C SCI 316 (Kong): Lisp Assignment 1 [Homework Exercises: Not for Submission]

Try to do these exercises before our 4th class meeting (on Sep. 13). You should definitely have done these exercises by the time of our 5th class meeting (on Sep. 20). You can test the expressions you write on **venus**.

      Before you work on these exercises, logon to **venus** and _enter the following command at_ **venus**'_s shell prompt_ (if you have not already done so): **/home/faculty/ykong/316setup**   After doing that, you should be able to start Clisp on **venus** by entering **cl** at the shell prompt. [If this doesn't work, it may be because your login shell is not **/bin/bash**—in that case you might be able to fix the problem by entering the **chsh** **‑s /bin/bash** command to change your shell, logging out, and then logging back in. **_Otherwise, try entering_** **./cl** or **~/cl** instead of **cl** but also **_let me know_** so I can look into the problem.]

      Alternatively, you can test your expressions on **euclid**: After entering the command **/home/faculty/ykong/316setup** on **venus** as the previous paragraph asked you to do, activate your **euclid** account by following the instructions on page 6 of the 1st-day announcements. Then you can start Clisp on **euclid** by entering **cl** at **euclid**'s shell prompt. (**Note**: You must activate your **euclid** account no later than **Thursday, Sep. 2** _regardless_ of whether or not you plan to use **euclid** for these exercises.)*

I recommend you write solutions to the following problems _on paper_, and then _test your expressions_ by entering them at Clisp's > prompt on venus or euclid:

1. Write a Lisp expression that multiplies 30 by the result of 7 minus 2. To do this, you must write a call of the function * with two arguments; the second argument will be a call of the function – with two arguments.

2. **[Exercise 1 on page 15 of Wilensky]** Write Lisp expressions to evaluate each of the following:
   (a) $3^2 + 4^2$ (b) $3*17 + 4*19$ (c) $12^3 + 1^3 – (9^3 + 10^3)$

3. (a) Write a Lisp expression that divides thirty by the result of seven minus three, in such a way that the result is a rational number (i.e., 15/2).
   (b) Write a Lisp expression that divides thirty by the result of seven minus three, in such a way that the result is a floating-point number (i.e., 7.5). Do **_not_** use the FLOAT function.

4. **[Exercise 3 on page 16 of Wilensky]** Write a Lisp expression that computes the mean of the five numbers eighty-three, eighty-three, eighty-five, ninety-one, and ninety-seven. Do this in such a way that the result is rational, and then in such a way that the result is a floating-point number. Do **_not_** use the FLOAT function.

5. Lisp always prints rational numbers (fractions) in _lowest terms_; for example, if you enter the number `20/12` at Clisp's > prompt (with no spaces before or after the "/") then Clisp will print the equivalent number `5/3`. Write a number that you can enter at Clisp's > prompt to determine whether or not the two integers 7,360,001,711 and 58,879,948,151 are relatively prime. If they are not relatively prime, then find their greatest common divisor. (Two integers are said to be _relatively prime_ or _coprime_ if no integer greater than 1 divides both of them. For example, 20 and 9 are relatively prime; but 10 and 8 are not relatively prime because 2 divides both 10 and 8.)

6. **[Exercise 4 on p. 16 of Wilensky]** The Lisp function SQRT returns the non-negative square root of any non-negative real argument. Use this function to write two Lisp expressions that evaluate to the roots of the equation $x^2 – 11x – 1302 = 0$. In other words, use SQRT to write two Lisp expressions that are respectively equivalent to the following:

   (a) $\dfrac{-(-11) + \sqrt{(-11) \times (-11) - 4 \times 1 \times (-1302)}}{2 \times 1}$ (b) $\dfrac{-(-11) - \sqrt{(-11) \times (-11) - 4 \times 1 \times (-1302)}}{2 \times 1}$

7. **[Exercise 7 on p. 17 of Wilensky]** Write a Lisp expression that can be entered at Clisp's > prompt to determine the order in which Common Lisp evaluates function arguments when it calls a function.   You may assume that the order is either _left-to-right_ or _right-to-left_, and that the same order is used in all Common Lisp function calls.   [**Hint**: Consider a function call (– <expr>$_1$ <expr>$_2$). To evaluate this call with _left-to-right_ argument evaluation order, do the following: First, evaluate <expr>$_1$; next, evaluate <expr>$_2$; finally, subtract the second result (i.e., the value of <expr>$_2$) from the first result (i.e., the value of <expr>$_1$).   To evaluate this call with _right-to-left_ argument evaluation order, do the following: First, evaluate <expr>$_2$; next, evaluate <expr>$_1$; finally, subtract the first result (i.e., the value of <expr>$_2$) from the second result (i.e., the value of <expr>$_1$).   _Normally, both evaluation orders would of course produce the same final value. But if evaluation of one of the two arguments has a side-effect, then evaluation order may make a difference. Try to think of an expression (– <expr>$_1$ <expr>$_2$) for which one of the argument evaluation orders would produce an error but the other would not._]