Assignment 5 Problem Set

Note: When you turn in an assignment to be graded in this class, you are making the claim that you neither gave nor received assistance on the work you turned in (except, of course, assistance from the instructor or teaching assistants).

Define the following Common Lisp functions and macros. Please save your common lisp code in a file named assignment5.lisp and upload the file to Blackboard when the assignment is complete. A few notes about the assignment:

- assignment5.lisp is expected to contain only the methods described in this assignment manual. (load "assignment5.lisp") should return T and print no other information to the console.
- Parameter names can be arbitrary. I gave the parameters names to identify them in the assignment manual but your implementation does not need to use those specific names.
    - Function and macro names, on the other hand, must be exactly as specified.
- Parameters **must** be in the order they are listed in the assignment manual.
- You may use as many optional parameters as you like, even if no optional parameters are described in the assignment manual.
- Functions are expected to return values, not print values to the console.
    - Functions should not print **any** information to the console when executed.
- Do not define any global variables with defvar. For local variables inside functions, use let and let* blocks.
- Assume all inputs are valid, you do not need to do any parameter validation in your functions.
- You are allowed (and encouraged) to utilize methods defined in Common Lisp which have not been discussed in class. The documentation for Common Lisp can be found at http://clhs.lisp.se/ (I advise using Google to search the site. Although the information is good, the web design aspect leaves much to be desired)
- You **must** comment your Common Lisp code. The ; character is used to denote comments.
- You **cannot** use the setq special operator, nor the setf, delete, or any derivatives of the setq special operator in any of your functions. These are destructive operations and go against the functional paradigm described in class.
- I do not have a preference on what environment you develop your Common Lisp code on. But I will be testing it on SBCL version 1.0.38, so please make sure it runs correctly on SBCL v 1.0.38 before submitting to Blackboard.

1. [5 points] Write a Common Lisp function named myList which creates the following list and returns it

```
(4 (7 22) "art" ("math" (8) 99) 100)
```

2. [10 points] Write a Common Lisp function named leapYear which takes no parameters and returns an ordered list containing all leap years from 1800 though 2018. The list of leapyears **must** be calculated, no points will be given for a hard-coded list.

3. [10 points] Write a Common Lisp function named union- which takes two list parameters. union- returns a single list which contains the separate unique entities from both lists, with no duplication. You are not allowed to use the predefined union function for this function.

4. [10 points] Write a Common Lisp function named avg with a single parameter named aList. avg returns the average of all elements in aList. Assume all elements in aList are numbers. If given an empty list, avg should return NIL. The avg function **must** be tail recursive.

5. [15 points] Write a Common Lisp function named isType which takes a single parameter named dataType. isType will return an anonymous function which takes a single parameter and returns true if the parameters data type is the data type specified in dataType. Otherwise the anonymous function returns false.

6. [15 points] Write a Common Lisp function named taxCalculator with three parameters: limit, rate, and values. limit and rate will be numbers, values will be a list. taxCalculator returns a list with the same elements and ordering of the values parameter EXCEPT every element which is greater than limit is multiplied by rate. Assume that all elements of the values list are numbers.
   BONUS: Make taxCalculator tail recursive +5 points

7. [20 points] Write a Common Lisp function named clean which takes two parameters: aFunc and aList. aFunc will be a function and aList a list. aFunc is expected to be a function which takes a single parameter and returns a boolean value. clean will return a list which contains all values in aList which, when passed to aFunc, return true. if aList contains sublists, clean should create a new sublist with only the values which return true when passed to aFunc

8. [15 points] Define a Common Lisp macro named threeWayBranch which takes three parameters, x y and toExecute. x and y will be numbers and toExecute a list with three sublists. The threeWayBranch macro will execute statements in toExecute's first sublist if x < y, the second sublist if x > y, and the third sublist if x = y. Assume each of toExecute's sublists contain an arbitrary number of statements.