Purely Functional Algorithm Specification
Exercises Day 2

Jan van Eijck
CWI & ILLC, Amsterdam

ESSLLI, Opole, August 7, 2012

homepages.cwi.nl/~jve/courses/12/esslli12/

```
module Exerc2

where
```

Exercises With Foldr

1. Define `length` in terms of `foldr`.
2. Define `elem x` in terms of `foldr`.
3. Find out what `or` does, and next define your own version of `or` in terms of `foldr`.
4. Define `map f` in terms of `foldr`.
5. Define `filter p` in terms of `foldr`.
6. Define `(++)` in terms of `foldr`.
7. Define `reversal` in terms of `foldr`.

Exercise with Foldl

```
for :: [a] -> (a -> b -> b) -> b -> b
for [] f y = y
for (x:xs) f y = for xs f (f x y)
```

8. Show that the function `for` that defines the for loop is a variant of `foldl`, by giving a definition of `for` in terms of `foldl`.

Hint: you will also need `flip`, for flipping the arguments of a function of type `a -> b -> c`.

Hoare Reasoning about GCD

```
euclidGCD :: Integer -> Integer -> Integer
euclidGCD = while2
              (\ x y -> x /= y)
              (\ x y -> if x > y
                        then (x-y,y)
                        else (x,y-x))
```

9. State a suitable loop invariant for the while loop in Euclid's GCD algorithm (the function `euclidGCD`).

Hoare Reasoning about Squaring

```
sqr :: Int -> Int
sqr y = let
          x = 0
          n = 0
        in sqr' y n x

sqr' y = while2
          (\ n _ -> n < y)
          (\ n x -> (n+1,x + 2*n + 1))
```

10. State a suitable loop invariant for the while loop in the squaring function (the function `sqr'`).