Functional Programming: Haskell and Clojure

An independent study by B Corfman

Self-Evaluation

This independent study was very ambitious – I realized shortly into the semester that even after trimming it down to its current state, it was still a lot of reading and a lot of new concepts to learn.

I made it through nine chapters (about 200 pages) of "Learn You a Haskell for Great Good," and experimented and practiced whenever I had the chance – Haskell was an incredibly fun language for me. I also solved some Project Euler problems using Haskell, and a number of problems from "99 Haskell Problems"

For Clojure, I read the entirety of "Clojure from the Ground Up," doing the practice problems and problems on 4clojure as I went along. I had a very short timeframe to do so, but I feel confident in saying that I accomplished a respectable amount of work, and learned a good deal about Clojure.

Finally, I put together a single final project, in Clojure – using Nightmod, a REPL designed for game making, I implemented the game Flood It. I didn't manage to finish it, but ran into, overcame, and learned from a large number of difficulties in implementation. The game is actually fairly close to complete, and I intend to finish it at some point soon as a continued exercise.

I didn't accomplish everything I wanted to, but I have definitely become a more flexible programmer, and gotten a good feel for functional programming languages. For other classes this semester, I've had to write code using Python, and I saw a qualitative difference in how my code was structured and organized. Despite quite a lot of stuttering, pauses, and setbacks, I'm fairly happy with what I learned in the context of this independent study.

Running Instructions

Haskell

Download GHC here: https://www.haskell.org/ghc/distribution_packages

Put euler.hs and H-99.hs in: \Haskell Platform\2014.2.0.0\bin

run ghci (ghci.exe ?)

Project Euler problems

Type :l euler into GHCi

Sum of even Fibonacci numbers below four million:

fibEvenSum

Longest Collatz chain starting at < 1,000,000:

  cGreatestLength 1000000

    This takes a little bit. You can use a lower number if you want

Number letter counts:

  numWordCount x

    Where x is the top of the range [1..x]

    This function counts the letters in all of these numbers (100 = "onehundred", 99 = "ninetynine", 101 = "onehundredandone"

    It can't go above 999

99 Haskell Problems

Type :l H-99 into GHCi

myLast [1, 2, 3]

myLast [7, 6, 5, 4]

myButLast [1, 2, 3]

myButLast [7, 6, 5, 4]

elementAt [1, 2, 3] 0

elementAt [7, 6, 5, 4] 2

compress [1, 1, 2, 2, 2, 2, 3, 5, 5, 7, 7, 2]

pack [1, 2, 2, 2, 2, 5, 5, 5, 2, 2, 3]

encode [1, 2, 2, 2, 2, 5, 5, 5, 2, 2, 3]

Clojure

Clojure.txt is typed up so that you can simply copy/paste it into a Clojure REPL and it should all work.

Example test inputs for all problems in Clojure.txt

```
(ispal "palindrome")                    ;is string a palindrome?
(ispal "racecar")
(ispal "test")
(ispal "sitonapotatopanotis")
(ccount "abcabcabccc")                  ;Number of c's in string
(ccount "cool cucumber")
(myfilter #(= % 5) [1 5 2 5 3 5])       ;Does the same thing as filter
(myfilter #(> % 10) (range 20))
(myfilter even? (range 20))
(primes-below 50)                       ;Number of primes below specified number
(primes-below 102)
(schedule 5)                            ;Just a case lookup function
(schedule 13)
(schedule 19)
(palsbelow 20)                          ;Number of palindromes below specified number
(palsbelow 100)
(palsbelow 1000000)
(id = 1 1 1 1)                          ;Calls the specified function with the given args
(id + 7 8 9 10)
(id map * (range 5) (range 5))
```

```
(exact (* 2452.45 100))                          ;Forces use of ratios

(exact (* 7.3748 20000))

(def x (promise))                                ;Pass sum out of thread using a promise

(.start (Thread. (fn [] (deliver x (sum 0 1e7)))))

(deref x )

(splitsum)                                       ;sum 0-1e7, using parallel processing via two futures
```

Flood it:

This is kind of cobbled together and unfinished so I haven't figured out how to export it yet.

If you download nightmod, you can just put it in the Nightmod folder (I'm not sure where that would be on a mac? On my computer it's in \Users\B)

Then run nightmod, select flood it at the top, and it should load up.

Right now, the following things work:

- If you click on any of the five @'s along the top, the cursor color will change to its color
  - (not actually, but it sets the painting color, basically)
- If you click on any @ in the grid, if the symbol is a different color than the cursor color, it will change to the cursor color

Not currently implemented:

- Cascading the color-change to adjacent @s of the same color of the @ clicked
  - There's some code at the top that was a beginning at trying to attempt this