Uebungen ▮ Exercices

11. Problemsammlung zur Musterentsprechung (Mustererkennen, musterkonformes Abarbeiten) ▮ Collection de problèmes concernant les correspondances et patrons (reconnaître les patrons, travailler conformement aux patrons)

Die Gliederung dieses Kurses folgt in groben Zügen dem Buch von Nancy Blachman: A Practical Approach.... Hinweis: Kapitel 11 lesen!
▮ L'articulation de ce cours correspond à peu près à celle du livre de Nancy Blachman: A Practical Approach....
Indication: Lire le chapitre 11.

Run mit WIN+*Mathematica* Version 5.2
▮ Testé avec *Mathematica* version 5.2+WIN

Aufgabe 1 ▮ Problème 1

Schreibe eine Funktion "geometrisches Mittel", die das geometrische Mittel einer Liste von Zahlen ausgibt.
Ecris une fonction "moyenne géométrique", qui donne la moyenne géométrique d'une liste de nombres.

Z.B. das geometrische Mittel von {a,b,c,d,e} ist (a b c d e)^(1/5).
▮ P.ex. la moyenne géométrique de {a,b,c,d,e} est (a b c d e)^(1/5).

```
In[1]:= Clear[geometrMittel];
        geometrMittel[{x__}]:= Times[x]^(1/Length[{x}])

In[3]:= geometrMittel[{1,1,1,1,1}]

Out[3]= 1
```

```
In[4]:= geometrMittel[{2,16,8,4,32}]
Out[4]= 8

In[5]:= geometrMittel[{2,3,5}]
Out[5]= 30^(1/3)

In[6]:= geometrMittel[{2,3,5}] // N
Out[6]= 3.10723
```

Aufgabe 2 ▮ Problème 2

Schreibe eine eigene Version der "Join-Funktion". ▮ Ecris ta propre version de la fonction "Join".

"Join" fuegt zwei Listen hintereinander im Gegensatz zur Mengenvereinigung, wo mehrfach vorkommende Elemente nur einmal aufgefuehrt werden.
▮ "Join" rassemble deux listes l'une après l'autre. Dans la réunion des ensembles par contre les éléments doubles, triples etc. n'apparaissent qu'une seule fois.

```
In[7]:= Clear[joinFkt];
        joinFkt[{x__},{}]:= {x};
        joinFkt[{},{x__}]:= {x};
        joinFkt[{x__},{y__}]:= Flatten[{{x},{y}}]
```

Ausprobieren: ▮ Essayer:

```
In[11]:= joinFkt[{a,b,c,d},{e,f,g}]
Out[11]= {a, b, c, d, e, f, g}

In[12]:= joinFkt[{a,b,c,d},{a,e,b,f,g}]
Out[12]= {a, b, c, d, a, e, b, f, g}

In[13]:= joinFkt[{a,b,c,d},{}]
Out[13]= {a, b, c, d}

In[14]:= joinFkt[{},{}]
Out[14]= joinFkt[{}, {}]
```

Was ist passiert? Erweitere die Definition sinngemäss!
▮ Que s'est-il passé? Elargis la définition d'une façon sensée!

Aufgabe 3 ▮ Problème 3

Baue eine Funktion "Fold" wie unten beschrieben
▮ Construis une fonction "Fold" comme il est décris ci-dessous.

a) Implementiere wie nachstehend folgt. Beantworte dann die nachstehenden Fragen b), c), d) und vergleiche mit den implementierten Funktionen "Fold" resp. "FoldList".

▮ "Implement" comme il suit. Réponds aux questions suivantes b), c), d) et compare avec les fonctions "implémentées" "Fold" resp. "FoldList".

```
In[15]:= Clear[falte];
         falte::usage = "falte[f, basis, liste] ergibt
                     --donne  f[...[f[f[basis,x1]x2],...xn]
                     wobei --étant  list = {x1,x2,...,xn}";
         falte[f_,basis_,{}]:=basis;
         falte[f_,basis_,{x1_,xrest___}]:=
                falte[f,f[basis,x1],{xrest}];
```

b) Was macht "falte[Plus,0,liste]"? Z.B. liste={a,b,c}.

