Lists

Lists: recap

- Can represent 0, 1, 2, … things
  - [], [3], [”apa”,”katt”,”val”,”hund”]
- They all have the same type
  - [1,3,True,”apa”] is not allowed
- The order matters
  - [1,2,3] /= [3,1,2]
- Syntax
  - 5 : (6 : (3 : [])) == 5 : 6 : 3 : [] == [5,6,3]
  - ”apa” == [’a’,’p’,’a’]   (type String = [Char])

Lists

-- how they work

Can we define Lists as a datatype?

```
data List = Empty | Add ?? List
```

- Our attempt at a ”home made” list is either:
  - An empty list
  - Formed by *adding an element* to a smaller list
- What to put on the place of the ??

Lists

```
data List a = Empty | Add a (List a)
```

A type parameter

- Add 12 (Add 3 Empty) :: List Int
- Add ”apa” (Add ”bepa” Empty) :: List String
- Haskell’s built-in lists can be thought of as a syntactic shorthand for this datatype

Haskell's lists

```
-- psudocode for Haskell lists
data [a] = [ ]  |  a : [a]
```

compare with

```
data List a = Empty | Add a (List a)
```

More on Types

- Functions can have ”general” types:
  - *polymorphism*
  - reverse :: [a] -> [a]
  - (++) :: [a] -> [a] -> [a]
- Sometimes, these types can be restricted
  - Ord a => … for comparisons (<, <=, >, >=, …)
  - Eq a => … for equality (==, /=)
  - Num a => … for numeric operations (+, -, *, …)

Example: "Quicksort"

```
qsort :: Ord a => [a] -> [a]

qsort []     = []
qsort (x:xs) = qsort small ++ [x] ++ qsort big
         where small = [y | y <- xs, y < x]
               big   = [z | z <- xs, z >= x]
```

sort lists of any type *a*, as long as *a* has comparison functions

```
qsort :: Ord a => [a] -> [a]

qsort []     = []
qsort (x:xs) = qsort small ++ [x] ++ qsort big
         where small = [y | y <- xs, y < x]
               big   = [z | z <- xs, z >= x]
```

list append

Introduces **local** definitions

definitions must be left-aligned

Some Examples from the Standard Prelude

[Demo in class]

- reverse a list
- append two lists
- append a list of lists
- take the first n elements from a list
- drop the first n elements from a list
- "zip" two lists together

see course book p121, 126-127