Purely Functional Algorithm Specification
Exercises Day 3

Jan van Eijck
CWI & ILLC, Amsterdam

ESSLLI, Opole, August 8, 2012

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

```
module Exerc3

where
import List
import While
import Assert
import Reasoning (update,updates)
import AssertiveCoding
```

Exercise

`merge` can be used as follows, to create a function for list sorting:

```
mergeSrt ::  Ord a => [a] -> [a]
mergeSrt [] = []
mergeSrt (x:xs) = merge [x] (mergeSrt xs)
```

Find a suitable assertion, and write an assertive version of this.

Follow-up Exercise

Another approach to merge sort is to start by splitting the list in equal parts, recursively sort the parts, next merge.

Implement this, using the following split function.

```
split :: [a] -> ([a],[a])
split xs = let
   n = (length xs) `div` 2
  in
   (take n xs, drop n xs)
```

Next, find a suitable assertion, and write an assertive version.

Testing Euclid's Algorithm

Suppose we want to write code for performing automated tests on Euclid's algorithm for large ranges of values, as follows:

```
myEuclid :: Integer -> Bool
myEuclid k = let
    triv = (\_ -> True)
  in
    and [ triv $ (assert2 isGCD euclid) n m |
                        n <- [1..k], m <- [1..k] ]
```

Use your Haskell system to check if this is a reasonable test.

If there is something wrong, can you correct it?