CS 344 – More Higher Order Functions

Note Title 3/23/2012

Announcements

- HW due Monday
- HW 7 is up, due next Friday

How about reversing a list?

Code this one yourself, & think recursively!

Start:

rev :: [a] → [a]

```
rev :: [a] -> [a]
rev [] = []
rev [x] = [x]
rev (x:xs) = rev xs ++ [x]
```

Another example: zipWith

```
zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]

-- base cases
zipWith _ [] _ = []
zipWith _ _ [] = []

zipWith f (x:xs) (y:ys) =
        f x y : zipWith f xs ys
```

Another useful function:

Flip : gives inverse function

```
flip' :: (a -> b -> c) -> (b -> a -> c)
flip' f = g
  where g x y = f y x
```

Other useful functions

Map: takes a function & a list & applies the function to every member of the list

Filter: takes a predicate (function which returns a boolean) & returns new list w/ things that satisfy the predicate

Ex: > filter (>3) [1, 2, 6, 5, 1, 3, 9]

> filter (even) [1..10]

(More on filter - nice application)

```
quicksort :: (Ord a) : [a] -> [a]
quicksort [] = []
quicksort (x:xs) =
    let smaller = quicksort (filter (<=x) xs)
        bigger = quicksort (filter (>x) xs)
    in smaller ++ [x] ++ bigger
```

Ex: find largest # under 100,000 divisible by 3829

```
let largestDiv =
head (filter p [100000, 99999..])
    where p x = x `mod` 3829 == 0
```

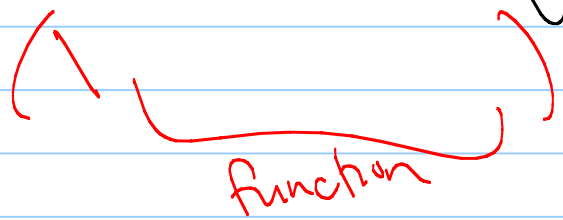
Lambdas : λ

"Anonomous" functions. → \

Look at previous example.

p was only defined to pass along (which is a bit silly).

So a better way: use \ & don't have to give a name!



Another lambda example:

```
zipWith (\a b -> (a*2+5)/b) [1,2,3,4,5] [5,4,3,2,1]
```

The $ function

Can be useful — makes a function low priority

sqrt (9 + 16) ↝ 5

answer?

sqrt $ 9 + 16

Next week

- Modules
- Making our own types