commalists-tools-l3

LaTeX3

Macros for manipulating numeral comma separated lists: sorting, adding, removing, etc

Version 0.20b – 24/03/2026

Cédric Pierquet
c pierquet – at – outlook . fr
https://github.com/cpierquet/latex-packages/tree/main/commalists-tools

```
\def\mytestlist{14,10,15,11,9,10}
\ctlenoflist*{\mytestlist}
\ctminoflist*{\mytestlist}
\ctmaxoflist*{14,10,15,11,9,10}
\ctsortasclist*{\mytestlist}
\ctmeanoflist*{\mytestlist}
\ctremovevalinlist*{10}{\mytestlist}
\cttestifvalinlist{15}{\mytestlist}{true}{false}
\ctgetvaluefromlist*{\mytestlist}{3}
\ctgetindexfromlist*{15}{\mytestlist}
\ctsublist*{\mytestlist}{*}{4}
```

We consider the list: 14,10,15,11,9,10

There's 6 values in the list

The minimum value is 9 and the maximum is 15

Ascending sorted list is 9,10,10,11,14,15

Mean value of the list is 11.5

If we remove the value 10, then the list is 14,15,11,9

We test if 15 is in the list: true

The third value of the list is 15

15 is in index 3 in the list

Sublist [-,4] of the list is: 14,10,15,11

Contents

1 Introduction

In order to load `commalists-tools-l3`, simply use:

```
\usepackage{commalists-tools-l3}
```

All code is written in LaTeX3, so no extra packages are needed.

💡 See also the `tuple` package on CTAN, which provides a more complete and fully expandable approach to numeral list operations, with an elegant `object.method` chaining syntax!

2 Global usage

2.1 Tools

Package `commalists-tools-l3` supports (basic) manipulations on numeral comma separated lists:

- sorting;
- adding values;
- removing values;
- counting values;
- mean, sum, product;
- get value, get length;
- etc.

Starred versions only print the result, whereas non starred versions store the result into a macro.

💡 Macros are prefixed with `\ct...` (for commalists-tools).

2.2 The macro for printing

```
%just printing, with optional separator
\ctshowlist[sep]{list}
```

```
\ctshowlist{1,   2,3, 4, 6}
```

```
1,2,3,4,6
```

```
\def\mytmplist{12, -4,   5,7 ,8, 9, 0}
\ctshowlist[\,/\,]{\mytmplist}
```

```
12/-4/5/7/8/9/0
```

3 The macros for calculating

3.1 Length, minimum, maximum

```
%getting length of list, only printing
\ctlenoflist*{list}
%storing length of list into macro (\resmylen by default)
\ctlenoflist{list}
%getting min of list, only printing
\ctminoflist*{list}
%storing min of list into \macro (\resmin by default)
\ctminoflist{list}[\macro]
%getting max of list, only printing
\ctmaxoflist*{list}
%storing max of list into \macro (\resmax by default)
\ctmaxoflist{list}[\macro]
```

```
%only printing
\ctlenoflist*{14,10,15,11,9,10}\\
%only printing
\ctminoflist*{14,10,15,11,9,10} and \ctmaxoflist*{14,10,15,11,9,10}
```

6
9 and 15

```
%storing len/min/max of list
\def\mytestlist{10,14.5,20,12,8.5}
\ctlenoflist{\mytestlist}[\mylen]Length of list is \mylen\ \&
\ctminoflist{\mytestlist}[\mymin]Min of list is \mymin\ \&
\ctmaxoflist{\mytestlist}[\mymax]Max of list is \mymax
```

Length of list is 5 & Min of list is 8.5 & Max of list is 20

3.2 Mean, sum, prod

```
%getting mean of list, only printing
\ctmeanoflist*{list}
%storing mean of list into \macro (\resmean by default)
\ctmeanoflist{list}[\macro]
%getting sum of list, only printing
\ctsumoflist*{list}
%storing max of list into \macro (\ressum by default)
\ctsumoflist{list}[\macro]
%getting prod of list, only printing
\ctprodoflist*{list}
%storing max of list into \macro (\resprod by default)
\ctprodoflist{list}[\macro]
```

```
%only printing
\ctmeanoflist*{14,10,15,11,9,10} \\
%storing
\ctmeanoflist{14,10,15,11,9,10}[\mymean]\mymean \\
%only printing
\ctsumoflist*{14,10,15,11,9,10} and \ctprodoflist*{14,10,15,11,9,10} \\
%storing
\ctsumoflist{14,10,15,11,9,10}[\mysum]\ctprodoflist{14,10,15,11,9,10}[\myprod]
The sum is \mysum\ and the prod is \myprod
```

