Winfried Auzinger
Dirk Praetorius

Sommersemester 2016
12. April 2016

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 oft auch Hintergrundinformationen über das jeweilige Thema und können daher gelegentlich etwas ausführlicher geraten. Was *konkret zu tun* ist, ist jeweils in *kursiver Schrift* ausformuliert.

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 immer wieder vor, dass Sie etwas benötigen, das in der Vorlesung (noch) nicht besprochen wurde. Für derartige Fälle werden Hinweise gegeben, manchmal einfach nur das richtige Stichwort, für das Sie Details in der Hilfe nachschlagen können. Lesen Sie die Angaben und 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, z.B. was die Erstellung von Grafiken, Animationen etc. betrifft.

Nützen Sie generell die Maple-Hilfe systematisch – für die praktische Arbeit ist dies unumgänglich. Es sei auch darauf hingewiesen, dass das Verhalten eines derartigen Systems nicht immer genau vorhersehbar ist und man daher manches durch Ausprobieren herauszufinden versuchen wird. (Auch von Version zu Version kann sich das Verhalten manchmal ändern.)

Ihre Codes sollten Sie so weit wie möglich anhand von Beispielen testen. Bei der Vorführung im Computerlabor sollen Sie in dieser Weise die Korrektheit Ihrer Ausarbeitungen dokumentieren.

Dokumentieren Sie Ihre Worksheets auch in angemessener Weise mittels Zwischentexten bzw. Kommentaren (`#`). Das wesentliche Ziel dabei sollte immer sein, dass Sie selbst später noch erkennen können, was sie sich dabei gedacht haben.

Aufgaben mit $(*)$ (kommt manchmal vor) sind ein wenig anspruchsvoller.

$\longrightarrow$

Exercise 5.1: ***Basic syntax, simple functions.***

a) The 'from-to' operator `..` is used to represent a range of integers.[1]

Check what happens when you apply `seq` *to an object* `m..n`.

What is the meaning of `m..n` *if* m=n *or* m>n*? Check this.*

b) *Design a function* `rangetolist(intrange)` *which expects an expression* intrange *of the type* `..`*, i.e., of the form* `m..n` *as its argument and converts it to the list*

`[m,m+1,...,n]`

The case when m=n *or* m>n *should also be handled in a correct way.*

c) *Design a function* `cumsum(L)` *which expects a list* L *as its argument and returns the cumulated sum*

$$\sum_{k=1}^{n} \sum_{j=1}^{k} L_j$$

where L_j *are the elements of the list* L *and* n *is the number of elements in* L.

Hint: The number of elements of a list is obtained using `nops(...)` or `numelems(...)`.

Exercise 5.2: ***Manipulation of a symbolic expression.***

This exercise is related to the binomial formula

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k\, b^{n-k}.$$

a) *Try to* `expand` *the expression* `(a+b)^n`*, where* `n` *has no value assigned. Observe what happens here.*

b) *Repeat the same experiment with particular numerical values for* `n`*, and check the behavior of* `factor` *applied to the result.*

c) *Design a function* `binomi(a,b,n)` *which returns the expression*

$$\sum_{k=0}^{n} \binom{n}{k} a^k\, b^{n-k}$$

Test your function with different numerical values for `n` *and compare the results with the results obtained in* **b)** *– they should be identical.*

Hint: Use `add` and `binomial`.

d) *Modify your function from* **c)**, *using* `sum` *instead of* `add`*. Check what happens if you call this function with* `n` *not assigned a numerical value.*

Hint: It is usually a good idea to try to `simplify` the result.

In this particular case, explicitly declaring `n` to be a positive integer via `assume(n,posint)` has an influence on the result.

[1] Remark: `..` is also used to represent real intervals $[a, b]$, e.g., when specifying the lower and upper limits of a definite integral $\int_a^b \ldots$ (details to be discussed later on).

Exercise 5.3: ***List manipulation.***

a) *Design a function* `reverse(L)` *which expects a list L as its argument and returns this list with the order of elements reversed.*

b) *Design a function* `ispalindromic(L)` *which returns* `true` *if the list L is palindromic and* `false` *otherwise.*

Hint: Use **a)**, `=`, and `evalb`. (Note that `=` works for comparing not only single objects but also lists, for instance.)

c) The function from **b)** can be realized more efficiently without explicitly generating the reversed list, by directly comparing the first with the last element of the list and so on. To this end you can use the construct

`'if'(`*bexpr,t,f*`)`

which returns the value t if the Boolean expression *bexpr* is true and f otherwise. Example:

`'if'(x=0,0,1)`

returns 0 if $x = 0$ and 1 otherwise.

Realize such a version. Note that there is some potential for optimization (with a little bit more programming effort) since it is only necessary to scan half of the list.

Exercise 5.4: ***Discrete sets and functions.***

a) *Design two functions* `add_element(`*elem,A*`)` *and* `remove_element(`*elem,A*`)` *which expects any object (elem) and a set A as its arguments and returns a new set with elem added to or removed from A, respectively.*

Hint: Use standard set operations.

For **b)**–**d)**, let $A := \{1, \ldots, n\}$ ($n \in \mathbb{N}$) and $f\colon A \to \mathbb{N}$ be a given function (choose your own functions f for testing).

b) *Design a function* `img(f)` *which returns the image of f, i.e., the set* $f(A) = \{f(a)\colon a \in A\}$.

c) *Design a function* `isinjective(f)` *which returns* `true` *if f is injective and* `false` *otherwise.*

Hint: Use `'if'`.

d) Inverse problem: Assume f is injective, i.e., f considered as $f\colon A \to f(A)$ is bijective. Think about the problem of coding the inverse function $g := f^{-1}\colon f(A) \to A$. How to realize this in an efficient way will depend on the type of f and of the size of n.

