Lab 06 Specification – Exploring LISP - A Functional Programming Language
Due (via your git repo) no later than 8 a.m., Tuesday, 13th November 2018.
50 points

Lab Goals

The goal for this Lab, is to explore LISP programming techniques and to take a look at some fundamental concepts in Functional Programming.

LISP An Intro

LISP is an oldest language after Fortran. One of the pioneers of Functional Programming Languages, LISP is best used to process any dataset that can be structured as a list. Our goal is not to become a master in LISP programming. But instead, our goal is to use LISP as a base, to understand some fundamental concepts behind Functional Programming. In the last few classes, we had discussed some fundamental concepts in Functional Programming and LISP such as Variable Scoping [Local Variables and Global Variables], Conditional Logic, Recursion, LIST processing techniques such as "CAR" and "CDR". By using the preliminary learning that we had done in our class so far, let us try to do more LISP coding in this lab assignment.

Where Do I Start?

Go to the online tutorial "LISP tutorial" at: `https://www.tutorialspoint.com/lisp/` and just start reading. Skip the parts you think you don't need, but try the examples.

If you are one of those, who like to learn by watching a video tutorial, watch this YouTube video at:
`https://www.youtube.com/watch?v=ymSq4wHrqyU&t=2461s`

The video itself can be considered long [1hr, 15 minutes]. Skip the parts you think you don't need, but try some of the examples mentioned in the video.

There are many other online tutorials available in the web. Feel free to explore other web resources. I recommend the above mentioned tutorials, as I find those as an interesting sources to learn LISP programming. But, feel free to look at other resources as well based on what you find.

I recommend you to also review the textbook chapter 10, lecture slides, lecture notes, code in the coding folder, and in-class activities to get more insight on LISP programming.

Assignment

- Part 1: By using a reflection document in PDF format, answer the following questions:
  1. How is functional programming different from imperative programming? You may refer to online sources to best answer this question. Additionally, I recommend presenting your solution by using a table that list all the differences that you are able to find.
  2. Explain one fundamental concept in Functional Programming and provide an implementation example using LISP programming language? You may provide a simple example, just to showcase your conceptual understanding.
  3. Present one example of Recursion by providing an implementation example in both Java and LISP? Compare both Java and LISP by clearly stating the difference between both approach in term of implementation details. We had already discussed Fibonacci example in class. So, present an example that is different from Fibonacci sequence.

4. Explain the difference between "CAR" and "CDR" by providing an example? Is it possible to use the combination of "CAR" and "CDR"? If your answer is yes, then state clearly if there is an actual limit (in terms of number of times) to utilize such a combination in LISP.
5. What is one thing you like about LISP programming?
6. What is one thing you dislike about LISP programming?

- Part 2: By using lisp code, answer the following questions:
  1. Write a function in LISP called "EvenorOdd" inside a lisp program called EvenorOdd.lisp, that would take an input list and find if the total number of items in the list, is an even number or odd number. You are not supposed to use built-in length function, instead you may use an iterator to count the number of items in list. You may use similar approach, as we had discussed in class examples. After, finding the length of the list, you may need to use conditional operators and mathematical operators to print in console, if the length is even or odd. Inside your LISP code, you are required to call "EvenorOdd" function by passing a list with multiple elements. Test your code, to make sure the code correctly prints the even or odd statement as indicated in the requirement above.
  2. Write a function in LISP called "palcheck" inside a lisp program called PalCheck.lisp, that would take an input list of characters and find if the elements in the list is a palindrome or not. An example palindrome are ’M’ ’O’ ’M’ and ’D’ ’A’ ’D’. You may use built-in functions to handle this requirement. I am not providing more details on the implementation steps for this part, as I want you to think and figure out the implementation steps by yourself. Your code execution should state clearly if the given list is palindrome or not. You may hardcode the list in your LISP code.

Required Deliverables

Please submit electronic versions of the following deliverables to your GitHub repository by the due date:

1. Properly completed and commented source programs.
2. A report document with your guide in PDF format.

Grading Rubric

1. If you complete [Part 1] completely as per the requirement outlined above, you will receive 30 points. There is 5 points awarded for each part.
2. If you complete [Part 2] completely as per the requirement outlined above, you will receive 20 points. There is 10 points awarded for each part.
3. If you fail to upload the lab solution file to your git repo by the due date, there will be no points awarded for your submission towards this lab assignment. Late submissions will be accepted based on the late submission policy described in the course syllabus. It is the student’s responsibility to communicate to the professor if it is a late submission. If the student had not communicated in advance about the late submission, the lab work shall not be graded as such.
4. Partial credit will be awarded, based on the work demonstrated in the lab submission file.
5. If you needed any clarification on your lab grade, talk to the Professor. The lab grade may be changed if deemed appropriate.

Wish you
all the best