Programming Paradigms

Lab 3. Higher-order functions and lists in Racket

Outline

- Higher-order functions recap
- Exercise: using foldl, map and filter
- Exercise: rendering 2D grid with higher-order functions

Warm-up exercises: length and average

Exercise 3.1.
Implement function `len-via-foldl` that computes the length of the list using `foldl`.

Exercise 3.2.
Implement function `len-via-map` that computes the length of the list using `map`.

Exercise 3.3.
Implement function `average` that computes the average for a list of numbers.

Removing duplicates

Exercise 3.4.
Implement function `my-remove-duplicates` that produces a new list without duplicate values. Use `equal?` predicate to compare values in the list.

Exercise 3.5.
Implement function `my-remove-duplicates-with` that produces a new list without duplicate values and uses a user-provided predicate to check for equality.

Exercise 3.6.
Formulate assumptions (restrictions) about the input predicate used in Exercise 3.5.

Rendering 2D grid

Exercise 3.7.
Implement function `for-range` that produces a list by applying a given function to each number in a given range.

Exercise 3.8.
Implement function `for-range-2D` that produces **a list of lists** by applying a given function to each pair of integer 2D coordinates in a given 2D area.

Exercise 3.9.
Implement function `render-2D` that produces a picture by appending 1×1 pictures generated from a given function at every integer 2D coordinates in a given range.

Extra exercises

Exercise 3.10.
Implement `map` using `foldl`.

Exercise 3.11.
Implement `reverse` using `foldl`.

Exercise 3.12.
Using `render-2D` implement a function that renders a given vector field. Represent a vector field as a function from a point in space to a vector (represented by a pair of numeric values). Render the gradient vector field for F(x, y) = sin x cos y