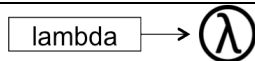
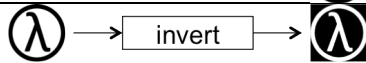
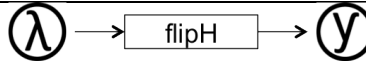
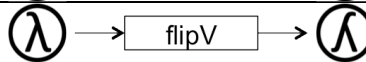
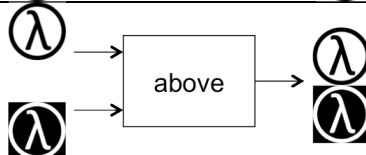
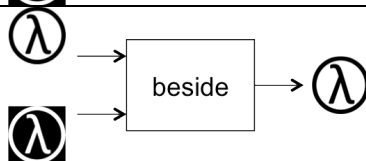


Übung: Funktionale Grafiken

In dieser ersten Übung verwenden Sie vordefinierte Funktionen um Grafiken zu bearbeiten.
Auf dem AD finden Sie unter 01_Intro/Assignment/ folgende Dateien:

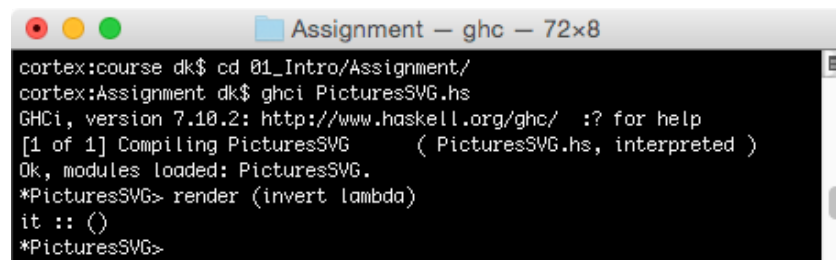
PicturesSVG.hs	Haskell Modul mit vordefinierten Funktionen.
PicturesSVG.html	Eine HTML Seite, die das Resultat der Funktionen anzeigt.
lambda.png	Ein Beispiel Bild mit dem Sie arbeiten können.
result.svg	Resultierende Grafik die dann in PicturesSVG.html angezeigt wird.

Das Haskell Modul PicturesSVG.hs enthält folgende Definitionen:

Signatur	Verhalten
<code>lambda :: Picture</code>	
<code>invert :: Picture -> Picture</code>	
<code>flipH :: Picture -> Picture</code>	
<code>flipV :: Picture -> Picture</code>	
<code>above :: Picture -> Picture -> Picture</code>	
<code>beside :: Picture -> Picture -> Picture</code>	

Folgen Sie der Anleitung um Bilder zu generieren:

- Öffnen Sie mit einem Webbrowser die Datei PicturesSVG.html
- Laden Sie das Modul PicturesSVG.hs in GHCi. Sie sollten nun diesen Prompt sehen:
`*PicturesSVG>`
- Geben Sie nun folgenden Ausdruck ein:
`render (beside lambda (invert lambda))`
- Als Resultat wurde eine neue result.svg Datei geschrieben. Im Browser sollte dieses Bild nun angezeigt werden.¹



```

cortex:course dk$ cd 01_Intro/Assignment/
cortex:Assignment dk$ ghci PicturesSVG.hs
GHCi, version 7.10.2: http://www.haskell.org/ghc/ :? for help
[1 of 1] Compiling PicturesSVG      ( PicturesSVG.hs, interpreted )
Ok, modules loaded: PicturesSVG.
*PicturesSVG> render (invert lambda)
it :: ()
*PicturesSVG>

```

Hinweis:

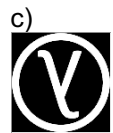
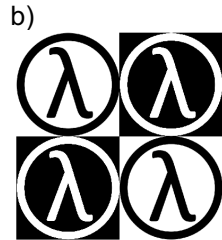
Funktionen die zwei Argumente erwarten, können statt prefix (am Anfang stehend) auch infix (zwischen den Argumenten stehend) geschrieben werden. Dazu muss die Funktion zwischen ` (Backticks) geschrieben werden:

Prefix	Infix
<code>beside lambda (invert lambda)</code>	<code>lambda `beside` (invert lambda)</code>

¹ Falls das nicht geklappt hat, müssen Sie möglicherweise die Seite neu laden.

Aufgabe 1: Vom Bild zum Code

Finden Sie die Ausdrücke um folgende Bilder zu erzeugen:



Aufgabe 2: Rotieren um 180°

Definieren Sie die Funktion `rotate180 :: Picture -> Picture`, die das Bild um 180° rotiert. Verwenden Sie dazu nur die vorgegebenen Funktionen. Sie können die Funktion gleich im File `PicturesSVG.hs` implementieren.

Aufgabe 3: Schrittweise Evaluation von Ausdrücken

Gegeben sind folgende zwei Funktionen:

```
times2 :: Integer -> Integer
times2 x = 2 * x           -- (t2)

square :: Integer -> Integer
square x = x * x           -- (sqr)

pyth :: Integer -> Integer -> Integer
pyth a b = square a + square b -- (py)
```

Geben Sie die Herleitung der folgenden Ausdrücke:

- a) `square (times2 3)`
- b) `pyth 2 2`
- c) `pyth (square 2) 3`

Beispiel:

```
times2 (times2 4)
~> times2 (2 * 4)      -- using (t2)
~> times2 8            -- arithmetic
~> 2 * 8               -- using (t2)
~> 16                  -- arithmetic
```