Skript-Aufgabe 130 (4 Punkte)

Define a type \( \mathbb{Z}_7 \) 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 \( \mathbb{Z}_7 \), implement addition and subtraction operators

\[
\begin{align*}
\text{Z}_7 \text{ operator+} & (\text{Z}_7 \text{ a, } \text{Z}_7 \text{ b}); \\
\text{Z}_7 \text{ operator-} & (\text{Z}_7 \text{ a, } \text{Z}_7 \text{ b});
\end{align*}
\]

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

<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

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:

\[
\begin{align*}
\text{// POST: r is normalized} \\
\text{void normalize (rational& r);} \\
\text{// POST: return value is the normalization of r} \\
\text{rational normalize (const rational& r);} \\
\end{align*}
\]

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 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:

```cpp
// 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.