COMP105 Lecture 8

List Recursion

Recap: lists

Lists have a **head** and a **tail**

```
ghci> head [1,2,3,4]
1
ghci> tail [1,2,3,4]
[2,3,4]
```

We will use these to do **recursion** on lists

A recursive list function

A function that adds all elements of a list together

```
sum' []     = 0
sum' (x:xs) = x + sum' xs
```

- The base case is the empty list
- The recursive rule breaks the list into its head and tail

Sum in action

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

$\rightarrow$ `1 + sum [2,3]`

$\rightarrow$ `1 + 2 + sum [3]`

$\rightarrow$ `1 + 2 + 3 + sum []`

$\rightarrow$ `1 + 2 + 3 + 0`

$\rightarrow$ `6`

The length of a list

```
length' []     = 0
length' (_:xs) = 1 + length' xs
```

This time we don't even care what the head of the list is

Building a list recursively

We can also **build** lists using recursion

```
down_from 0 = []
down_from x = x : down_from (x-1)
```

```
ghci> down_from 5
[5,4,3,2,1]
```

down_from in action

```
down_from 3

→  3 : down_from 2

→  3 : (2 : down_from 1)

→  3 : (2 : (1 : down_from 0))

→  3 : (2 : (1 : []))
```

Transforming lists

We can write recursive functions that **transform** lists

A function that squares every element of a list

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

The base case and recursive rule both return lists

Square_list in action

```
square_list [2,3,4]
→ 4 : square_list [3,4]
→ 4 : (9 : square_list [4])
→ 4 : (9 : (16 : (square_list [])))
→ 4 : (9 : (16 : []))
→ [4,9,16]
```

Exercises

1. Write a function `product'` that multiplies all elements of a list together

2. Write a function `upToTen` that takes one argument `x` and outputs a list containing all numbers between `x` and 10 inclusive

3. Write a function `halveList` that divides each element of a list by two