COMP105 Lecture 15

Map

Recap: transforming lists

```
double_list [] = []
double_list (x:xs) = 2 * x : double_list xs

ghci> double_list [1..5]
[2,4,6,8,10]
```

```
square_list [] = []
square_list (x:xs) = x * x : square_list xs

ghci> square_list [1..5]
[1,4,9,16,25]
```

Map

Map applies a function `f` to every element in a list

```
map' :: (a -> b) -> [a] -> [b]
map' _ [] = []
map' f (x:xs) = f x : map' f xs
```

```
ghci> map even [1..5]
[False,True,False,True,False]
```

Map examples

```
square x = x * x
```

```
ghci> map square [1..5]
[1,4,9,16,25]
```

```
ghci> map reverse ["the", "quick", "brown", "fox"]
["eht","kciuq","nworb","xof"]
```

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

Currying and map

It is common to use **curried functions** with map

```
ghci> map (*2) [1..5]
[2,4,6,8,10]
```

```
ghci> map (2^) [1..5]
[2,4,8,16,32]
```

```
ghci> map (drop 2) ["the", "quick", "brown"]
["e","ick","own"]
```

Anonymous functions and map

It is common to use an **anonymous function** with map

```
ghci> map (\x -> x*x) [1..5]
[1,4,9,16,25]
```

```
ghci> map (\(x, y) -> x + y) [(1,1), (2,2), (3,3)]
[2,4,6]
```

```
ghci> map (\(_:y:_) -> y) ["the", "quick", "brown"]
"hur"
```

Nested maps

When working with nested lists, it is common to use **nested maps**

```
ghci> map (map (*2)) [[1,2,3], [4,5,6], [7,8]]
[[2,4,6],[8,10,12],[14,16]]
```

```
import Data.Char

ghci> map (map toUpper) ["the", "quick", "brown"]
["THE","QUICK","BROWN"]
```

Note the use of currying for the inner map

Exercises

Use `map` to implement the following functions

1. Write a function `cubeList` that takes a list of numbers and returns a list containing the cubes of those numbers

2. Write a function `middleElem` that takes a list of three-element tuples and returns a list containing the second element of each tuple

3. Write a function `ruinStrings` that takes a list of strings and returns the same list where every instance of the character `'e'` is replaced by `'x'`