Contracts

Contracts tell us how to use a function. For example: `num-min :: (a :: Number, b :: Number) -> Number` tells us that the name of the function is `num-min`, it takes two inputs (both Numbers), and it evaluates to a `Number`. From the contract, we know `num-min(4, 6)` will evaluate to a `Number`. **Use the blank line under each contract for notes or sample code for that function!**

Name		Domain		Range
triangle	::	(side-length :: Number, style :: String, color :: String)	->	Image
triangle(80, "solid", "darkgreen")				
circle	::	(radius :: Number, style :: String, color :: String)	->	Image
circle(30, "outline", "fuchsia")				
star	::	(radius :: Number, style :: String, color :: String)	->	Image
star(50, "solid", "teal")				
rectangle	::	(width :: Num, height :: Num, style :: Str, color :: Str)	->	Image
rectangle(20, 80, "solid", "gold")				
ellipse	::	(width :: Num, height :: Num, style :: Str, color :: Str)	->	Image
ellipse(30, 70, "outline", "lightblue")				
square	::	(size-length :: Number, style :: String, color :: String)	->	Image
square(10, "outline", "red")				
text	::	(str :: String, size :: Number, color :: String)	->	Image
text("I'm thankful for…", 50, "green")				
overlay	::	(img1 :: Image, img2 :: Image)	->	Image
overlay(star(30, "solid", "gold"),circle(30, "solid", "blue"))				
beside	::	(img1 :: Image, img2 :: Image)	->	Image
beside(star(50, "solid", "orange"),circle(50, "solid", "green"))				
above	::	(img1 :: Image, img2 :: Image)	->	Image
above(triangle(30, "solid", "red"),square(30, "solid", "blue"))				
put-image	::	(img1 :: Image, x :: Number, y :: Number, img2 :: Image)	->	Image
put-image(star(30, "solid", "red"), 50, 150, rectangle(300, 200, "outline", "black"))				
rotate	::	(degree :: Number, img :: Image)	->	Image
rotate(35, rectangle(30, 80, "solid", "purple"))				
scale	::	(factor :: Number, img :: Image)	->	Image
scale(0.8, triangle(30, "solid", "red"))				