Introduction to Lists

(background for lab 1)

Lists

- … are very common in Haskell
- Example list: [1,2,3,4]
- Use : to add an element in front of a list:

```
*Main> 0:[1,2,3]
[0,1,2,3]
```

- Lists can be created by enumeration:

```
*Main> [1..10]
[1,2,3,4,5,6,7,8,9,10]
```

Some list operations

- From the Data.List module:

```
reverse        :: [a] -> [a]
  -- reverse a list

take           :: Int -> [a] -> [a]
  -- (take n) picks the first n elements

(++)           :: [a] -> [a] -> [a]
  -- append a list after another

replicate      :: Int -> a -> [a]
  -- make a list by replicating an element
```

[a] is the type of a list whose elements have type a

Some list operations

```
*Main> reverse [1,2,3]
[3,2,1]

*Main> take 4 [1..10]
[1,2,3,4]

*Main> [1,2,3] ++ [4,5,6]
[1,2,3,4,5,6]

*Main> replicate 5 2
[2,2,2,2,2]
```

Strings are lists of characters

```
type String = [Char]

Prelude> 'g' : "apa"
"gapa"

Prelude> "flyg" ++ "plan"
"flygplan"

Prelude> ['A','p','a']
"Apa"
```

Type synonym definition

Processing lists

- Lists can be processed using *list comprehension*:

```
squareEach :: [Integer] -> [Integer]
squareEach xs = [x*x | x <- xs]

allCAPS :: String -> String
allCAPS s = [toUpper c | c <- s]

*Main> squareEach [1,2,3]
[1,4,9]

*Main> allCAPS "Chalmers"
"CHALMERS"
```