Realize a variant where, first, you store the image $f(A)$ in a list. Then, your implementation of $g = f^{-1}$ makes use of this list. Return `NULL` *(= 'nothing') if your function* `g` *is called with an argument which is not contained in the image of f.*

Remark: Without using an explicit loop (to be discussed later on) this may be a little but tricky to realize (why?). Alternatively, you may look into the `ListTools` package for assistance (`? ListTools`).

Exercise 5.5: ***A combinatorial task.***

We consider lists containing 0 or 1 as elements.[2] Assume two such lists L and K of the same length be given. Reverse the number of elements of L, and afterwards replace 0 by 1 and 1 by 0, respectively. If the outcome of this operation is identical with the list K then we call L and K to be *'twins'*. We call L a *'selfie'* if it is its own twin.

Example: `[0,0,1,0]` and `[1,0,1,1]` are twins, and `[0,0,1,1]` is a selfie.

a) *Design a function* `istwin(L,K)` *which returns* `true` *if* `L` *and* `K` *are twins and* `false` *otherwise.*

Hint: Use `seq` and `'if'`. For simplicity you may assume that L and K are given with the same length. Alternatively, you may try to check this and return `false` if this is not the case.

b) *Design a function* `isselfie(L)` *which returns* `true` *if L is a selfie and* `false` *otherwise.*

---

[2] This example seems artificial, but there are combinatorial applications where such operations are of relevance. We also could simply consider binary integers or strings composed of '0' or '1', but using lists is somewhat more convenient here.

Exercise 5.6: ***Calculus with polynomials.*** [3]

Let $n \in \mathbb{N}$ be given and $a_0 + a_1\,x + \ldots + a_n\,x_n$ be an arbitrary polynomial expression of degree $\leq n$. We represent it by a list $p = [a_0, a_1, \ldots, a_n]$ of length $n+1$. (Warning: the first index in a list is always 1, so you have to reorganize your indices.

a) *Design a function* `evaluate(p,x)` *which evaluates the value* $p(x)$.

b) *Design a function* `derivative(p)` *which returns the polynomial* p' *again in form of a list of coefficients.*

c) *Design a function* `antiderivative(p,C)` *which returns the coefficients of the antiderivative (*Stammfunktion*) of* p *again in the same format, adding* C *as constant of integration.*

d) *Design a function* `definite_integral(p,a,b)` *which computes* $\int_a^b p(x)\,dx$ *using* `antiderivative`.

Hint: You may compare with the results provided by the built-in derivative `diff` and integral `int`.

Exercise 5.7: ***Symbolic summation.***

a) *Use* `sum` *to try to evaluate the sums*

$$\sum_{k=1}^{n} k, \quad \sum_{k=1}^{n} k^2, \quad \sum_{k=1}^{n} k^3, \ \ldots \qquad \sum_{k=1}^{n} \frac{1}{k}, \quad \sum_{k=1}^{n} \frac{1}{k^2}, \ \ldots$$

$$\sum_{k=1}^{n} x^k, \quad \sum_{k=1}^{n} k\,x^k, \quad \sum_{k=1}^{n} k^2\,x^k, \quad \ldots, \quad \sum_{k=1}^{n} k^p\,x^k$$

Here n, x, p *are symbolic objects (unassigned, i.e., no value assigned, and* n,p *are to be interpreted as positive integers). Check what happens, and try to convert the result into a form as simple as possible.*

b) *For* **a)** *you have used the syntax* `sum(...,k=1..n)`. *Do the same again using*

`sum(...,k)`

Denoting the resulting value by S_k, *compute/simplify the expression* $S_{k+1} - S_k$. *What do you observe? Try to interpret this.*

Hint: Think of `sum(...,k)` as a discrete analogue of `int(...,x)` (indefinite integration).

Exercise 5.8: ***Basic plots.***

Besides the simple `plot` command for real functions, there are many more commands for plotting functions or discrete data. (In particular, the package `plots` contains many special versions.) Consult the help pages.

a) *Use* `plot` *to plot a real functions of your choice. Check the help page and try to generate a 'nice' plot by adjusting parameters. E.g., modifying the default values of the parameters* `color`, `thickness`, `axes` *(and many others) may be useful.*

b) *A discrete analog of* `plot` *is* `plots[listplot]` [4] *contained in the package* `plots`. *Use* `listplot` *to visualize the discrete function* $f\colon \mathbb{N} \to \mathbb{N}$, *defined by*

$$f(n) := 0 \text{ if } n \text{ is not prime, and } n - p(n) \text{ otherwise, where } p(n) \text{ is the largest prime number} < n.$$

Here, e.g., modifying the default values for the parameters `color`, `symbol`, `symbolsize`, *and* `style` *(among others) may be particularly useful in order to produce a nice-looking listplot.*

Hint: Use `'if'(...)`, `isprime(...)` and `prevprime(...)`.

Warning: `prevprime(2)` is not defined.

---

[3] All the manipulations in this exercise can be performed directly in Maple without using lists. Therefore, consider this just as an exercise – you are building your own small computer algebra system for the manipulation of polynomials whose coefficients are represented by lists. (MATLAB, for instance, handles polynomial expressions in a similar way.)

[4] This syntax calls `listplot` without loading the complete contents of the `plots` package into memory.