CSC 351L – Comparative Programming Languages Lab
Project 6 – Introduction to Lisp
Due: April 8, 2021 at 11:59 PM

Purpose:

The purpose of this project is to introduce you to Lisp and the Lisp system we will be using, as well as getting you to write simple functions.

Description:

You will write some simple functions and show that they work properly. Your functions will be typed into a file and that file will then be loaded into the Lisp interpreter.

You should write all of these functions from scratch. In other words, you should not use any built-in functions such as `sort` or `reverse`. Additionally, your answers should be completely recursive and functional in nature. There should be no variables (i.e., `let` blocks) or `setf` function calls used.

Steps:

1) Define a Lisp function called `average3` that will find the average of any three numbers. For example, `(average3 2 4 9)` would return `5`.
2) Define a recursive Lisp function called `sum-list` that will add up all of the numbers in a list. For example `(sum-list '(1 4 6 3 9))` would return `23`.
3) Define a Lisp function called `average-list` that will find the average of any list of numbers of any length. For example, `(average-list '(2 7 4 9 8))` would return `6`. You can use `sum-list` and the built-in `length` function in your answer. Make sure you handle empty lists by returning false.
4) Define a Lisp function called `reverse-rest` that will reverse all but the first item on a list. You should do this with a recursive helper function and should not use the built-in `reverse` function. For example, `(reverse-rest '(a b c d e f))` would return `(a f e d c b)`.
5) Define a recursive Lisp function called `insert-it` that will take two parameters (an integer and a sorted list of integers) and will return a list with the new integer inserted into the correct place. Do not use the built-in `sort` function in your answer. For example, `(insert-it 5 '(1 2 6 7 9))` should return `(1 2 5 6 7 9)`.
6) Define a recursive Lisp function called `insertion-sort` that will take a list of integers and return that list in increasing order. For example, `(insertion-sort '(3 7 9 2 8 5 1 4 6))` would return `(1 2 3 4 5 6 7 8 9)`. You can use `insert-it` to write `insertion-sort`.

Deliverables:

When you complete the program, you will prepare two things: a project report and a zip file containing all of your code.

The project report must use the format given in the sample file on D2L. This report can be prepared as a text file, MS Word document, or PDF. The project report should not be included in the zip file.

The zip file should include all lisp files. You can create the zip file in one of two ways – using zip on brahe or on your own computer.

The two items must be uploaded to the Project 6 drop box on D2L. No other form of submission will be accepted.