Lambda Applicator Lab Overview

In this document, you will find an **Introduction**, a **Grading Guide**, and **Submission Instructions.**

Introduction: what are we doing?

Welcome to Lambda Lab!

In this lab you're going to program a full programming language.

The good news is that you don't have to design it! The specifications are already drawn up: it's lambda calculus. The further good news is that it's a super simple language, with only three constructs, and three possible actions.

You're going to create a REPL for this language. Remember what a REPL is? REPL stands for "Read-Evaluate-Print Loop". When you first started in Python, and typed a command and had it run right away, that was a REPL. Later, in Scheme, there was the definitions window and the interactions window. The interactions window was also a REPL.

That's pretty much the whole job.

What approaches can I take?

I have provided a "suggested approach" to programming this lab, but you are not required to use it at all. You are absolutely welcome to look at the required inputs and outputs, and come up with your own approach from scratch. Many people have done this in the past, and have had very successful projects! Some years I provided no guides at all. The guides I give you are a helping hand if you are just unsure about how to proceed, nothing more. There is no "one right way" to approach the Lambda Lab.

Introduction: why is this so hard?

This is a culmination of your last three years of computer science. For many of you, it will be one of the most intensive learning experiences of your high school career, and *all* of you will get a *lot* out of it.

Why is it a culmination? Because we are going to revisit and combine nearly everything you've learned over the last three years in order to get this done. It's going to make you (neurologically) unpack all of these ideas that you've learned (or maybe only partly learned), play with them, connect them to a bunch of other ideas, and re-pack them up much more efficiently. Remember when we talked about *constellations* in the beginning of the year?

You're going to come out of this with *much richer, much more efficient, and much more interconnected constellations*.

I have made it possible to do well on this assignment even if not everything is perfect in the end. That's because I know what you are setting out to accomplish, and what it's going to take to get there. I will be by your side the entire time. *You can do this!*

What about ChatGPT, past projects on GitHub, or other forms of skipping the work?

This lab is a unique opportunity for you. I make a promise that if you engage it directly, it will consolidate basically all that you have learned so far, and make you a much more powerful programmer (and computer scientist!) in anything you do for the rest of your life. It is *not easy* to get your brain to reorganize like that. This is the big message: don't miss out on these rewards.

The smaller message is that anything that skips past that work and effort will not only deny you the big neurological reward, if you are found out, it will be considered cheating. Once you have been caught cheating, the starting assumption will be that the entire project involved cheating, and your penalty will be commensurate with how much work I can prove that you have actually done.

Lambda Lab Grading Guide

The Lambda Lab is worth 15 weighted points, like a test and a quiz!

Your score will be as follows:

1. 20% the grade you think you deserved, which you will give me in notes.txt. (I will honor that self-awarded grade as long as you are taking this process seriously). See the Submission Instructions for
2. 10% is based on my quick-read of the code for organization, cleanliness, naming, spacing, etc.
3. The remaining 70% is split up in one of two ways: the **primary** rubric, and the **alternate** rubric.
    a. **Primary** rubric: 70% is based on the results of the testing, as per the **Scoring Sheet**. While there is extra credit, no final score may exceed 100%.
    b. **Alternate** rubric: Score may not exceed a total score of 80% (multiply 80% by 70% to get 56%). You can still get the 30% from the quick-read and the self-score, leading to a total possible score of 86%. The 80% (of the original 70%) is divided as follows:
        i. 20% for the **lexer** working
        ii. 20% for the **parser** working
        iii. 20% for **applications** (run) working (10% if applications run, but without alpha substitution)
        iv. 20% for the **dictionary** (=) working

Submission Instructions:

1. Everyone must actually submit the .java files! I don't want .class files, I don't want a .jar… I want *your java files*. Zip them up together and submit them.
2. Every one of the java files should have a comment ***at the top*** with your name (and your partner's name if you have one)
3. Every person must create a document called "notes.txt" and submit it. Every person needs to submit their own notes.txt, but only one person needs to submit the actual project.
4. Inside of notes.txt, write answers to the following:
    a. Your (and your partner's) name
    b. On a scale of 1-10, where 10 is extremely hard and 1 is extremely easy, how difficult was this lab for you?
    c. Estimate how many hours you worked on the lab.
    d. Estimate how many hours your partner worked on the lab. (If you have one.)
    e. Given what you know about (1) your output, (2) the circumstances you were under in your life during this period, (3) how hard you worked for this lab, and (4) how much your partner worked on this lab, ***what grade would you award to yourself?***
    f. Is there anything special I should know while testing and reading your code? (e.g. "such and such crashes my code, but if you skip past these tests, you can resume here and it should work fine." or "such-and-such is not as organized as I would like, but I am very proud of my code on this other group of functions over here!")