Lab 4

Due Sunday, October 8 by 10:00pm

Handout 11
CSCI 334: Fall 2023

Turn-In Instructions

Each question in this assignment must be written using LaTeX. I provide a LaTeX template in your repository for you to use to get started.

For full credit, you must submit both your `.tex` source file as well as the rendered `.pdf` file. Your source file should be called `lab-4.tex` and your PDF should be called `lab-4.pdf`. (5 points)

Note that your submission must be completed entirely using LaTeX. To draw trees, please use the `forest` package. The supplied LaTeX template includes an example for you to start with and modify.

Turn in your work using the Gitlab repository assigned to you. The name of the Github repository will have the form `https://evolene.cs.williams.edu/cs334-f22/<YOUR_USERNAME>/lab04.git`. For example, if your CS username is `22abc1`, the repository would be `https://evolene.cs.williams.edu/cs334-f22/22abc1/lab04.git`.

Honor Code

This is a partner lab. You may work with another classmate if you wish, and you may co-develop solutions. Remember: although you can work on code together, you must each independently write up and submit your solution. No code copying is allowed. **Tell me who your partner is** by committing a `collaborators.txt` file to your repository (5 points). Be sure to commit this file regardless of whether you work with a partner; if you worked by yourself, `collaborators.txt` should contain something like "I worked by myself."

This assignment is due on Sunday, October 8 by 10:00pm.

Sanity Check: Students sometimes submit incomplete assignments, accidentally forgetting to run `git add` for all of their files. Fortunately, there is an easy way to make sure that this does not happen to you. Before you are done, `git clone` your repository to a new folder and then try building/running everything. It only takes a couple minutes and can spare you from headaches later on.

Reading

1. **(Required)** "Introduction to the Lambda Calculus, Part 2"

Problems

Q1. (30 points) ..... Lambda Calculus Reduction

(a) Reduce the following lambda expression.

$$(\lambda x.\lambda y.xy)(\lambda x.xy)$$

Begin by performing alpha reduction. Do all possible reductions to find the normal form. Your reduction should be in the two-column format shown in the course packet.

(b) How do you know that your final expression is a normal form?

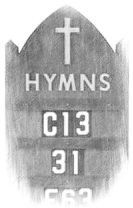
(c) What goes wrong if you do not rename bound variables? Perform a second reduction (again in two-column format) that shows a mistake that can happen when you fail to perform alpha reduction.

Q2. (30 points) ..... Lambda Calculus Reduction

Reduce the following lambda expression.

$$(\lambda x.x)(\lambda x.y)((\lambda x.x\ x)(\lambda x.x\ x))$$

Q3. (20 points) ..... Church Numerals



Church encoding is a means of representing data and operators purely in the lambda calculus. The data and operators form a mathematical structure which is embedded in the lambda calculus. The Church numerals are a representation of the natural numbers using lambda notation. The method is named for Alonzo Church, who first encoded data in the lambda calculus this way.

The natural numbers are written using Church numerals as follows.

Number	Lambda Expression
0	$\lambda f.\lambda x.x$
1	$\lambda f.\lambda x.fx$
2	$\lambda f.\lambda x.f(fx)$
3	$\lambda f.\lambda x.f(f(fx))$
...	...
n	$\lambda f.\lambda x.f^n x$

Subtraction by one can be achieved using the `pred` function.

$$\texttt{pred} \equiv \lambda n.\lambda f.\lambda x.n(\lambda g.\lambda h.h(gf))(\lambda u.x)(\lambda u.u)$$

Prove that 1 - 1 = 0 by performing a reduction using Church numerals and the above `pred` definition.

Q4. (10 points) ..... Translation into Lambda Calculus

A programmer is having difficulty debugging the following Python program. In theory, on an "ideal" machine with infinite memory, this program would run forever. In practice, this program crashes because it runs out of memory, since extra space is required every time a function call is made.

```
def f(g):
    g(g)

f(f)
```

Prove that the program does not terminate by translating the definition of `f` into lambda calculus and then reducing the application `f(f)`.

If you're having trouble getting started, try rewriting the above program using Python's `lambda` feature. Note that an equivalent program in a statically typed language like Java or ML would not compile.

Q5. ($\frac{1}{10}^{\text{th}}$ bonus point) ..... Optional: Feedback

I always appreciate hearing back about how easy or difficult an assignment is.

For $\frac{1}{10}^{\text{th}}$ of a bonus to your final grade, please fill out the following Google Form.