11.5
11.5
69 and 2079000
The sum is 69 and the prod is 2079000

4 Macros for manipulating

4.1 Sorting

```
%sorting (asc), only printing
\ctsortasclist*{list}
%sorting (asc) and storing (overwrite)
\ctsortasclist{list}
%sorting (des), only printing
\ctsortdeslist*{list}
%sorting (des) and storing (overwrite)
\ctsortdeslist{list}
```

```
%sorting (asc), only printing
\ctsortasclist*{14,10,15,11.5,9,10}\\
%sorting (asc) and storing within \myreslist
\def\tmpsortlist{14,10,15,11.5,9,10}
\ctsortasclist{\tmpsortlist}[\myreslist]\myreslist\\
%analysing
\readlist*\tmpSORTlist{\myreslist}
\showitems{\tmpSORTlist}
```

9,10,10,11.5,14,15
9,10,10,11.5,14,15
$\boxed{9}\boxed{10}\boxed{10}\boxed{11.5}\boxed{14}\boxed{15}$

```
%sorting (des), only printing
\ctsortdeslist*{14,10,15,11.5,9,10}\\
%sorting (asc) and storing within \myreslist
\def\tmpsortlist{14,10,15,11.5,9,10}
\ctsortdeslist{\tmpsortlist}[\myreslist]\myreslist\\
%analysing
\readlist*\tmpSORTlist{\myreslist}
\showitems{\tmpSORTlist}
```

15,14,11.5,10,10,9
15,14,11.5,10,10,9
$\boxed{15}\boxed{14}\boxed{11.5}\boxed{10}\boxed{10}\boxed{9}$

4.2 Extract element, add/remove element

```
%extract value, only printing
\ctgetvaluefromlist*{list}{index}
%extract value and storing into macro (\myelt by default)
\ctgetvaluefromlist{list}{index}[\macro]
```

```
%extract value, only printing
\def\listtmp{1,2,3,6,3,1,5,7,3}%
\ctgetvaluefromlist*{\listtmp}{4}\par\smallskip
%storing
\ctgetvaluefromlist{\listtmp}{-1}[\mylastelt]The last element is \mylastelt
```

6

The last element is 3

```
%extract value (cyclic list), only printing
\ctgetvaluefromcycllist*{list}{index}
%extract value (cyclic list) and storing into macro (\myelt by default)
\ctgetvaluefromcycllist{list}{index}[\macro]
```

```
%extract value (cyclic list), only printing
\def\listtmp{A,B,C,D,E,F}%
\ctgetvaluefromcycllist*{\listtmp}{10}\par\smallskip
%storing
\ctgetvaluefromcycllist{\listtmp}{-10}[\myelt]\myelt
```

D

C

```
%adding, only printing
\ctaddvalinlist*{values}{list}
%adding and storing (overwrite)
\ctaddvalinlist{values}{list}
%removing, only printing
\ctremovevalinlist*{value}{list}
%removing and storing (overwrite)
\ctremovevalinlist{value}{list}
```

```
%only printing
\ctaddvalinlist*{3}{1,2,5,6}\\
%defining and adding
\def\tmpaddlist{1,2,4,5,6}
\ctaddvalinlist{3}{\tmpaddlist}\tmpaddlist\\
%analysing
\readlist*\tmpADDlist{\tmpaddlist}
\showitems{\tmpADDlist}
```

1,2,5,6,3
1,2,4,5,6,3
$\boxed{1}\boxed{2}\boxed{4}\boxed{5}\boxed{6}\boxed{3}$

```
%only printing
\ctremovevalinlist*{3}{1,2,3,6,3,1,5,7,3}\\
%defining and removing within \mytmplist
\def\tmpremlist{1,2,3,6,3,1,5,7,3}
\ctremovevalinlist{3}{\tmpremlist}\mytmplist\\
%analysing
\readlist*\tmpREMlist{\mytmplist}
\showitems{\tmpREMlist}
```

