Winfried Auzinger
Dirk Praetorius

Sommersemester 2014
9.–11. April 2014

Übungsaufgaben zur VU Computermathematik

Serie 5

Generelle Anmerkung zu den Maple-Übungen:

Die Aufgabenstellungen sind in englischer Sprache formuliert. (Nebeneffekt: ein bisschen gewöhnen an die englische Fachsprache.) Nehmen Sie sich die Zeit, die Aufgabenstellungen genau durchzulesen; diese enthalten auch Hintergrundinformationen über das jeweilige Thema und können daher gelegentlich etwas ausführlicher geraten. Viele der Übungsaufgaben sind keine reinen 'Maple-Aufgaben', sondern enthalten auch eine mathematische Problemstellung, die Sie, ggf. mit entsprechenden Hinweisen, zunächst verstehen bzw. knacken sollen. Manche andere wieder sind experimenteller Natur (was für den Physiker das Labor ist, ist für den Mathematiker der Computer). Und bedenken Sie: Der Name der LVA ist Computermathematik.

Für vielen Fragestellungen gibt es innerhalb von Maple schon fertige Lösungen Es spricht aber nichts dagegen, so etwas als Übungsaufgabe zu verwenden (auch in anderen Übungen berechnen oder beweisen Sie Dinge, die schon andere vor Ihnen berechnet bzw. bewiesen haben).

Es kommt gelegentlich vor, dass Sie einen Befehl benötigen, der in der Vorlesung (noch) nicht besprochen wurde Das ist kein Drama; für derartige Fälle werden Hinweise gegeben, manchmal einfach nur das richtige Stichwort, für das Sie Details in der Hilfe nachschlagen können. Lesen Sie die Hinweise genau durch; manches müssen Sie ggf. noch selbst herausfinden. Orientieren Sie sich auch mit Hilfe des Flyers (siehe Homepage). Manche der Aufgaben haben auch den Zweck, dass Sie sich einen in der Vorlesung (aus Zeitgründen) nicht oder noch nicht im Detail besprochenen Stoff aktiv anhand von Beispielen selbst erarbeiten. Nützen Sie generell die Maple-Hilfe systematisch – für die praktische Arbeit ist dies unumgänglich.

Ihre Codes sollten Sie so weit wie möglich auf Korrektheit testen. Dokumentieren Sie Ihre Worksheets auch in angemessener Weise mittels Zwischentexten bzw. Kommentaren (`#`). Das Hauptziel dabei sollte immer sein, dass Sie selbst später noch erkennen können, was sie sich dabei gedacht haben.

Exercise 5.1. Design a function `binomi(x,y,n)` which uses `binomial(...)` and returns the expression

$$\sum_{k=0}^{n} \binom{n}{k} x^k\, y^{n-k} \quad (n \in \mathbb{N})$$

a) First version: Use `sum` and see what happens if you call `binomi` with an unassigned or assigned variable `n`.

b) Second version: Use `add` and see what happens if you call `binomi` with an unassigned or assigned variable `n`. What happens if you `factor` the later result?

c) Generate a list of lists representing a part of the Pascal triangle.

Hint: Use `seq` in a nested ('geschachtelt') way.

Exercise 5.2. The *multinomial coefficient* is a generalization of the binomial coefficient: Let $k = (k_1, \dots, k_d) \in \mathbb{N}_0^d$ with $|k| := k_1 + \cdots + k_d = n$, Then,

$$\binom{n}{k} := \frac{n!}{k_1! \cdots k_d!}$$

a) Design a function `mnc(n,k)` which expects a integer number `n` and a list of integer numbers `k` as its arguments and returns the multinomial coefficient.

Hint: Use `!` and `mul` do evaluate the denominator.

b) 'Verify' by examples the identity

$$\binom{n}{k} = \prod_{j=1}^{d} \binom{\sum_{\ell=1}^{j} k_\ell}{k_j}$$

Exercise 5.3.

a) By default, lists of the same length are added element-wise: `[a,b]+[c,d]` gives `[a+c,b+d]`.

Design a function `lmul(l[1],l[2])` which expects two lists of the sam length as its arguments and returns the list containing the products `l[1][1]*l[2][1]`, `l[1][2]*l[2][2]`, ...

Hint: Use `seq`.

b) Design an a function `lip(l[1],l[2])` which returns the inner product of the lists `l[1]` and `l[2]`, i.e., the sum of the products of their elements.

