Specs for 201 - NewLangWhoDis

Project Roadmap

Welcome to the first ever EECS 201 Project! This is one of the four options that we'll have available to choose from, and it deals with concepts that might not be practical, but are super interesting and useful: Functional Languages / Rust!

What we're going to ask for this project is rather simple, but it will require a lot of work to accomplish, requiring you to learn a lot about languages and learn a lot of new things about programming in general. This will not be for the faint of heart!

To complete the project, here's the general roadmap of what you need to do:

1. Read the Project Introduction
    - This project will take some time, as we do not want to assume you have web experience and want you to learn more tools.
    - Be sure to read the entire project completely before starting - certain requirements might require you to do things in a certain order, or they might require you to organize your project in a certain way.
    - Feel free to talk to us! We're here to see you succeed, and OH has been relatively quiet these past few weeks...
2. Implement each section entitled "sub-assignment".
    - Each of these sections will give you resources, along with a general idea as to what you need to implement for the project.
    - You're allowed to use as many libraries as you want or like, unless we state that you NEED to use X library or Y library. This is mainly as, for some projects, we want you to learn something new, and we know that certain libraries start out a lot easier but end up being less versatile than others.
3. After implementing each sub-section, join us in OH for a chat!
    - This is mainly so we know who is working on these and how far they are.
    - Additionally, this lets us make sure that everyone is accountable for what they said they'd do, ensuring we can give a grade that is fair and reasonable to

anyone who embarks on these tasks.

- Completing any sub-section of this project will be worth 10 points, so completing all of them WILL get you all the points you need for your assignment. However, each section builds on the previous sections, so to get the full points we suggest starting from the top and moving down.

Sub-Assignment 1: Getting into _____

Here, the goal is to get you started with some programming language that is lesser-utilized in industry or lesser-documented in general. It must be one of the following:

- Haskell
- Lisp
- OCaml
- Rust

If there is some other language you'd like to look at - feel free to ask us at OH! We'll see if it's different enough from C++/Python/JavaScript for it to be your project, but if it is, feel free to do it!

Once you've decided a language, all we're going to ask you is to write up a simple test program, just so you understand the syntax and know what you're doing. Namely:

- Implement a program that solves the following problem:

> Problem 30 - Project Euler
>
> Surprisingly there are only three numbers that can be written as the sum of fourth powers of their digits: As 1 = 1 4 is not a sum it is not included. The sum of these numbers is 1634 + 8208 + 9474 = 19316.
>
> https://projecteuler.net/problem=30

When you're working on this, it might be useful to look at best practices of design, so you're doing things in a manner that uses the language to it's best usage.

Sub-Assignment 2,3,and 4: 0bby

Here, the spec itself is rather short, as we simply want you to re-implement 0hh1, a project from EECS183, in the language you worked on above!

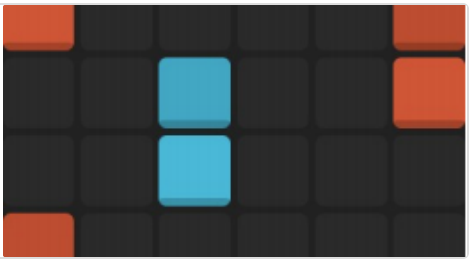
In particular, all we care about is the driver program, as we want to have you working with input and output in a non-traditional programming language and as we want you to show us that you can solve something in a manner you may or may not be used to yet.

Here's the spec:

EECS 183 Project 3: 0hh1

In this project, you will develop a command-line application to read, check, solve, and play basic instances of 0h h1, a Sudoku-like puzzle game.

https://web.archive.org/web/20201028133402/https://eecs183.github.io/p3-0hh1/



As you're working through this, we'd suggest the following:

- Look up idioms in your language of choice! Looking through learning.oreilly.com for what your language can or cannot do is a VERY important part of developing your understanding as to how the language can solve tasks, so we want you to really try to get out of thinking like you're programming with C++!
- Create tests! While we do not ask you for test code, writing tests when you're programming in an unfamiliar language can help debug some of the more mysterious errors you'll face, and give you a better insight as to why an error might exist and why you're internal model of what code does is wrong.
- Google error messages! It might seem simple, but learning what an error message actually means can help you understand what your program is doing and what the compiler/interpreter is doing, allowing you to learn a lot more than what SO says the error is.

When we review this with you, we're looking for the following:

- Does your program perform 0hh1?
- Do you use the unique features of the language well? We know that OCaml, for example, can look a lot like C++ if you write it a certain way, but we're looking for you to stick with the functional side of it as that's why people use it.