▮ Que fait "falte[Plus,0,liste]"? P.ex. liste={a,b,c}.

```
In[19]:= falte[Plus,0,{a,b,c}]
Out[19]= a + b + c

In[20]:= falte[Plus,0,{1,2,3,4}]
Out[20]= 10

In[21]:= falte[Plus,5,{1,2,3,4}]
Out[21]= 15
```

c) Was macht "falte[Max,-Infinity,liste]"? Z.B. liste={a,b,c}.

▮ Que fait "falte[Max,-Infinity,liste]"? P.ex. liste={a,b,c}.

```
In[22]:= falte[Max,-Infinity,{a,b,c}]
Out[22]= Max[a, b, c, -∞]

In[23]:= falte[Max,-Infinity,{6,8,3}]
Out[23]= 8

In[24]:= falte[Max,9,{6,8,3}]
Out[24]= 9
```

d) Vergleiche mit "Fold" und "FoldList":

▮ Compare avec "Fold" et "FoldList":

In[25]:= **??Fold**

```
Fold[f, x, list] gives the last element of FoldList[f, x, list]. Mehr...

Attributes[Fold] = {Protected}
```

In[26]:= **??FoldList**

```
FoldList[f, x, {a, b, ... }] gives {x, f[x, a], f[f[x, a], b], ... }. Mehr...

Attributes[FoldList] = {Protected}
```

Aufgabe 4 ▮ Problème 4

Schreibe die APL-Funktion "Deal" resp. "?" in *Mathematica*: ▮ Ecris la fonction APL "Deal" resp. "?" dans *Mathematica*:

"L ? R" in APL wählt (streicht) aus dem "Range[r]" pseudo-zufällig L Zahlen aus, ohne sie in R zu ersetzen.
▮ "L ? R" dans APL choisit (biffe) dans le "Range[r]" d'une façon pseudo-aléatoire L nombres, sans les remplacer dans R.

a) Was ist "Range" - Wie erzeuge ich die "Zufallszahlen"?
▮ Qu'est "Range" - Comment générér les "nombres aléatoires"?

In[27]:= **??Range**

```
Range[imax] generates the list {1, 2, ... , imax}. Range[imin, imax] generates
   the list {imin, ... , imax}. Range[imin, imax, di] uses step di. Mehr...

Attributes[Range] = {Listable, Protected}
```

In[28]:= **Range[14]**

Out[28]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

In[29]:= **??Random**

```
Random[ ] gives a uniformly distributed pseudorandom Real in the range
   0 to 1. Random[type, range] gives a pseudorandom number of the specified
   type, lying in the specified range. Possible types are: Integer, Real and
   Complex. The default range is 0 to 1. You can give the range {min, max}
   explicitly; a range specification of max is equivalent to {0, max}. Mehr...

Attributes[Random] = {Protected}
```

b) Die Beschreibung der Funktion "deal" in *Mathematica*:
▮ La description de la fonction "deal" dans *Mathematica*:

```
In[30]:= RemoveAll[deal];
         deal::usage="deal[n,r] streicht zufällig n Integers
                      aus der Liste  Range[r] ohne Ersetzung.
                      -- biffe de façon aléatoire n Integers
                      dans la liste Range[r] sans les
                      remplacer."
```

```
Out[31]= deal[n,r] streicht zufällig n Integers               aus der Liste
           Range[r] ohne Ersetzung.              -- biffe de façon aléatoire n
           Integers              dans la liste Range[r] sans les               remplacer.
```

c) Hier eine mögliche Variante:

■ Voici une variante possible:

```
In[32]:= Clear[deal];
         deal[0,r_] :=Range[r];
         deal[n_,r_]:=Sort[Rest[RotateLeft[deal[n-1,r],
                               Random[Integer,{1,r}]]]]
```

d) Ausprobieren:

■ Essaier

```
In[35]:= deal[2,6]
```

```
Out[35]= {1, 2, 4, 5}
```

e) Erklärungen:

■ Explications:

```
In[36]:= ??Rest
```

```
Rest[expr] gives expr with the first element removed. Mehr...

Attributes[Rest] = {Protected}
```

```
In[37]:= ??RotateLeft
```

```
RotateLeft[expr, n] cycles the elements in expr n positions to the left.
   RotateLeft[expr] cycles one position to the left. RotateLeft[expr, {n1,
   n2, ... }] cycles elements at successive levels ni positions to the left. Mehr...

Attributes[RotateLeft] = {Protected}
```

```
In[38]:= RotateLeft[{a,b,c},11]
```

```
Out[38]= {c, a, b}
```

```
In[39]:= ??Sort
```

```
Sort[list] sorts the elements of list into canonical
   order. Sort[list, p] sorts using the ordering function p. Mehr...

Attributes[Sort] = {Protected}
```