Hint: Use `add`.

c) Design a function `lop(l[1],l[2])` which returns the outer product of the lists `l[1]` and `l[2]`, i.e., a list of lists `[L[1],L[2],...]` with `L[i][j]=l[1][i]*l[2][j]`. All occurring lists have the same length.

Hint: Use `seq`. (In linear algebra terms, think of `l[1]`, `l[2]` as vectors and of `[L[1],L[2],...]` as a matrix.)

Exercise 5.4. Assume that `A` und `B` are finite sets. Design functions which expect sets a arguments and perform the following operations:

a) $(\mathtt{A}, \mathtt{B}) \mapsto (\mathtt{A} \setminus \mathtt{B}) \cup (\mathtt{B} \setminus \mathtt{A})$

b) $(\mathtt{A}, \mathtt{B}, \mathtt{x}, \mathtt{y}) \mapsto$ `true`, if $\mathtt{x} \in \mathtt{A}$, $\mathtt{y} \in \mathtt{B}$, $\mathtt{x} \notin \mathtt{B}$, and $\mathtt{y} \notin \mathtt{A}$, and `false` otherwise. Use `evalb`.

c) $(\mathtt{A}, \mathtt{B}) \mapsto \mathtt{A} \times \mathtt{B}$ (Cartesian product), a set of ordered pairs which you can represent by lists `[a,b]` of length 2.

Hint: Use `seq(...)`.

Exercise 5.5.

a) Design a function `numdigits(x)` which returns the number of decimal digits of an integer.

Hint: Use `ilog10`.

b) Design a function `sumdigits(n)` which expects an integer number as its argument and returns the sum of its decimal digits.

Hint: Use `iquo` and `mod`.

Exercise 5.6. A bit of summation.

a) Design a function `geomsum(x,k,n)` which returns $\sum_{j=k}^{n} x^j$ without explicitly summing up the x^j.

Compare with `sum`. What about $x = 1$?

b) Design a function `cumsum(L)` which expects a list `L` of numbers as its argument and returns its cumulated sum, i.e., the sum of all partial sums `L[1], L[1]+L[2], ...L[1]+...+L[nops(L)]`. Use `add` in a nested way.

c) Optimize `cumsum` such that only a single call of `add` is required.

d) Check the result of `sum(n^2,n)` and interpret it.

Exercise 5.7.

a) Design a function `islin(A,B,C)` which expects 3 lists `A=[`a_1, a_2`]`, `B=[`b_1, b_2`]`, and `C=[`c_1, c_2`]` with integer entries as its arguments. `A,B,C` represent Cartesian coordinates of points in the plane, and your function returns `true` if these are located on a common line, otherwise `false`.

Hint: There are different possibilities to check this. The best way is to use a determinant. Use `evalb` to check whether this determinant is nonzero.

b) Design a function `det3(M)` which computes the determinant of a 3×3-matrix `M`, represented as a list of length 3, where the elements of `M` are also lists of length 3 representing the rows of the matrix.

c) Design a function `tri(n)` which for a given positive integer `n`, generates the list

```
[[0,0],[1,0],[1,1],[2,0],[2,1],[2,2],[3,0],...,[n,n-1],[n,n]]
```

Use `seq`.

d) A list `L` is *palindromic* if `L[`i`]=L[`$n+1-i$`]` for $i = 1 \ldots n$, where n denotes the length of `L`.

Design a function `ispalindromic(L)` which returns `true` if `L` is palindromic, and `false` otherwise.

Exercise 5.8.

a) Given a 'black box' function `f`: $\mathbb{R} \to \mathbb{R}$, you want to check whether it is affine, i.e., $f(x) = a\,x + b$ with some $a, b \in \mathbb{R}$. Design a function `checkaffine(f)` which returns `true` if `f` is affine, otherwise `false`.

Hint: Call `f` with unassigned arguments `x` and `y`, and use `simplify` and `evalb`.

b) Design a function `phi(f,x)` which returns the arithmetic mean $\frac{1}{n}\sum_{i=1}^{n} x_i$ for a given function `f` and a list `x` containing the values x_i (n = length of `x`).

c) Design a function `oddeven(f)` which returns, for a given function `f`, a list of two functions, namely the even and odd part of `f`.
