Homework 8

Due Sunday, November 6 by 10:00pm

Handout 20
CSCI 334: Fall 2022

Turn-In Instructions

This assignment must be completed using LaTeX. I provide a LaTeX template in your repository for you to use to get started.

The template I provide should compile successfully without modification. Please note that for full credit, you must submit both your `.tex` source file as well as the rendered `.pdf` file. (5 points) Your source file should be called `project-brainstorm.tex` and your PDF should be called `project-brainstorm.pdf`. (5 points)

Your submission must be completed entirely using LaTeX.

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>/lab08.git`. For example, if your CS username is `22abc1`, the repository would be `https://evolene.cs.williams.edu/cs334-f22/22abc1/lab08.git`.

Honor Code

This is a solo lab. You may study with another classmate if you wish, however you are not permitted to share solutions. Your work must be exclusively your own. Please refer to the section "single author programming assignments" in the honor code handout for additional information. You are always welcome to ask me for clarification.

This assignment is due on Sunday, November 6 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)** Read "How to Fix a Motorcycle" from the course packet.

Problems

Q1. (40 points) ..... Project Brainstorming

For your final project, you will design and implement a programming language. For this part of the assignment, you should think of three different possible final projects you might explore. Two of those proposals should involve artworks at WCMA's Object Lab. The third can be any project that you want. Feel free to let your imagination run wild.

If you find the notion of creating your own programming language daunting, don't worry. For future project checkpoints, you are welcome to fall back to one of the "default" Object Lab projects whose scope will be a little less open-ended. For now, throw caution to the wind with the assurance that at this stage any idea you dream up is non-binding.

Note that a programming language need not be a so-called general purpose programming language. A general purpose programming language is equivalent in expressive power to the lambda calculus or a Turing machine, meaning that is can be used to compute anything. Instead, I encourage you to explore domain specific programming languages, or DSLs. DSLs are usually tailored toward solving a specific problem.

You have probably used a DSL without even realizing it. Some example DSLs are:

(a) `make`
(b) GraphViz
(c) Hypertext Markup Language (HTML)
(d) Extensible Markup Language (XML)
(e) Structured Query Language (SQL)
(f) Markdown
(g) LaTeX
(h) Scalable Vector Graphics (SVG)
(i) JavaScript Object Notation (JSON)

If you have a personal itch, scratch it. For example, I often have to grade student work, and the part I always mess up involves computing scores. Therefore, the grading rubrics I supply to my teaching assistants are actually written in a domain specific language I designed called `tabulator`.

Be creative! Do you love music, or art, or literature? Can you make a language that generates music? Could you make a language that creates art? Could you make a language that generates poetry? Some of your former classmates have designed languages that have done precisely these things and more.

For each potential language, describe

(a) The purpose of the language. What problem does it solve? When I have trouble focusing on this part, it often helps me to think in terms of *inputs* and *outputs*.
    i. *Programs*, written in the language you design, are the inputs.
    ii. The output is what happens *when one of those programs is evaluated*.
(b) What a sample program in the language might look like. Feel free not to be constrained by reality at this point. Just imagine that your programming language already exists. Write two sample programs for your language.

For the two Object Lab pieces, you can propose anything computational involving that artwork. However, as a "default," consider the following purpose: your language should generate artwork in the style of the artist. One way to do this would be to either make your program non-deterministic (it "interprets" the program differently every time), or perhaps it takes an input that tells the interpreter how to proceed with variations deterministically (e.g., by specifying a random seed).

For example, for the Josef Albers painting, "Homage to the Square: Warming," one kind of "fantasy program" might be:

```
5 squares, from gray to orange, stacked and 75% smaller, variation 1.
```

With small tweaks to the same syntax, we can also get very different outputs.

```
8 squares, from gray to blue, side-by-side and equal size, variation 2.
```

What's the grammar? Does it take any additional inputs? What are the outputs? All good questions! This means it's probably an interesting language. At this stage, we don't have to say how the above programs would be interpreted. At a future stage, we will have to say precisely how they are interpreted, so although you don't have to write anything down now, you might spend a few minutes thinking about it.

Be sure to submit your project brainstorm in a file called "`project.tex`" along with a pre-built "`project.pdf`" in a folder called "`project`".