Aufgabe 5 ∎ Problème 5

Zu Funktionen, die als Argument eine variable Zahl von Listen aufnehmen: Quant aux fonctions qui acceptent comme argument un nombre variable de listes:

Das Muster "x___List" als Argument in einer Funktion lässt eine variable Zahl von Listen zu. Wie kann man auf einen Schlag alle Längen einer Anzahl beteiligter Listen finden? Schreibe eine Funktion dafür!
∎ Le patron "x___List" comme argument dans une fonction permet un nombre variable de listes. Comment peut-on, d'un coup, trouver toutes les longueurs d'un nombre de listes faisant partie du problème? Ecris une fonction pour cela!

```
In[40]:= Clear[listenLaengen];
         listenLaengen[x___List]:=Map[Length, {x}]
```

Ausprobieren: ∎ Essayer:

```
In[42]:= listenLaengen[{1,2,3}, {1,2,3,4,5}, {1,1,1,1},
                       {a,b,c,d,e,f,g}]

Out[42]= {3, 5, 4, 7}
```

Aufgabe 6 ∎ Problème 6

Lauflängencodierung in Listen (Häufigkeitszählungen) ∎ Comper des fréquences dans les listes

Konstruiere eine kurze *Mathematica*-Funktion, die die Häufigkeit der Elemente einer Liste zählt. Die Funktion "laufCod" soll eine Liste von Paaren {Listenelement, absolute Häufigkeit} ausgeben Beispiel:
liste = {e,b,a,b,e,c,c,d,e,a,a,d,u};
laufCod[liste] soll das Resultat {{a, 3}, {b, 2}, {c, 2}, {d, 2}, {e, 3}, {u, 1}} ergeben. Die Funktion "laufDeCod" soll aus dem Output von "laufCod" wieder die ursprüngliche Liste machen.
∎ Construis une fonction de *Mathematica* courte qui compte la fréquence des éléments d'une liste. La fonction "laufCod" doit sortir une liste de paires {élément de liste, fréquence absolue}. Exemple:
liste = {e,b,a,b,e,c,c,d,e,a,a,d,u};
laufCod[liste]doit donner le résultat
{{a, 3}, {b, 2}, {c, 2}, {d, 2}, {e, 3}, {u, 1}}. La fonction "laufDeCod" doit transformer l'output de "laufCod" de nouveau dans la liste d'origine.

a) laufCod

```
In[43]:= laufCod[liste_List]:=Transpose[{Union[liste],
                       Table[Count[liste,Part[Union[liste]
                       ,n]], {n,Length[Union[liste]]}]}]
```

(Hat Schweiss gekostet!) Aber jetzt ausprobieren:
∎ (Cela nous a fait transpirer!) Mais maintenant on essaie:

```
In[44]:= RemoveAll[liste];
         liste = {1,1,2,2,2,3,3,3,3};
         output = laufCod[liste]

Out[46]= {{1, 2}, {2, 3}, {3, 4}}

In[47]:= liste = {e,b,a,b,e,c,c,d,e,a,a,d,u};
         output = laufCod[liste]

Out[48]= {{a, 3}, {b, 2}, {c, 2}, {d, 2}, {e, 3}, {u, 1}}
```

a) laufDeCod

```
In[49]:= laufDeCod[output_]:= If[MatrixQ[output],
                     output/.{x_,n_}:> Flatten[
                                   Table[x,{n}]],                    Print["Bitte
                    Print["Bitte Matrix eingeben --
                           S.v.p. entrer la matrice"]]

In[50]:= laufDeCod[{{1,2},{3,5},{4,3}}]

Out[50]= {{1, 1}, {3, 3, 3, 3, 3}, {4, 4, 4}}

In[51]:= laufDeCod[{{a,3},{b,5},{c,2},{d,4}}]

Out[51]= {{a, a, a}, {b, b, b, b, b}, {c, c}, {d, d, d, d}}
```

c) Eine andere Variante für "laufCod" mit Ersetzungsregeln (funktioniert nur für eine Liste von Integers)

