Homework #4

Due: Tues, Feb 18, 10:59pm

How to Submit

- Submit one solution *per team* (each team can have 1–3 members), through TEACH. Put the names and ONID usernames of all team members as a comment at the top of the file.
- Your submission should consist of a file named `HW4.<your-username>.hs`, where `<your-username>` is the ONID username of the team member who submitted the file, and possibly an HTML file with your amazing picture (see below).
- Your Haskell file *must compile without errors or warnings* in GHCi. Put all non-working parts of your solution in comments! If your file does not compile, the TA will not evaluate it.
- Please preserve the existing `doctest` comments in the template (the lines starting with >>> and the results underneath). This will help the TA during grading.
- If you can't solve a problem, you can get partial credit by describing in comments what you tried and where you got stuck.

Files

Template: HW4.template.hs

Support files (*don't edit or submit these*):

- MiniMiniLogo.hs
- Render.hs

You should download both of these files and put them in the same directory as your `HW4.<your-username>.hs` file.

Description

MiniMiniLogo is a simplified version of the MiniLogo language that you worked with in HW3, which is itself a simplified version of the Logo language for programming simple 2D graphics.

MiniMiniLogo is like MiniLogo, except that it doesn't have macros, variables, or addition. This leaves only a very simple syntax remaining.

int	::=	(any integer)	
prog	::=	ε | *cmd* `;` *prog*	sequence of commands
mode	::=	`down` | `up`	pen status
cmd	::=	`pen` *mode*	change pen mode
	|	`move (` *int* `,` *int* `)`	move pen to a new position

The following MiniLogo program (to draw a 2x2 square with its bottom-left corner at the origin) is also a valid MiniMiniLogo program.

```
pen up; move (0,0);
pen down; move (2,0); move (2,2);
          move (0,2); move (0,0);
```

A Haskell implementation of the abstract syntax of MiniMiniLogo is already provided for you in the file MiniMiniLogo.hs. It also provides some functions to generate programs that draw simple shapes.

In this assignment, you will be implementing the *semantics* of MiniMiniLogo. The meaning of a MiniMiniLogo

program is: (1) the change in the state of the pen and (2) a list of lines to draw.

Conceptually, the execution environment for a MiniMiniLogo program consists of two parts:

- a *canvas* rooted at position `(0,0)` and extending infinitely upward and to the right
- a *pen* which is always located at a certain absolute point on the canvas, and which can be in one of two states, either `up` or `down`

The `move` command moves the position of the pen from one absolute point to another. If the pen is `down` when it moves, it draws a straight line connecting the two points. If the pen is `up`, it just moves to the new point but does not draw a line. The state of the pen can be changed using the `pen` command as illustrated in the example program above.

The support file Render.hs defines types for representing the state of the pen, and provides a library for rendering the output of a MiniMiniLogo program as an SVG image in an HTML5 file.

Tasks

1. Implement `cmd`, the semantic function for MiniMiniLogo commands (`Cmd`). Note that a command updates the state of the pen and possibly draws a line. Therefore, the semantic domain is `State -> (State, Maybe Line)`.

2. Implement `prog`, the semantic function for MiniMiniLogo programs (`Prog`). A program changes the state of the pen and may draw several lines. Therefore, the semantic domain is `State -> (State, [Line])`.

After you have implemented `prog`, you can use the `draw` function in the template to run a MiniMiniLogo program and render its output to HTML, which you can then load in your browser. To see if your semantics is working correctly, you can compare the result of running `draw demo` with this image: MiniMiniLogo-Demo.png

Extra credit

Use your creativity to produce a MiniMiniLogo program that draws an amazing picture! I will show off the most amazing pictures (anonymously) in class.

To get extra credit, you must do two things:

1. Define the `amazing` variable in the template file. This should be the MiniMiniLogo program that generates the amazing picture.

2. Submit the `.html` file produced by running `draw amazing`. To make things easier for me, first *rename this file* to reflect your amazing picture. For example, if your amazing picture is the Mona Lisa, name the file `MonaLisa.html`.

The amount of extra credit awarded will be proportional to the technical impressiveness and/or artistry of the amazing picture.

To make a truly amazing picture, you will probably need to define some helper functions that generate parts of your MiniMiniLogo program, similar to `box`, `nix`, and `steps` in `MiniMiniLogo.hs`. You are free to define as many of these helper functions as you wish.

In the past, some students have also modified the `Render.hs` file to make their pictures even more amazing. If you decide to go down the rabbit hole, please also submit your modified `Render.hs` file so that we can reproduce your amazing picture.

Back to course home page