Purely Functional Algorithm Specification
Exercises Day 2 — With Answers

Jan van Eijck
CWI & ILLC, Amsterdam

ESSLLI, Opole, August 7, 2012

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

```
module Answers2

where
```

Exercises With Foldr

1. Define `length` in terms of `foldr`.

   Answer:

```
myLength = foldr (\ _ n -> n+1) 0
```

2. Define `elem x` in terms of `foldr`.

   Answer:

```
myElem x = foldr (\ y b -> x == y || b) False
```

3. Find out what `or` does, and next define your own version of `or` in terms of `foldr`.

   Answer:

```
myOr = foldr (||) False
```

4. Define `map f` in terms of `foldr`.

   Answer:

```
myMap f = foldr (\ x xs -> f x : xs) []
```

5. Define `filter p` in terms of `foldr`.

   Answer:

```
myFilter p = foldr (\ x xs -> if p x then x:xs else xs)
```

6. Define `(++)` in terms of `foldr`.

   Answer:

```
myAppend = flip (foldr (:))
```

7. Define `reversal` in terms of `foldr`.

   Answer:

```
reversal' = foldr (\ x xs -> xs ++ [x]) []
```

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`.

   Answer:

   ```
   myFor xs f y = foldl (flip f) y xs
   ```

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`).

   Answer: there several possibilities.

   Let f be the step function of the while loop, and let (x', y') be fxy. Then an obvious choice for the loop invariant is the statement that the set of divisors does not change in the step from x, y to x', y'.

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'`).

    Answer:

    Let n, x be the inputs to the step function for `sqr'` and let (n', x') be the output. Then $x = n^2$ and $x' = n'^2$ is an obvious choice.