SW483, Summer 2017
CS131 Homework Assignment #9, “Looping Multiplier”, in Lisp!

A Looping Multiplier Program

In this assignment, you will create a multiplier program that uses a lisp for loop to perform the multiplication instead of the multiplication operator. Your multiplier program will take two integers from the user, but rather than use the normal multiplication function (the asterisk, *) to multiply them, it should do the multiplication by using the addition function (+) repeatedly in a loop. Your code only needs to work with non-negative integers (0 or positive).

For example, if the integers 10 and 3 are entered, the output from your Multiplier program should look like this:

```
Enter two integers.  Press return after each integer.
10
3
10 times 3 is
30
```

Notice that there is a carriage return after “is”. Your multiplier program should have its result on the last line by itself, so use “~%” with the format function to print a carriage return.

Your starter code for this assignment will be a file called looping_multiplier.lisp which takes two integers from the console and has a loop, but is otherwise incorrect. You can compile, run, and test the incomplete program using the same steps as for previous assignments. There is a zip archive containing the code (looping_multiplier.lisp).

To compile and run the looping multiplier program:

1. Download LoopingMultiplier.zip. It contains a Lisp source code file, looping_multiplier.lisp, a bash shell script, assignment_9_grader.sh, and a file with test cases: test_cases.csv.

2. Use the bash shell and change directories to the folder created by unpacking the .zip file.

3. Then simply execute “clisp looping_multiplier.lisp” and follow the instructions in the console to run the program.

Modifying the Multiplier Program to Make it Loop:

1. Modify the code in looping_multiplier.lisp so that it prints the results of Multiplying the first number by the second, but so that it does not use the multiplication operator (*). The text output from your program should be formatted exactly as it is above. For this part, no instructions are provided, only hints:

    a. You will probably need to have a product symbol/variable that stores the product of the multiplication. Name it anything you like (as long as it is a valid Lisp symbol).

    b. Multiplication can be performed through repeated addition. Your code should calculate the product of the multiplication by adding one of the integers a number of times equal to the other integer. For example, to get the product of 7 times 5, you can calculate 7+7+7+7+7, adding 7 to the product variable 5 times.

    c. Use a loop to perform the repeated addition. Before the loop, "setq" your product symbol the value 0, and then, within each iteration of the loop, increase the value of the product variable by firstInteger. Do this by setting the product variable to the sum of itself + firstInteger. Set up the loop so that it executes secondInteger times.

    d. After the loop completes, your product symbol should have the product of the multiplication. Print it to the console using "format".

    e. Your code only needs to work with non-negative integers (0 or positive).

Extra Credit: Using Recursion

For extra practice with recursion, try implementing the looping multiplier with a recursive function instead of a loop. Below is a sketch of how you might do this by defining a recursive function called "recursive-mult" and making a call to that function with firstInteger and secondInteger as arguments:

```
(defun recursive-mult (x y)
  (cond ((eq y 1) x)
      (T (+ … ))))
```

The first s-expression in the "cond" represents the "base case" where the second argument, y, to recursive-mult is 1. In this case the product is simply x, and x is what the cond evaluates to; no recursive call is made. If the second argument is anything other than 1, the second s-expression matches (since the test is T, which is always non-NIL). You should implement the second s-expression so that it evaluates to an addition of two numbers, one being a result of a recursive call to recursive-mult where the first argument is still x, but the second argument is slightly smaller than y.

Avoiding Infinite Loops

When working on your program and submitting your program to the autograder, you'll need to be careful not to implement a loop that executes infinitely many times. If when you run your program it seems to freeze and doesn't perform the inputs and outputs you expected, it may be stuck in an infinite loop. You can stop your program by pressing Ctrl-C.

Running the Autograder Script

Run the autograder script by executing "./assignment_9_grader.sh" from the bash shell in the zip directory. Your Lisp file must be named "looping_multiplier.lisp" with a lower-case "l" and all other letters lower-case.

The autograder script will search for any occurrences of the multiplication operator (*) in your looping_multiplier.lisp file, and will give you a zero if it finds any, so don't try to use the multiplication operator.

The script will attempt to compile your looping_printer.lisp and will test your code by executing it using the clisp interpreter on a number of inputs. When it completes, the autograder script will print your final score (out of 100) to the console. Ignore any compiler warnings.

The autograder does not run your program with a timeout, so be careful. If your program loops infinitely it will freeze the autograder script and you will need to press Ctrl-C to exit it.