Functies in haskell

Opgave 11

Erik Kooistra

UvA

February 14, 2019

Partial application

Een manier om functies om te vormen naar andere functies.

```
add      :: Int -> Int -> Int

addOne =  (+) 1

addOne 4

AddOne :: Int -> Int
```

Higher order functions

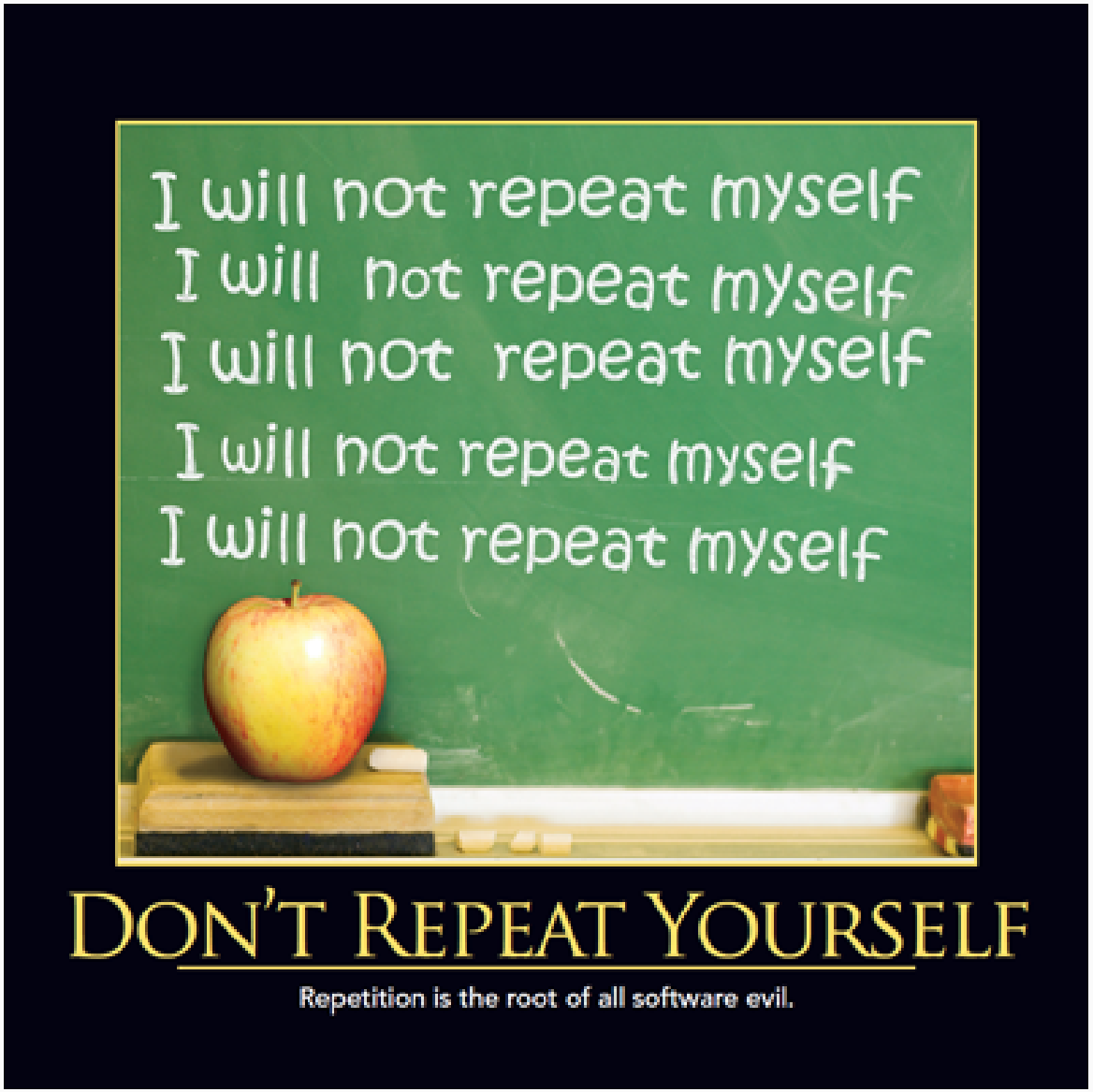
```
doubleList []     = []
doubleList (x:xs) = 2*x : doubleList xs
tripleList []     = []
tripleList (x:xs) = 3*x : tripleList xs
```

Higher order functions 2

ghci

```
Prelude> :t doubleList
doubleList :: Num a => [a] -> [a]
Prelude> :t tripleList
tripleList :: Num a => [a] -> [a]
```

DRY


I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
I will not repeat myself
DON'T REPEAT YOURSELF
Repetition is the root of all software evil.

Multi list

stukke flexibeler

```
multList :: Num t => t -> [t] -> [t]
```

Multi list

```
multList n [] = []
multList n (x:xs) = n*x : multList n xs
doubleList = multList 2
```


WE NEED TO GO
DEEPER

operlist

```
operlist n bop [] = []
operlist n bop (x:xs) =
          bop n x : operlist n bop xs
```


WE NEED TO GO
DEEPER

Map

```
mapList f [] = []
mapList f (x:xs) = f x : mapList f xs
```

Opgave 11

Oplossing

Examples

```
AddList l i w i2
        | i2 == i = w
        | otherwise l i2
```