

C SCI 316: Lisp Assignment 3

To be submitted *no later than*: Friday, September 30*

Program in a functional style, *without using SETF or SETQ*.

This assignment counts 0.5% towards your grade if the grade is computed using rule A. You may work with up to two other students, but each student must write up the solutions individually in his/her own file; *no two students should submit copies of the same file*. To submit, put your function definitions in a file named `your last name in lowercase-3.lsp` and leave a copy of your final version of that file in your home directory on *euclid*. [Note that if your last name is Winston (say) and your euclid username is `xxxxx_yyyy316`, then you should use the filename `winston-3.lsp`—and not `xxxxx_yyyy-3.lsp`!]

If you are working with others, include the name(s) of your partners in comment(s) at the beginning of the file. Within the file, your solutions should appear *in the same order as the problems*, and your solution to each problem should be preceded by a comment of the following form, where N is the problem number: `; Solution to Problem N`
If you cannot solve a problem, put a comment of the following form where a solution to that problem would have appeared: `; No Solution to Problem N Submitted`

Lisp must be able to LOAD the file `your last name in lowercase-3.lsp` without error. (If you have forgotten how LOAD is used in clisp, re-read the "Working with Lisp Files" subsection on pages 3 – 4 of the Assignment 2 document.) If clisp cannot LOAD your file without error you may well receive no credit for your submission—even if the only error is one missing parenthesis!

For further instructions regarding submission, see p. 3.

*NOTES: 1. The deadline will *not* be extended if euclid unexpectedly goes down after 6 p.m. on the due date. Try to submit no later than noon that day, and sooner if possible.

2. If you have difficulty with these problems, you are encouraged to see me either during my office hours on Wednesday 9/28, or by appointment if you cannot come at that time. If you come, be sure to bring a printout of the *solutions.lsp* file you created for Assignment 2; that file may give me an indication of your understanding of Lisp and help me to give explanations at the right level.
3. Functions that are incorrectly named may receive no credit—e.g., if you name the function you write for problem 5 "MNTH->INTEGER" or "MONTH-INTEGER" instead of "MONTH->INTEGER" then you may get no credit for that problem.
4. Unless you are running clisp as a subprocess of emacs, each time you write or modify a function it is a good idea to save the file, re-LOAD it into clisp, and immediately test the new or modified function. (Then whenever the file fails to LOAD you know it must be because of a syntax error in the new or modified function.)
5. Questions about these problems that are e-mailed to me will only be answered after the submission deadline.

1. Define a LISP function MIN-2 with the following properties. MIN-2 takes two arguments. If those arguments have values A and B, and A and B are numbers, then MIN-2 returns A if $A \leq B$ and returns B if $A > B$; but if A or B is not a number MIN-2 returns the symbol ERROR.

Examples: (MIN-2 21.3 7/2) => 7/2 (MIN-2 17.5 29) => 17.5
(MIN-2 5 'APPLE) => ERROR (MIN-2 '(31) '(54)) => ERROR

2. **[Exercise 4 on page 72 of Wilensky]** Write a LISP function SAFE-AVG that takes 2 arguments and returns the average of those arguments if they are numbers. If one or both of the arguments is not a number, the function should return NIL.

Examples: (SAFE-AVG 23 47.4) => 35.2 (SAFE-AVG 3 8) => 11/2
 (SAFE-AVG '(23.1) 47.3) => NIL (SAFE-AVG 'ORANGE 'PLUM) => NIL

3. **[Exercise 2 on page 72 of Wilensky]** Write a LISP predicate ODD-GT-MILLION that takes one argument, and which returns T if its argument is an odd integer greater than a million, but returns NIL otherwise. Hint: Make use of the predicate INTEGERP. Examples:

(ODD-GT-MILLION 92010231) => T (ODD-GT-MILLION 17) => NIL (ODD-GT-MILLION 92010232) => NIL
 (ODD-GT-MILLION 21/5) => NIL (ODD-GT-MILLION 1718671.24) => NIL
 (ODD-GT-MILLION '(2010231)) => NIL (ODD-GT-MILLION 'APPLE) => NIL

4. **[Exercise 3 on page 72 of Wilensky]** Re-read the discussion of the predicate MEMBER on p. 51 of Winston and Horn. Then write a LISP predicate MULTIPLE-MEMBER that takes two arguments and behaves as follows: If the first argument is a symbol or number and the second is a list, then MULTIPLE-MEMBER returns a *true* value if the first argument occurs at least twice in the second argument, and returns NIL otherwise.

Examples: (MULTIPLE-MEMBER 'A '(B A B B A C A D)) => (A C A D)
 (MULTIPLE-MEMBER 'A '(B A B B C C A D)) => (A D)
 (MULTIPLE-MEMBER 'A '(B A B B C D)) => NIL

[Notice that the behavior of MULTIPLE-MEMBER is unspecified in cases where the first argument is not a symbol or number, and in cases where the second argument is not a list. In other words, your definition may return any value or produce an evaluation error in such cases.]

