CS3723 Pgm3 Lisp (10 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.
* Your code must be executed on a **fox** server using the specified test cases. Use (load "p3*abc123*.lisp" :echo T :print T). Replace *abc123* with your abc123 ID.
* Turn in your source LISP code.
* Turn in your log from the terminal window.
* 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*)

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 bodyexpr1 bodyexpr2 ... bodyexprN*)

* **iterate** is passed a *controlVariable* which is used to count for *beginValueExpr* to *endValueExpr* (inclusive).
* For each iteration, it evaluates each of the one or more body expressions.
* Since *beginValueExpr* and *endValueExpr* are expressions, they must be **evaluated**.
* Assume the increment is by 1.
* The *endValueExpr* is evaluated before processing the rest of the macro. This means the macro cannot alter the termination condition; 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.* For 2 points bonus, use **gensym** to generate the name of that variable.

Examples:

1. > (iterate i 1 5

(print (list 'one i))

)

(one 1)

(one 2)

(one 3)

(one 4)

(one 5)

T

2. > (setf n 5)

5

> (iterate i 1 n

(print (list 'two i n))

(++ i)

)

(two 1 5)

(two 3 5)

(two 5 5)

T

3. > (setf n 5)

5

> (iterate i 1 n

(print (list 'three i n))

(++ n)

)

(three 1 5)

(three 2 6)

(three 3 7)

(three 4 8)

(three 5 9)

T

Test cases:

;; test ++

(setf x 10)

(++ x)

(print x)

;; iterate

(iterate i 1 5

(print (list 'one i) )

)

(setf n 5)

(iterate i 1 n

(print (list 'two i n))

(++ i)

)

(setf n 5)

(iterate i 1 n

(print (list 'three i n))

(++ n)

)

(setf n 5)

(iterate i 1 (+ n 2)

(print (list 'four i n))

(++ n)

(++ i)

)