Winfried Auzinger
Dirk Praetorius

Sommersemester 2013
24./25. April 2013

Übungsaufgaben zur VU Computermathematik

Serie 5

Einige Themen,wie z.B. plot-Befehle, Integration mittels `int` und das `‘if‘`-Konstrukt, wurden in der ersten VO noch nicht besprochen. Dafür werden jedoch genaue Hinweise gegeben. Schauen Sie auch in die Maple-Hilfe.

Es sei darauf hingewiesen, dass wir hier nach wie vor noch keine ‘echten’ Prozeduren (Unterprogramme), sondern nur Funktionen verwenden, weiters keine expliziten Steuerkonstrukte wie `if`, `do` (Schleifen) etc. In den folgenden Aufgaben lässt sich alles durch ‘Einzeiler’ mittels impliziter Schleifen ausdrücken, unter Verwendung von `seq(...)`, `add(...)`, etc. In Aufgabe 5.5 verwenden Sie die Kurz-Syntax `‘if‘(...)`, um eine einfache Fallunterscheidung zu realisieren.

Exercise 5.1.

a) Use the command[1] `plots[listplot]` to plot the points $(x_n, y_n) = (n, p_n)$ in the (x, y)-plane. Here, $p_n =$ `ithprime(n)` is the n-th prime number. For larger values of n, e.g., n up to 1000, this looks pretty regular.

b) Generate (e.g., for n up to 1000) a list of the numbers

$$\frac{n \ln n}{p_n}$$

evaluated using `evalf`. This gives a conjecture concerning the asymptotic distribution of prime numbers within the natural numbers. Which one?

Exercise 5.2. Indexing of objects of type `list` begins with 1. However, sometimes it is more useful to start, e.g., with 0 (like in C). This is supported by the data type `Array` (later), but you can also realize this ‘manually’ using lists, as an exercise:

a) Design a function `el(l,pos,startindex)` which assumes that indexing begins with the integer `startindex` and returns the `pos`-th element of the list `l` with respect to this indexing. E.g.,

```
el([1,2,3,4],-1,-2)  -->   2
```

b) Design a function `elC(l,pos)` which works like `el(l,pos,0)`, e.g.,

```
elC([1,2,3,4],2)  -->   3
```

c) Design a function `setelC(l,pos,value)` which assigns the value `value` to the `pos`-th element in the list `l` with respect to ‘C-indexing’ according to **b)**, and returns the new, modified list as its function value. E.g.,

```
setelC([1,2,3,4],3,999)  -->   [1,2,3,999]
```

Hint: This would be easy to realize within a procedure. But since `setelC` should be a function (a ‘one-liner’),

```
setelC := (l,pos,value) -> ...
```

you have to construct a ‘formula’ which generates the modified list.

---

[1] Syntax: `plots[listplot](...)` activates the function `listplot` contained in the package `plots`. You can also activate the complete package using `with(plots);` and simply call `listplot`.

Remark: Concerning storage, it would be more efficient to modify the given list instead of creating a new one.[2] This way of manipulating data by functions or procedures is called *call by reference*, and is not directly supported in Maple. (In C, you would pass a pointer to the object). On the other hand, you can 'overwrite' the old list with the new one:

```
l := setelC(l,pos,value);
```

Exercise 5.3.

a) Design a function `merge(L,K)` which merges two lists `L,K` into a single list according to

```
[L[1],K[1],L[2],K[2],...]
```

If one of the lists is longer than the other one, the remaining elements are appended at the end of the new list. Make sure that empty lists are correctly handled.

Hint: Use `seq`. Find out how to generate a sequence of sequences `L[i],K[i]`.

b) Design a function `listp(L,K)` which returns a list of lists as indicated in the following example:

```
> L := [1,2,3]:
> K := [A,B,C,D]:
> listp(L,K);

    [[[1,A],[1,B],[1,C],[1,D]], [[2,A],[2,B],[2,C],[2,D]], [[3,A],[3,B],[3,C],[3,D]]]
```

Hint: Use `seq`.

Exercise 5.4. Use `plots[pointplot]` to draw a regular polygon with n vertices (e.g., an equal-sided triangle for $n = 3$, a square for $n = 4$). Pack the your plot command into a function `draw_polyhedron(n,...)` which also expects more parameters, e.g., for the color or line thickness of the plot (these are passed as options to `pointplot`).

