Algorithms and Data Structures
Jacobs University Bremen
Dr. Florian Rabe

Homework 1
given: 2017-02-07

Homework 1

You have to submit your solutions as announced in the lecture.
Unless mentioned otherwise, all problems are due 2017-02-16
There will be no deadline extensions unless mentioned otherwise in the lecture.

Problem 1.1 *Euclidean Algorithm*

Points: 4+4+4

Implement the Euclidean algorithm in multiple programming languages.

Specifically, pick one language from **each** of the following groups:

- untyped languages: Python, Javascript, PHP
- typed languages favoring imperative programming: Java, C++
- typed languages favoring functional programming: SML, Scala, Haskell

Your implementations should work with *arbitrary* natural numbers, i.e., numbers of unlimited size. The technical term for that is *arbitrary precision arithmetic*. All programming languages provide that in one way or another. But the built-in type *int* may or may not offer arbitrary precision—if it does not, large numbers such as 35! will cause overflow errors. See https://en.wikipedia.org/wiki/List_of_arbitrary-precision_arithmetic_software for an overview of how to do arbitrary precision arithmetic in different languages.

Hint: If you do not want to install multiple programming languages, you can use a number of websites to program online. Just google `online ide` to find several good sites.

Problem 1.2 *Rational Numbers*

Points: 5+5+5

The data structure of rational numbers consists of the following:

- a rational number is a pair of an integer d and a positive natural number e (both using arbitrary precision)
- two rational numbers (d, e) and (d', e') are equal iff $d \cdot e' = e \cdot d'$
- the normal form of a rational number (d, e) is $(d/g, e/g)$ where $g = \gcd(d, e)$

For two rational numbers, we can define several operations

- addition: $(d, e) + (d', e') := (d \cdot e' + e \cdot d', e \cdot e')$
- additive inverse (negative): $-(d, e) := (-d, e)$
- multiplication: $(d, e) + (d', e') := (d \cdot d', e \cdot e')$
- multiplicative inverse: $1/(d, e) := (e, d)$

Implement this data structure and all mentioned operations in **the same three** programming languages as above. Use the Euclidean algorithm to compute the normal form.

Hint: If you do not understand the definitions, work out some examples. It should become clear quickly.