Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Theoretische Informatik
Dr. B. Gärtner, Prof. Dr. J. Hromkovič

6. Dezember 2011

Informatik für Mathematiker und Physiker Serie 11 HS 11

URL: `http://www.ti.inf.ethz.ch/ew/courses/Info1_11/`

Skript-Aufgabe 130 (4 Punkte)

Define a type `Z_7` for computing with integers modulo 7. Mathematically, this corresponds to the finite ring $\mathbb{Z}_7 = \mathbb{Z}/7\mathbb{Z}$ of residue classes modulo 7.

For the type `Z_7`, implement addition and subtraction operators

```
// POST: return value is the sum of a and b
Z_7 operator+ (Z_7 a, Z_7 b);

// POST: return value is the difference of a and b
Z_7 operator- (Z_7 a, Z_7 b);
```

according to the following table (this table also defines subtraction: $x - y$ is the unique number $z \in \{0, \ldots, 6\}$ such that $x = y + z$).

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Skript-Aufgabe 139 (4 Punkte)

We want to have a function that *normalizes* a rational number, i.e. transforms it into the unique representation in which numerator and denominator are relatively prime, and the denominator is positive. For example,

$$\frac{21}{-14}$$

is normalized to

$$\frac{-3}{2}.$$

There are two natural versions of this function:

```
// POST: r is normalized
void normalize (rational& r);

// POST: return value is the normalization of r
rational normalize (const rational& r);
```

Implement one of them, and argue why you have chosen it over the other one.

Hint: you may want to use the function `gcd` from section 3.2, modified for arguments of type `int` (how does this modification look like?).

Skript-Aufgabe 142 (8 Punkte)

The C++ standard library also contains a type for computing with *complex numbers*. A complex number where both the real and the imaginary part are doubles has type `std::complex<double>` (you need to `#include <complex>` in order to get this type). In order to get a a complex number with real part r and imaginary part i, you can use the expression

```
std::complex<double>(r,i); // r and i are of type double
```

Otherwise, complex numbers work as expected. All the standard operators (arithmetic, relational) and mathematical functions (`std::sqrt`, `std::abs`, `std::pow`...) are available. The operators also work in mixed expressions where one operand is of type `std::complex<double>` and the other one of type `double`. Of course, you can also input and output complex numbers.

Here is the actual exercise. Implement the following function for solving quadratic equations over the complex numbers:

```
// POST: return value is the number of distinct complex solutions
//       of the quadratic equation ax^2 + bx + c = 0. If there
//       are infinitely many solutions (a=b=c=0), the return
//       value is -1. Otherwise, the return value is a number n
//       from {0,1,2}, and the solutions are written to s1,..,sn
int solve_quadratic_equation (std::complex<double> a,
                              std::complex<double> b,
                              std::complex<double> c,
                              std::complex<double>& s1,
                              std::complex<double>& s2);
```

Test your function in a program for at least the triples (a, b, c) from the set

$$\{(0,0,0),(0,0,2),(0,2,2),(2,2,2),(1,2,1),(i,1,1)\}.$$

Die **Aufgaben 127** aus den Vorlesungsunterlagen ist die **Challenge Aufgabe** und gibt 8 Punkte.

Abgabe: Bis 13. Dezember 2011, 15.15 Uhr.