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

29. November 2011

Informatik für Mathematiker und Physiker Serie 10 HS 11

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

Skript-Aufgabe 116 (6 Punkte)

In how many ways can you own CHF 1? Despite its somewhat philosophical appearance, the question is a mathematical one. Given some amount of money, in how many ways can you partition it using the available denominations (bank notes and coins)? Today's denominations in CHF are 1000, 200, 100, 50, 20, 10 (banknotes), 5, 2, 1, 0.50, 0.20, 0.10, 0.05 (coins). The amount of CHF 0.20, for example, can be owned in four ways (to get integers, let's switch to centimes): $(20), (10, 10), (10, 5, 5), (5, 5, 5, 5)$. The amount of CHF 0.04 can be owned in no way, while there is exactly one way to own CHF 0.00 (you cannot have 4 centimes in your wallet, but you *can* have no money at all in your wallet).

Solve the problem for any given input amount, by writing a program `partition.cpp` that defines the following function (all values to be understood as centimes).

```
// PRE: [first, last) is a valid nonempty range that describes
//      a sequence of denominations d_1 > d_2 > ... > d_n > 0
// POST: return value is the number of ways to partition amount
//       using denominations from d_1, ..., d_n
unsigned int partitions (unsigned int amount,
                         const unsigned int* first,
                         const unsigned int* last);
```

Use your program to determine in how many ways you can own CHF 1, and CHF 10. Can your program compute the number of ways for CHF 50? For CHF 100?

Skript-Aufgabe 121 (5 Punkte)

The following function finds an element with a given value x in a sorted sequence (if there is such an element).

```
// PRE: [first, last) is a valid range, and the elements *p,
//      p in [first, last) are in ascending order
// POST: return value is a pointer p in [first, last) such
//       that *p = x, or the pointer last, if no such pointer
//       exists
const int* binary_search (const int* first, const int* last, const int x)
{
  const int n = last - first;
  if (n == 0) return last;          // empty range
  if (n == 1) {
    if (*first == x)
      return first;
    else
      return last;
  }
  // n >= 2
  const int* middle = first + n/2;
  if (*middle > x) {
    // x can't be in [middle, last)
    const int* p = binary_search (first, middle, x);
    if (p == middle)
      return last; // x not found
    else
      return p;
  } else
    // *middle <= x; we may skip [first, middle)
    return binary_search (middle, last, x);
}
```

What is the maximum number $T(n)$ of comparisons between sequence elements and x that this function performs if the number of sequence elements is n? Try to find an upper bound on $T(n)$ that is as good as possible. (You may use the statement of Exercise 122.)

Skript-Aufgabe 124 (5 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.

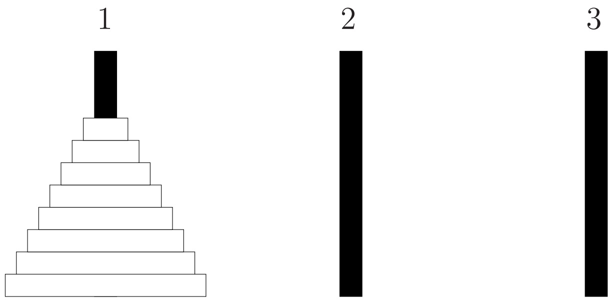

Abbildung 1: The Tower of Hanoi

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).

Die **Aufgaben 125** und **126** aus den Vorlesungsunterlagen sind die **Challenge Aufgaben** und geben jeweils 8 Punkte, wenn sie vollständig gelöst werden.

Abgabe: Bis 6. Dezember 2011, 15.15 Uhr.