CCPS506 Lab 6 – Haskell: Custom types and type classes

Preamble

In this lab, we will continue exploring types in Haskell by creating our own custom type and making it a member of several common type classes.

Getting Started

Create a Haskell source file called **`Lab6.hs`**. In this file you will define a module named **`Lab6`**. All functions written for this lab will be placed in this single module. Make sure each of your functions is named precisely according to the specifications below.

Lab Description

Create a custom Haskell type for representing regular polygons. Regular polygons are equilateral and equiangular (all sides have the same length, all angles between sides are the same). Common examples are square, pentagon, hexagon, octagon, etc.

This custom type must be called **`RegularPoly`** and will have two fields – one field for the number of sides (`Int`), and another for the length of each side (`Float`). Your **`RegularPoly`** type must support the following functionality:

i) Implement a function called **`polyArea`** that accepts a **`RegularPoly`** as input and returns the area of that polygon. Use Google to find a general formula for computing the area for any regular polygon.

ii) Your **`RegularPoly`** should be an instance of the **`Eq`** type class. Equality between two regular polygons is defined as having the same area. Note that two regular polygons need not have the same number of sides or side length to have the same area!

    As we know at this point in our computer science careers, comparing equality between floating point values is fraught. Thus, your (==) function should check equality to within some small value (i.e. 1e-8). If the areas are within this tiny value of each other, we call them equal ("close enough")

iii) Your **`RegularPoly`** should also be an instance of the **`Show`** type class. Come up with a visually pleasing way of displaying instances of your regular polygon type. How exactly you display this is up to you, but the number of sides and side length should be included somehow.

Submission

Labs are to be completed and submitted *individually*. Submit your **Lab6.hs** file containing the **`Lab6`** module and all completed functions on D2L, under the submission for Lab 6.