Report

Assignment 2

Name: Yixi Rao

Lab: Fri15 L124_left 15:00-17:00 Alex Smith

Id: U6826541

Introduction

My program is designed to draw some picture such as polygons and triangles, user just need to press a single key and it will return the picture to you.

Program design

To draw a picture in Haskell, my choice is to use the `polyline` function in the CodeWorld, because I think it is the principle of drawing any picture and it is easy to control and manipulate it though coordinates. In order to demonstrate a picture, I divided the problem into three parts, first I should generate some turtle command, the reason is that it can contain some very useful information such as the point in Cartesian coordinates. Second, the command must be interpreted and translated into picture, to do that, the helper function may be very useful because I consider it can separate a big problem into small one, which I can deal with it step by step. Finally, the essence of the function of sierpinski is also just a command.

The `polygon` functions I decided to use helper function because I considered it can store more inputs that allow the code have more different choices to do some operations on the inputs. Besides I used a guard, because it can make me easier to distinguish different conditions, then I can perform different operation according to it.

I defined two data type. The first one is `TurtleState`, the reason why I defined this type is that it can collect all the information about the point that make the operation more convenient to perform and do the operation that I want, make the function easier to solve the problem as well. The second one is `UqandDown`, I defined this because it can differentiate and separate two different situations, which make the problem more obvious to solve.

In order to interpret the command, I wrote three helper functions as I think when commands are transformed to the picture it should undergo a lot of states, namely, type. The first phase is `comToState`, I wrote this by using case statement, because it can classify all the situation of the variable, then they will all be transformed into TurtleState. Moreover, I added another input of `comToState`, the reason why I did this is that it can act as a role of recording every current state of the Turtle that I can use. The last phase is `stateToPic`, which interprets the state of turtle and turns it into picture. I used the case statement, I think it is a good idea, because it can satisfy the different condition of pen and make the code more readable.

On the `sierpinski` function, I decided to define a new helper function to deal with it, since I realized that a helper function can insert new type into original type declaration, which allow me to do more manipulation of the variable. I decided to use guard to implement the function, the reason is that it can help me easier to finish recursion and make the code more succinct, besides, I used where statement to replace some expression to abbreviation, which let the code shorter and understandable.

Testing

On the task 1 function, I tested it by calling the function, and sketched the graph on the paper according to the result of the function call, then I collated it by running the whole program (cabal run) to have the actual picture, moreover I changed the value of the parameters in Main.hs to make sure it is correct.

On the task 2 function, I tested it by using `drawingOf` function, I used some extreme example such as when the polygon just has 2 sides or 3 sides and what it will return when the turtle command is an empty list. In addition, I tested every helper function individually though writing down what it should be, and calling the function to see whether I am correct or not. Besides, I tested it by pressing "- "and "+" key to see what is going on.

My decision of how to test task 3A function was changing the value of the parameters in Main.hs, and I changed the value of depth from 1 to 10. I checked it by searching the picture on the internet.

Beyond that, I wrote some test on the TurtleTest.hs. About the `polygon` and `triangle`

function, I found some properties of polygon and triangle such as sides of the polygon and the perimeter of polygon or triangle. Therefore, I considered it is a good way to test on this property, because I can decipher the list of commands to extract useful information and calculate it, then I wrote some helper function that can help me to test on these properties. How I tested on task 2 three helper functions is c very similar to how I tested on task 1, just replace the commands to states. But I tested on `StateToPic` in a different way, which is just test on the polyline function, since it is the authentic function that will produce a picture. So, I made use of the `comp1100`, for it contain PenDown and PenUp conditions, which can eliminate contingency.

Reflection

I chose this way to implement my program since the problem of this program can be divided into three parts, each part can tackle a different but related small problem, and propagate the output to the next part, which can be used. In the task 2, there is another way to write the function by using just one helper function, which inserts a transitional type 'turtle state' to the original type declaration. The advantages of this alternative are that it is very succinct, short as well, and it will produce the picture directly. however, the disadvantages of this are it is not easier to understand, and it will contain a tremendous amount of case statement, which will confuse the reader.

I am be of opinion of that I can write the code of task 3 more better, because the code should be more regular, then I can make it shorter and more distinct to the reader and myself And, I can delete some redundant PenUp and PenDown in Task 1, because it can make the code more efficient.