Exercise 5.5.

a) Functions can also be recursive. Consider the following recursive definition:

$$f_0(x) = \frac{e^x - 1}{x}\,, \qquad f_n(x) := \frac{1}{x}\Big(-1 + \sum_{j=0}^{n-1} \frac{f_j(x)}{n-j}\Big), \quad n = 1, 2, 3, \ldots$$

Design a function `frec(x,n)` which implements this recursion.

Hint: For the recursion you need an `if`-construct. Use

`'if'(n=0,` *value for* $n = 0$`,` *value for* $n \neq 0$`);`

(`'if'` with backward quotes).

b) Verify experimentally for $n = 0, 1, 2, 3$ that

$$f_n(x) = (-1)^n \int_0^1 e^{(1-t)\,x} \binom{-t}{n}\, dt\,.$$

(In fact, this true for all $n \in \mathbb{N}$, but the proof is not so easy.) Use `int(...,t=0..1)` for evaluating the integrals. Here,

$$\binom{\tau}{n} = \frac{\tau(\tau-1)\cdots(\tau-n+1)}{n!}\,, \quad \tau \in \mathbb{R},\ n \in \mathbb{N}$$

is the generalized binomial coefficient (`binomial(tau,n)`), a polynomial of degree n in τ. You have to `expand` this expression under the integral to make it work (try without and with `expand`).

[2] This issues are, of course, only relevant for codes manipulating very large objects. In most cases, you will not care very much about this.

Exercise 5.6. Some analysis:

Use `sum(...,n=0..infinity)` to 'verify' the following Taylor series expansion[3] ($x, \tau \in \mathbb{R}$):

$$(1+x)^\tau = \sum_{n=0}^{\infty} \binom{\tau}{n} x^n$$

This is the *binomial series* – a generalization of the binomial theorem. Evidently, `sum` 'knows' this series – it is built-in as *static knowledge* like many other series expansions from analysis. Now we ask ourselves for what values of x this series is convergent (Maple does not provide this information to you). Use the definition of the generalized binomial coefficient and the quotient criterion to decide on this question. This requires a little bit of computation which you can do by hand or with the help of Maple.

Can you also prove that the limiting function is indeed $f(x) = (1+x)^\tau$?

The series is also convergent for $x = 1$. Check using `sum` – a nice formula.

Exercise 5.7. The *Möbius strip* is a two-dimensional manifold in $\mathbb{R}^3$ which is not orientable. It can be represented by the coordinate functions

$$\begin{aligned} x(\varphi,t) &= \cos\varphi\,(r + t\cos(\varphi/2)),\\ y(\varphi,t) &= \sin\varphi\,(r + t\cos(\varphi/2)), \qquad\qquad \varphi \in [0,2\pi],\ t \in [-a,a]\\ z(\varphi,t) &= t\,\sin(\varphi/2)). \end{aligned}$$

Here, r is the radius, and $2a$ is the width of the strip. Consult `?plot3d` in order to produce a nice plot.

Note that you also can manipulate the plot interactively in various ways (right mouse button).

Furthermore, animate such a plot using `plots[animate3d]` and produce a video. Here the principle is that a parameter, e.g., the radius, can be varied, and `animate3d` produces a sequence of the corresponding plots. `animate3d` works like `plot3d` but with an additional argument controlling the animation. After calling `animate3d` you see the first frame and you can start the video interactively. Export the video to `gif` format.

Exercise 5.8. A matrix[4] $M \in \mathbb{R}^{m\times n}$ (m rows, n columns) can be represented by a list of length m consisting of lists of length n (the rows of M).

a) Design a function `mcount(M)` which returns the number of elements in such an object which are not equal to 0.

b) Design a function `mfliplr(M)` which reverses the ordering of elements in each row and returns the result in the same format.

c) Design a function `mrowtocol(M)` which converts such an object into the analogous 'columnwise' representation, where the role of rows and columns is interchanged.

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

---

[3] The proof of the validity of this series expansion is an elementary using Taylor's theorem. But this is not the topic of this exercise.

[4] Lists are usually only efficient for storing static, 'read-only' objects, but for 'small' objects efficiency is not an issue. A more economical and versatile way to represent (larger) matrices is to use the data type `Array` or `Matrix`. This will be discussed later on in the lecture.