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

9. Dezember 2008

Informatik für Mathematiker und Physiker Serie 13 HS 08

URL: http://www.ti.inf.ethz.ch/ew/courses/Info1_08/

Aufgabe 1 (4 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 `double`s 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 task: 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 (b=c=d=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);
```

Write a program that tests your function. For example, you may substitute the solutions returned by the above function into $ax^2 + bx + c = 0$ and check whether the expression indeed evaluates to (approximately) zero.

Aufgabe 2 (4 Punkte)

Rewrite the following recursive function in iterative form and test with a program whether your iterative version is correct. What can you say about the runtimes of both variants for values of n up to 100, say?

```
unsigned int f (unsigned int n)
{
if (n == 0) return 1;
return f(n-1) + 2 * f(n/2);
}
```

Aufgabe 3 (4 Punkte)

Consider the following recursive function defined on all nonnegative integers, also known as *McCarthy's 91 Function*.

$$M(n) := \begin{cases} n - 10, & \text{if } n > 100 \\ M(M(n+11)), & \text{if } n \leq 100. \end{cases}$$

a) Provide a C++ function `mccarthy` that implements McCarthy's 91 Function.

b) What are the values of the following four function calls?

(i) `mccarthy(101)`

(iii) `mccarthy(100)`

(iii) `mccarthy(99)`

(iv) `mccarthy(91)`

c) Explain why the function is called McCarthy's 91 Function! More precisely, what is the value of $M(n)$ for any given number n?

Aufgabe 4 (4 Punkte)

A struct may have data members of struct type that may have data members of struct type, and so on. It can be quite cumbersome to set all the data members of the data members of the data members manually whenever an object of the struct type needs to be initialized. A preferrable solution is to write a function that does this. As an example, consider the type

```
struct rational_vector_3 {
rational x;
rational y;
rational z;
};
```

where `rational` is as before defined as

```
struct rational {
int n;
int d; // INV: d != 0
};
```

Write a function

```
rational_vector_3 create_rational_vector_3
(int n1, int d1, int n2, int d2, int n3, int d3)
```

that returns the rational vector $(n_1/d_1, n_2/d_2, n_3/d_3)$. The function should also make sure that d_1, d_2, d_3 are nonzero.

Abgabe: Bis 16. Dezember 2008, 15.15 Uhr.