Une autre variante pour "laufCod" avec des règles de remplacement (ne fonctionne que pour une liste d'Integers)

```
In[52]:= RemoveAll[f];
         f[liste_List]:=Module[{i},
         i=0;
         a=liste;
         a//.{y___,x_,x_,z___}:>
         {y,x,++i b,x,z}//.{y___,x_Integer,w_Integer,z___}:>
         {y,x,d, c,w,z}/. {x___}:>{c, x, d}//.
         {x___,y___ b,z___}:>{x,z}//. {x___,b,z___}:>{x,z}//.
         {y___,c,x__,d,z___}:> {y,{x},z}
         ];
         laufCod1[liste_List]:={Map[First,f[liste]],
                                Map[Length,f[liste]]}

In[55]:= f[{1,1,1,2,2,2,2,2,3,3,3,3,4,5,5}]

Out[55]= {{1, 1, 1}, {2, 2, 2, 2, 2}, {3, 3, 3, 3}, {4}, {5, 5}}

In[56]:= laufCod1[{1,1,1,2,2,2,2,2,3,3,3,3,4,5,5}]

Out[56]= {{1, 2, 3, 4, 5}, {3, 5, 4, 1, 2}}

In[57]:= laufCod1[{0,0,0,1,1,2,2,2,3,3,3,3,4,4,4,5,5,6,6,6}]

Out[57]= {{0, 1, 2, 3, 4, 5, 6}, {3, 2, 3, 4, 3, 2, 3}}
```

Was ist los?

Que se passe-t-il?

```
In[58]:= laufCod1[{a,a,a,1,b,b,b,2,2,2,3,3,3,3,4,4,4,5}]

Out[58]= {{{0, 0, 0, d, {1, 1}, {2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4}, {5, 5}, c, 6, 6, 6}, 3, 4, 5},
          {7, 4, 3, 1}}
```

d) Noch eine andere Variante:

■ Encore une autre variante:

d) Noch eine andere Variante für "laufCod" mit Hilfe anonymer Funktionen (vgl. Kap. 12):

■ Encore une autre variante pour "laufCod" à l'aide de fonctions anonymes (comparer chap. 12):

```
In[59]:= Attributes[mal]={Flat};
         mal[{elem_,m_},{elem_,n_}]:=mal[{elem,m+n}];
         laufCod2[liste_List]:= List @@ mal @@({#,1}& /@
                                liste)

In[62]:= laufCod2[{1,1,1,2,2,2,2,2,3,3,3,3,4,5,5}]

Out[62]= {{1, 3}, {2, 5}, {3, 4}, {4, 1}, {5, 2}}
```

Wie arbeitet dieses Programm?

■ Comment ce programme travaille-t-il?

steht für "anonyme Variable", & für "anonyme Funktion", /@ für "Map".
Durchlaufen wird die Liste.

■ # signifie "variable anonyme", & signifie "fonction anonyme", /@ signifie "Map". La liste est parcourue.

```
In[63]:= out1=({#,1}& /@ {1,1,1,2,2,2,2,2,3,3,3,3,4,5,5})

Out[63]= {{1, 1}, {1, 1}, {1, 1}, {2, 1}, {2, 1}, {2, 1}, {2, 1},
          {2, 1}, {3, 1}, {3, 1}, {3, 1}, {3, 1}, {4, 1}, {5, 1}, {5, 1}}
```

@@ steht für "Apply".

■ @@ signifie "Apply".

```
In[64]:= out2=mal @@out1

Out[64]= mal[{1, 3}, {2, 5}, {3, 4}, {4, 1}, {5, 2}]

In[65]:= ??List

        {e1, e2, ... } is a list of elements. Mehr...

        Attributes[List] = {Locked, Protected}

In[66]:= out3=List @@ out2

Out[66]= {{1, 3}, {2, 5}, {3, 4}, {4, 1}, {5, 2}}
```

Probiere eigene Varianten aus!

■ Essaie des variantes à toi!

Testgelände:
Terrain de test:

In[67]:=

"Putzmaschine" einsetzen
Employer la "machine de nettoyage"

In[68]:= **(* Old Form: Remove["Global`@*"] *)**

In[69]:= **Remove["Global`*"]**