5. Define a LISP function MONTH->INTEGER which takes as argument a symbol that should be the name of a month, and which returns the number of the month. For example, (MONTH->INTEGER 'MARCH) => 3 and (MONTH->INTEGER 'JUNE) => 6. If the argument is not a symbol that is the name of a month, the function should return the symbol ERROR. E.g., (MONTH->INTEGER 'C) => ERROR, (MONTH->INTEGER 7) => ERROR, and (MONTH->INTEGER '(MARCH)) => ERROR.

6. Define a LISP function SCORE->GRADE which takes a single argument, s, and returns a symbol according to the following scheme:

$s \geq 90$	A	$73 \leq s < 77$	C+
$87 \leq s < 90$	A-	$70 \leq s < 73$	C
$83 \leq s < 87$	B+	$60 \leq s < 70$	D
$80 \leq s < 83$	B	$s < 60$	F
$77 \leq s < 80$	B-		

If the argument s is not a number then the function should return NIL.

Examples: (SCORE->GRADE 86.3) => B+ (SCORE->GRADE 106) => A (SCORE->GRADE -10.1) => F
 (SCORE->GRADE 59.9) => F (SCORE->GRADE 83) => B+ (SCORE->GRADE 74) => C+
 (SCORE->GRADE 67) => D (SCORE->GRADE 87.0) => A-
 (SCORE->GRADE '(86.3)) => NIL (SCORE->GRADE 'DOG) => NIL

Solve problems 7, 8, and 9 **without** using COND, IF, WHEN, UNLESS, or CASE.

7. Define a LISP function GT with the following properties. GT takes two arguments. It returns T if both arguments are numbers and the first argument is strictly greater than the second. In *all* other cases GT returns NIL.

8. Define a LISP function SAME-SIGN with the following properties. SAME-SIGN takes two arguments. It returns T if both arguments are equal to zero, if both are negative numbers, or if both are positive numbers. In *all* other cases SAME-SIGN returns NIL.

Examples: (SAME-SIGN 0 -2) => NIL (SAME-SIGN -3 -9) => T (SAME-SIGN 'A 'A) => NIL

9. Define a LISP function SAFE-DIV with the following properties. SAFE-DIV takes two arguments. If both arguments are numbers and the second is not equal to zero, then it returns the result of dividing the first argument by the second. In *all* other cases it returns NIL.

Examples: (SAFE-DIV 6 4) => 3/2 (SAFE-DIV 6.0 4) => 1.5 (SAFE-DIV 6 0) => NIL

(SAFE-DIV 6 0.0) => NIL (SAFE-DIV '(6) 4) => NIL

How to Submit Your Solutions

To submit, leave a copy of the final version of your file ***your last name in lowercase-3.lsp*** in your home directory on **euclid** ***no later than midnight on the due date***. [Note: If euclid unexpectedly goes down after 6 p.m. on the due date, the deadline will not be extended. Try to submit no later than noon that day, and sooner if possible.]

Let's assume your last name is Winston and that your **euclid** username is **xxxxx_yyyy316**. Then, **if you are working on venus**, you can enter the following command—**including the colon at the end**—on **venus** to copy the file **winston-3.lsp** from your working directory on **venus** to your home directory on **euclid**: `scp winston-3.lsp xxxxx_yyyy316@euclid.cs.qc.edu:`

If you are working on a PC, use an scp or sftp client to upload your file to euclid. For example, assuming the PuTTY programs have been installed on your PC into the `... \putty` folder (which can be done by downloading the file `putty-0.67-installer.msi` from <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html> and running that installer), one way to upload `c:\316lisp\winston-3.lsp` to euclid would be to enter the following command in a **cmd.exe** window* and then enter your euclid password:

`" ... \putty\pscp" c:\316lisp\winston-3.lsp xxxxx_yyyy316@euclid.cs.qc.edu:`
The double quotes in `" ... \putty\pscp"` and the colon at the end are needed! As an example, if the PuTTY programs have been installed into the `c:\program files (x86)\PuTTY` folder then the above command is:

`"c:\program files (x86)\putty\pscp" c:\316lisp\winston-3.lsp xxxxx_yyyy316@euclid.cs.qc.edu:`

*You can open a **cmd** window on your PC as follows:

1. Type **Win-r** (i.e., hold down the Windows key  and type **r**) to open the Run dialog box.
2. Type **cmd** into the "Open:" textbox and press .

IMPORTANT: **Try doing such a file transfer at least one week before the submission deadline**, so that if you have problems then you will be able to see me during my office hours well before the submission deadline. **Problems with transferring files to euclid will *not* be considered as legitimate reasons for late submission!** Indeed, it is partly because students occasionally have such problems that you are allowed as many as 3 late submissions this semester without penalty.

After leaving the file ***your last name in lowercase-3.lsp*** on euclid as explained above, you should login to your euclid account and **test your Lisp functions on euclid**: Start Clisp by entering **cl** at the `euclid.cs.qc.cuny.edu>` prompt, enter

`(load "your last name in lowercase-3")`

at Clisp's `>` prompt, and then call each of your functions with test arguments.

Do NOT open your submitted file in any editor on euclid after the due date, unless you are (re)submitting a corrected version of your solutions as a late submission!

As mentioned on page 3 of the first-day handout, you are required to keep a backup copy of your submitted file on venus, and another copy elsewhere.