FUNCTIONAL PROGRAMMING

Haskell

HIGH ORDER FUNCTIONS

A function is called higher order if it takes a function as an argument or returns a function as a result

```
applyTwice :: (a -> a) -> a -> a

applyTwice f x = f (f x)
```

```
ghci> applyTwice (+3) 10
16

ghci> applyTwice (++ " HAHA") "HEY"
"HEY HAHA HAHA,,

ghci> applyTwice ("HAHA " ++) "HEY"
"HAHA HAHA HEY"
```

SECTIONS

- Infix functions can also be partially applied by using ***sections***. To section an infix function, simply surround it with parentheses and supply a parameter on only one side.

```
divideByTen :: (Floating a) => a -> a

divideByTen = (/10)
```

```
isUpperAlphanum :: Char -> Bool

isUpperAlphanum = (`elem` ['A'..'Z'])
```

THE MAP FUNCTION

- The map function takes a function and a list, and applies that function to every element in the list, producing a new list. Here is its definition:

- ghci> map (replicate 3) [3..6]

```
[[3,3,3],[4,4,4],[5,5,5],[6,6,6]]
```

- ghci> map (map (^2)) [[1,2],[3,4,5,6],[7,8]]

```
[[1,4],[9,16,25,36],[49,64]]
```

- ghci> map fst [(1,2),(3,5),(6,3),(2,6),(2,5)]

```
[1,3,6,2,2]
```

THE FILTER FUNCTION

- As another example, let’s find the largest number under 100,000 that’s divisible by 3,829. To do that, we’ll just filter a set of possibilities in which we know the solution lies:

```
largestDivisible :: Integer

largestDivisible = head (filter p [100000,99999..])

 where p x = x `mod` 3829 == 0
```

SCANS

- The **scanl** is the function that except all the intermediate accumulator states in the form of a list.

- ghci> scanl (+) 0 [3,5,2,1]

```
[0,3,8,10,11]
```

- ghci> scanl (flip (:)) [] [3,2,1]

```
[[],[3],[2,3],[1,2,3]]
```

THE COMPOSITION FUNCTION

- For example, say we have a list of numbers and we want to turn them all into negative numbers. One way to do that would be to get each number’s absolute value and then negate it, like so:

```
ghci> map (\x -> negate (abs x)) [5,-3,-6,7,-3,2,-19,24]
[-5,-3,-6,-7,-3,-2,-19,-24]
```

Notice the lambda and how it looks like the result of function composition. Using function composition, we can rewrite that as follows:

```
ghci> map (negate . abs) [5,-3,-6,7,-3,2,-19,24]
[-5,-3,-6,-7,-3,-2,-19,-24]
```

PRACTICES

- Write a program in Haskell **using Filter** where there is a list of years from 1990 until the current year. Find out the first year in the list which one is a leap year.

- Write a program in Haskell **using Scans** where input is a number and the output is the multiplication of that number with 5.