Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inf. P. Michel
Dipl.-Inf. C. Feller

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Exercise Sheet 6: Specification and Verification with Higher-Order Logic (Summer Term 2012)

Date: 23.05.2012

Exercise 1 Case Study: Greatest Common Divisor

You have seen a *model*, *property* and *proof* for the Euclidean algorithm in the lecture. We want to keep the model, but specify the property differently and prove it.

```
fun gcd :: "nat ⇒ nat ⇒ nat" where
  "gcd m 0 = m"
| "gcd m n = gcd n (m mod n)"
```

a) Start a new theory file and prove that the function `gcd` computes the greatest common divisor of `m` and `n`:

   1. The result of gcd divides both arguments, i.e., its a common divisor.
   2. The result of gcd is greater than (or equal to) every common divisor, provided not both arguments are zero.

   You can prove (1) directly, but for (2) it might be useful to define and prove the following two properties of `gcd` first:

   - Each common divisor divides the result of `gcd`.
   - The result of `gcd` is not zero if at least one argument is not zero.

   From this you can prove (2) by showing that a natural number dividing a non-zero natural number has to be less than (or equal to) that number.

   Hint: In Isabelle/HOL, the property that a divides b is expressed by: `a dvd b`.

b) Prove the following property of `gcd`: `k * gcd m n = gcd (k * m) (k * n)`.

c) Consider a slightly different implementation of the greatest common divisor function:

```
fun gcd :: "nat ⇒ nat ⇒ nat" where
  "gcd m n = (if n = 0 then m else gcd n (m mod n))"
```

   - Prove that this implementation is equivalent to the first one.
   - Prove the property of b) for this implementation.

d) Use the main property of a) to define the greatest common divisor directly (not recursively) with the Hilbert-Choice operator (`SOME`), in particular not using the Euclidean algorithm.

   Prove the equivalence of this function to the original `gcd`.