# CSE341: PROGRAMMING LANGUAGES

**Lisp Warmup Assignment**

**due:**

The purpose of this assignment is to try out Lisp, get used to working with lists and thinking in terms of recursive functions. The assigned functions should be implemented and run on the PCs.

1. Design a **recursive** Lisp function *(greaterlist alist val)* whose parameters are:

   - alist: a list of integers
   - val: a single integer value

   The function should return a new list containing all the integers of alist that are larger than val. Test it on the expression

   (greaterlist '(3 7 4 10 2 19) 5)

2. The above function has the test for greater than built into it. But Lisp allows you to pass the name of a function as a parameter. Design another *recursive* function *(condlist alist op val)* whose parameters are:

   - alist: a list of anything
   - val: a value to be compared with elements of the list
   - op: the name of a predicate to be used in the comparison

   The function should return a new list containing all the elements e of alist for which (op e val) is true. HINT: You can use *apply* to do this.

   Test your function with at least three different predicates: $>$ and $<$ for lists of integers and *notequal* for lists of arbitrary elements, where notequal is defined by

   ```
   (defun notequal (l1 l2) (not (equal l1 l2)))
   ```

   Test it on the following:

   (a) (condlist '(1 3 5 8 0 -18) #'> 2)
   (b) (condlist '(1 3 5 8 0 -18) #'< 2)
   (c) (condlist '((a b) 37 'fun (a b) -18 (a c) 'hi) #'notequal '(a b))

3. Design and implement a Lisp function called *interleave* that takes in two lists L1 and L2 and returns a list with the first element of L1 followed by the first element of L2, followed by the second element of L1, followed by the second element of L2, etc. (ie. the interleave of L1 and L2). Note that there is a nice solution using *apply*, *mapcar*, and a *lambda* function. We will give extra credit for this kind of elegant solution.