CS 360

Winter 2018

Programming Language Concepts

CS 360-001 Tuesday/Thursday 15:30-16:50 (UCross 151)

CS 360-002 Tuesday/Thursday 14:00-15:20 (UCross 151)

CS 360-003 Tuesday 18:30-21:20 (UCross 153)

Instructor: Geoffrey Mainland
mainland@drexel.edu (mailto:mainland@drexel.edu)
Office: University Crossings 106
Office hours: Mondays 4pm–7pm; Thursdays 5pm–6pm.

Teaching Assistant: Xiao Han
CLC office hours: Wednesday 2pm–4pm; Thursday 12pm–2pm

Lab 1: More Scheme

Due Monday, January 21, 11:59:59PM EST.

Accept this assignment on GitHub Classroom here (https://classroom.github.com/a/-QXCOGLY). The standard homework instructions (..) apply.

You may work with 1 partner on this assignment. You **must both acknowledge your partner** in your README.md file. You should both turn in the assignment by pushing to GitHub. Is is perfectly acceptable to turn in the same or similar code.

This assignment asks you to write several Scheme functions that make use of state and streams. Implement, document (i.e., write a specification for), and test the following functions in scheme. Scheme includes some of the constructs below, but you **may not** use these implementations. Instead, you must implement them yourself. Functions that return a stream may not use lists internally, e.g., they may not construct a list and then convert the list to a stream as the final step. We **will** test your code with infinite streams. Make sure all functions are thoroughly tested.

You must implement the functions as specified. You may write other helper functions and define test data in your file, but you **may not** change the functions' names or the number or order of arguments. You **must** also write a short description of what the function does as we did for the examples in class.

We expect you to use proper Scheme style. We use proper Scheme style in class. You may also refer to this Scheme style guide (http://community.schemewiki.org/?scheme-style).

You should complete the homework by modifying the file `lab1.rkt` that we provide as part of your repository.

This assignment is worth 50 points. There are 50+1 possible points.

Problem 1: `(stream-scan f z s)` (10 points)

Modify your `scan` function from Homework 1 so that it takes a stream as an argument and returns a stream as the result— `stream-scan` must not use lists.

Problem 2: `(stream-take-n n s)` (10 points)

Write a function `stream-take-n` that takes two arguments, an integer `n` and a stream `s`, and returns a list consisting of the first `n` items in the stream.

Problem 3: `(stream-pair-with f s)` (10 points)

Write a function `stream-pair-with` that takes a function `f` and a stream `s`, consisting of elements $x_1, x_2, \ldots$, and returns a new stream where each element x_i of `s` has been paired with $f\ x_i$

Examples:

```
(stream->list (stream-pair-with (lambda (x) (+ x 1)) (stream 1 2 3 4))) => ((1 . 2) (2
 . 3) (3 . 4) (4 . 5))
```

Problem 4: `(cycle-lists xs ys)` (10 points)

Write a function `cycle-lists` that takes two lists, `xs` and `ys`, and returns a stream. The lists may or may not be the same length, but you may assume they are both non-empty. The elements produced by the stream are pairs where the first part is from `xs` and the second part is from `ys`. The stream cycles forever through the lists. **Your solution should not require the use of mutable state, i.e., it should not use `set!`.** A constant-time ($O(1)$) solution is possible.

Examples:

```
(stream-take-n 8 (cycle-lists '(1 2 3) '("a" "b"))) => ((1 . "a") (2 . "b") (3 . "a")
(1 . "b") (2 . "a") (3 . "b") (1 . "a") (2 . "b"))
```

Hint: Think about how you could create a stream that endlessly cycles through all the elements of a single, given list. You can use a local environment to store a copy of the list through which you must cycle and a helper function that uses this stored copy when it runs out of elements to cycle through.

Problem 5: `(seen x)` (10 points)

Write a function `seen` that takes a single argument `x` and return `#t` if it has been previously called with an `equal?` argument and `#f` otherwise.

Hint: You may use a global variable if you like, but a solution that does not use global state is possible.

Problem 6: Homework Statistics (1 point)

How long did spend on each problem? Please tell us in your `README.md`. You may enter time using any format described here (https://github.com/wroberts/pytimeparse).