Lab: Cryptol Data Structures

Exercise 1:
In a new file `solutions.cry`, declare a type `Circle` of three elements: a `radius` of type `Rational`, a `left` of type `[64]`, and a `right` of type `[64]`. Create an object `aCircle` of type `Circle` with `radius` 5/2, `left` of 20, and `top` of 16. Display `aCircle`. ■

Exercise 2:
In `solutions.cry` write a function called `areaCircle` that takes a `Circle` object `circle` as input and returns the area of `circle`. Area of a circle is π times the radius squared. Use 355/113 as a close approximation to π. Find the area of `aCircle`. ■

Exercise 3:
In `solutions.cry` declare a type `Displacement` containing elements `left` of type `[64]` and `top` of type `[64]`. Create an object `disp` of type `Displacement` with `left`=12, `top`=2. Display `disp`. Create a function called `nudgeCircle` that takes as input a `Circle` object `circle` and a `Displacement` object `d` and outputs `circle` modified so its `left` and `top` are increased by `left` and `top` of `disp`. Let `movedaCircle` be the output of `nudgeCircle` applied to `aCircle` and `disp`. Display `movedCircle`. Create a new `Displacement` object `newDisp` with `top`=2 and `left`= -32. Let `secondTry` be the output of `nudgeCircle` applied to `aCircle` and `newDisp`. Display `secondTry`. What happened? ■

If negative numbers are going to show up they can be accommodated with type `Float` instead of `[64]`. To do this, the module `Float` must be imported. Thus, at the top of `solutions.cry` add this:

```
import Float
```

Now redefine `Circle` like this:

```
type Circle = { radius : Rational, left : Float16, top : Float16 }
```

Also, define Displacement like this:

```
type Displacement = { left : Float16, top : Float16 }
```

The following do not change: aCircle, areaCircle, aCircleArea, nudgeCircle, disp, newDisp. The definition of movedaCircle and secondTry do not change but the value of secondTry does:

```
Main> :l solutions.cry
Loading module Cryptol
Loading module Float
Loading module Main
Main> movedaCircle
{radius = (ratio 5 2), left = 0x20.0, top = 0x12.0}
Main> secondTry
{radius = (ratio 5 2), left = -0xc.0, top = 0x12.0}
```

Exercise 4:

Body Mass Index (BMI) is defined, for imperial units as weight, in pounds, divided by the square of height, in inches, times 703. For metric system units the weight is in kilograms, the height is in centimeters and the 703 is replaced by 10000. In `solutions.cry` define a type called `BMI` whose fields are `weight` and `height` and a type called `BMIimp` whose fields are `weight`, `feet`, and `inches` (example input for the latter is 175 pounds, 5 feet, 10 and ½ inches). Write function calcBMI that takes a BMI object as input and outputs a BMI as described above for metric units and a function calcBMIimp that takes a BMIimp object as input and outputs a BMI as described above for imperial units. Try out a few examples. ■