be a .txt file named p3Out.txt
* Your code **must** follow my **LISP programming standards**.
* Any use of code from another web site will result in a 0 on this assignment, and may result in you being expelled from UTSA.

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

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

Example:

> (setf x 1 y 10)

10

> (+= x 5)

6

> (+= y x)

16

> (print (list x y))

6 16

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 typeing '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 for *beginValueExpr* to *endValueExpr* (inclusive) by the specified increment *incrExpr.*
* 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* is evaluated on every iteration. This means a use of the macro can alter the termination condition; it can also change the value of the *controlVariable.*
* The *incrExpr* is evaluated **before processing** the rest of the macro. This means the code within the user's use of the macro cannot alter the effective increment.
* The functional value of iterate will always be the final value of the *controlVariable*.
* The execution should fall out of the loop when the value of *controlVariable* > the value of *endValueExpr*.
* You can create an intermediate variable named **incValue** for the *incrExpr.* For a 2 point bonus, use **gensym** to generate the name of that variable. You will not receive any extra credit if your assignment is late.

Examples:

> ;; Iterate test 1 - simple

> (setf n 5)

5

> (iterate i 1 (+ n 2) 2

(print (list 'one i (+ n 2) 2) )

)

(ONE 1 7 2)

(ONE 3 7 2)

(ONE 5 7 2)

(ONE 7 7 2)

9

> ;; Iterate test 2 - changing ctrl variable

> (setf n 5)

5

> (iterate i 1 (+ n 2) 2

(print (list 'two i (+ n 2) 2))

(+= i 1)

)

(TWO 1 7 2)

(TWO 4 7 2)

(TWO 7 7 2)

10

;; Iterate test 3 - changing the inc should not affect iterations

> (setf n 5)

5

> (setf inc 2)

2

> (iterate i 1 (+ n 2) inc

(print (list 'three i (+ n 2) inc))

(+= inc 1)

)

(THREE 1 7 2)

(THREE 3 7 3)

(THREE 5 7 4)

(THREE 7 7 5)

9

;; Iterate test 4 - changing termination condition will affect iterations

> (setf n 5)

5

> (setf incr 2)

2

> (iterate i 1 (+ n 2) incr

(print (list 'four i (+ n 2) incr))

(+= n 1)

)

(FOUR 1 7 2)

(FOUR 3 8 2)

(FOUR 5 9 2)

(FOUR 7 10 2)

(FOUR 9 11 2)

(FOUR 11 12 2)

(FOUR 13 13 2)

15

;; Iterate test 5 - changing ctrl var and termination will affect iterations

> (setf n 5)

5

> (setf inc 2)

2

> (iterate i 1 (+ n 2) inc

(print (list 'five i (+ n 2) inc))

(+= n 1)

(+= i 1)

(+= inc 1)

)

(FIVE 1 7 2)

(FIVE 4 8 3)

(FIVE 7 9 4)

(FIVE 10 10 5)

13