CSCI 316: Lisp Assignment 3

To be submitted *no later than* Thursday, Oct. 13*: Submissions after this due date will be accepted as *late* submissions until a late-submission deadline in December that will be announced later. See p. 3 of the 1st-day announcements document for information on late-submission penalties.

Program in a functional style, *without using SETF*. *Don't* use any features of Lisp other than those introduced in lectures and/or the assigned reading. Follow the indentation and spacing rules at:
https://phantom.cs.qc.cuny.edu/kong/316/indentation-and-spacing-guidelines-for-Lisp-Assignments.pdf

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, as stated on page 2 of the first-day announcements document, *each student working with a partner must write up his/her submission independently, 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 Touretzky (say) and your euclid username is xxxxx_yyyy316, then you must use the filename touretzky-3.lsp—and *not* xxxxx_yyyy-3.lsp!]

This file must include *definitions of any helping functions that are used*. At the beginning of the file there must be a comment that shows *your name* and, if you are working with one or two partners, *the name(s) of your partner(s)*. 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 on euclid. (If you have forgotten how LOAD is used in clisp, re-read the "Working with Lisp Files" subsection on pages 4 – 5 of the Assignment 2 document.) If clisp cannot LOAD your file without error on euclid (i.e., if LOAD gives a `Break ...>` prompt on euclid), you can expect to receive *no credit at all* for your submission, *even if the only error is a single missing or extra parenthesis*!

For further instructions regarding submission, see p. 3.

***NOTES**: **1. There will be *no extension* if euclid unexpectedly goes down after 6 p.m. on the due date: Try to submit no later than noon on the due date, and sooner if possible!**

2. If you have difficulty with these problems, you may make an appointment to see me during my office hours on Wednesday, 10/12; email me by 10/11 to make such an appointment. If you decide to see me, be ready to show me the *solutions.lsp* file you created for Assignment 2 during our meeting.

3. Functions that are incorrectly named may receive no credit—e.g., if the function you write for q. 5 is named "MNTH->INTEGER" or "MONTH-INTEGER" instead of "MONTH->INTEGER", then you may get no credit for that function.

4. Unless you are running clisp as a subprocess of emacs, each time you write or modify a function it is good to save the file, re-LOAD it into clisp, and immediately test the new / modified function.

5. Questions about the problems that are e-mailed to me will *not* be answered before the submission deadline.

6. In this document the term *list* should be understood to mean *proper* list.

1. Define a LISP function MIN-2 with the following properties. MIN-2 takes two arguments, A and B. If the arguments A and B are numbers such that A ≤ B, then MIN-2 returns A. If A and B are numbers such that A > B, then MIN-2 returns B. If A or B is not a number, then 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 p. 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 p. 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 p. 72 of Wilensky]** 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 non-null value if the first argument occurs at least twice in the second argument, but 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. Your definition may therefore 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 (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. For example:
   (MONTH->INTEGER 'C) => ERROR (MONTH->INTEGER 7) => ERROR
   (MONTH->INTEGER 'QUOTE) => ERROR (MONTH->INTEGER '(MAY)) => 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 below ***without*** using COND, IF, WHEN, UNLESS or CASE.
Note: WHEN, UNLESS, and CASE have not been covered in this course, and so students are not expected to have any knowledge of those three macros.

7. Define a LISP function GT with the following properties. GT takes two arguments. It returns T if the arguments are numbers and the first argument is greater than the second; otherwise it returns NIL.
   Examples: (GT 0 –1) => T (GT –3 –7) => T (GT 40 40) => NIL (GT 'B 'A) => NIL
   Remember not to use COND, IF, WHEN, UNLESS, or CASE!

Reminder: Do ***not*** use COND, IF, WHEN, UNLESS or CASE when solving the two problems below.

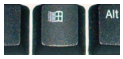
8. Define a LISP function SAME-PARITY with the following properties. SAME-PARITY takes two arguments. It returns T if both arguments are even integers or if both arguments are odd integers. In *all* other cases SAME-PARITY returns NIL.
   Examples: (SAME-PARITY 0 –1) => NIL (SAME-PARITY –3 –9) => T (SAME-PARITY 30 90) => T
   (SAME-PARITY 'A 3) => NIL (SAME-PARITY 4 3.7) => 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 does not satisfy ZEROP, then the function 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 (SAFE-DIV 6 T) => 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, there will be ***no*** extension. Try to submit no later than noon that day, and sooner if possible!]
Be sure to follow all the instructions given on p. 1.

If you are working on venus, a PC, or a Mac, you can use scp or sftp to copy the file to euclid. For example, if your last name is Winston and your **euclid** username is **`xxxxx_yyyy316`**, you can enter the following command—***including the colon at the end***—on **venus**, or in a **powershell** window* on a PC, or in a terminal window on a Mac, to copy **`winston-3.lsp`** from your working directory** to your home directory on **euclid**: **`scp winston-3.lsp xxxxx_yyyy316@euclid.cs.qc.cuny.edu:`**
IMPORTANT: ***Try copying a file to euclid in this way at least 12 days before the submission deadline***, so that if you are unable to copy the file then you can see me well before the deadline. ***Problems with copying files to euclid will not be considered as legitimate reasons for late submission***!
Indeed, it is partly because students occasionally have such problems (e.g., because of forgotten passwords) that you are allowed as many as 3 late submissions this semester without penalty!

*You can open a **powershell** window on a PC as follows: Type **Win-r** (i.e., hold down the Windows key  and type **`r`**), then type **`powershell`** into the "Open:" textbox, and then press  .

If the file you wish to submit is in another directory (e.g., in **`c:/316lisp` on your PC), use the **`cd`** command (e.g., **`cd c:/316lisp`**) to change your working directory to that directory before you enter the **`scp`** command!

After leaving the file *your last name in lowercase*`-3.lsp` on euclid as explained above, login to your euclid account and ***test your Lisp functions on euclid*** as follows: Start Clisp by entering **`cl`** at the `euclid>` prompt, enter **`(load "`*your last name in lowercase*`-3")`** at Clisp's > prompt, and then ***call each of your functions with test arguments***. (**Important**: If the above call of **`load`** produces a **`Break ... >`** prompt, your assignment has ***NOT*** been successfully submitted!)

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! Also do not execute `mv`, `chmod`, or `touch` with your submitted file as an argument after the due date. (However, it is OK to view a submitted file using the `less` file viewer after the due date.)

As stated on p. 3 of the first-day announcements document, you are required to keep a backup copy of your submitted file on venus, and another backup copy elsewhere. One way to do that is to enter the following two commands on **euclid** to email a copy of your submitted file to yourself and to put a copy of the same file on **venus**:

`echo . | mailx -s "copy of submission" -a` *your last name in lowercase*`-3.lsp $USER`
`scp` *your last name in lowercase*`-3.lsp` *your venus username*`@mars.cs.qc.cuny.edu:`

The colon at the end of the second command is needed!