

Informatik für Mathematiker und Physiker **Serie 11** **HS 09**URL: http://www.ti.inf.ethz.ch/ew/courses/Info1_09/**Skript-Aufgabe 122 (4 Punkte)**

The *Towers of Hanoi* puzzle (that can actually be bought from shops) is the following. There are three wooden pegs labeled 1, 2, 3, where the first peg holds a stack of n disks, stacked in decreasing order of size, see Figure Abbildung 1.

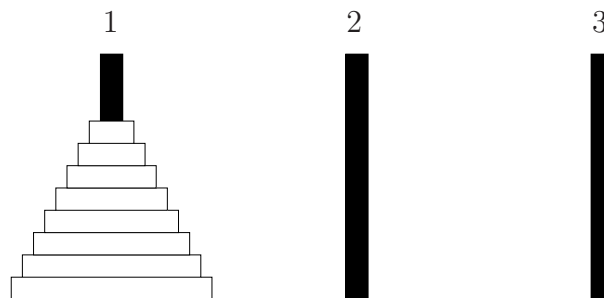


Abbildung 1: The Tower of Hanoi

The goal is to transfer the stack of disks to peg 3, by moving one disk at a time from one peg to another. The rule is that at no time, a larger disk may be on top of a smaller one. For example, we could start by moving the topmost disk to peg 2 (move (1,2)), then move the next disk from peg 1 to peg 3 (move (1,3)), then move the smaller disk from peg 2 onto the larger disk on peg 3 (move (2,3)), etc.

Write a program `hanoi.cpp` that outputs a sequence of moves that does the required transfer, for given input n . For example, if $n = 2$, the above initial sequence (1,2)(1,3)(2,3) is already complete and solves the puzzle. Check the correctness of your program by hand at least for $n = 3$, by manually reproducing the sequence of moves on a piece of paper (or an actual Tower of Hanoi, if you have one).

By the way, if you search for Towers of Hanoi on the Internet you will find a lot of information. There's also an iPhone application, and incidentally somebody who programmed a robot to play the game on his iPhone. Watch <http://www.youtube.com/watch?v=ETNpYCq2AN8>, it's really funny.

Skript-Aufgabe 128 (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 $z \in \{0, \dots, 6\}$ 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 140 (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:

```
// 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  $s_1, \dots, s_n$ 
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 125 und 133 aus den Vorlesungsunterlagen sind die **Challenge Aufgaben** und geben jeweils 8 Punkte, wenn sie vollständig gelöst werden.

Abgabe: Bis 8. Dezember 2009, 15.15 Uhr.