1,2,6,1,5,7
1,2,6,1,5,7
$\boxed{1}\boxed{2}\boxed{6}\boxed{1}\boxed{5}\boxed{7}$

4.3 Reverse

```
%reversing, only printing
\ctreverselist*{list}
%reversing and storing (overwrite)
\ctreverselist{list}
```

```
%only printing
\ctreverselist*{14,10,15,11,9,10}\\
%reversing and storing
%storing
\ctreverselist{14,10,15,11,9,10}[\myreverse]\myreverse\\
%analysing
\readlist*\tmpREVERSElist{\myreverse}
\showitems{\tmpREVERSElist}
```

10,9,11,15,10,14
10,9,11,15,10,14
$\boxed{10}\boxed{9}\boxed{11}\boxed{15}\boxed{10}\boxed{14}$

4.4 Sub-list

```
%extracting a sub-list, only printing (* = start or end of list)
\ctsublist*{list}{begin}{end}
%extracting a sub-list and storing into \macro (\mysublist by default)
\ctsublist{list}{begin}{end}[\macro]
```

```
\def\mylist{10,20,30,40,50,60,70}
%indices 2 to 5, only printing
\ctsublist*{\mylist}{2}{5}\\
%from start to index 4, only printing
\ctsublist*{\mylist}{*}{4}\\
%from index 3 to end, only printing
\ctsublist*{\mylist}{3}{*}\\
%whole list (* on both sides), only printing
\ctsublist*{\mylist}{*}{*}
```

20,30,40,50
10,20,30,40
30,40,50,60,70
10,20,30,40,50,60,70

```
\def\mylist{10,20,30,40,50,60,70}
%storing sub-list [2,5] into \myresult
\ctsublist{\mylist}{2}{5}[\myresult]Sub-list [2,5]: \myresult\\
%storing sub-list [*,3] into \myresultb
\ctsublist{\mylist}{*}{3}[\myresultb]Sub-list [*,3]: \myresultb\\
%storing sub-list [5,*] into \myresultc
\ctsublist{\mylist}{5}{*}[\myresultc]Sub-list [5,*]: \myresultc
```

Sub-list [2,5]: 20,30,40,50
Sub-list [*,3]: 10,20,30
Sub-list [5,*]: 50,60,70

```
%slicing by formula (x variable), only printing
\ctslicelist*{list}{formula}
%slicing by formula and storing into \macro (\myslicelist by default)
\ctslicelist{list}{formula}[\macro]
```

```
\def\mylist{10,20,30,40,50,60,70,80,90,100}
%even indices (2*x), only printing
\ctslicelist*{\mylist}{2*x}\\
%odd indices (2*x-1), only printing
\ctslicelist*{\mylist}{2*x-1}\\
%formula 3*x+1, only printing
\ctslicelist*{\mylist}{3*x+1}\\
%square indices (x^2), only printing
\ctslicelist*{\mylist}{x^2}
```

20,40,60,80,100
10,30,50,70,90
40,70,100
10,40,90

```
\def\mylist{10,20,30,40,50,60,70,80,90,100}
%storing slice with 2*x into \myresult
\ctslicelist{\mylist}{2*x}[\myresult]Slice formula 2*x: \myresult\\
%storing slice with x^2 into \myresultb
\ctslicelist{\mylist}{x^2}[\myresultb]Slice formula x\^{}2: \myresultb
```

Slice formula 2*x: 20,40,60,80,100
Slice formula xˆ2: 10,40,90

4.5 Transform

```
%transforming a list through formula, only printing
\cttransformlist*[round]{list}{formula}
%transforming a list through formula and storing into \macro (\mytransformlist by default)
\cttransformlist*[round]{list}{formula}[\macro]
```

```
\def\mylist{10,20,30,40,50,60,70,80,90,100}
%transform with 3x+1, only printing
\cttransformlist*{\mylist}{3*x+1}\\
%transform with x^2, only printing
\cttransformlist*{\mylist}{x^2}\\
%formula sqrt(x), only printing
\cttransformlist*{10,20,30}{sqrt(x)}\\
%formula round(sqrt(x),3), only printing
\cttransformlist*[3]{\mylist}{sqrt(x)}\\
```

31,61,91,121,151,181,211,241,271,301
100,400,900,1600,2500,3600,4900,6400,8100,10000
3.1622776601683795,4.47213595499958,5.477225575051661
3.162,4.472,5.477,6.325,7.071,7.746,8.367,8.944,9.487,10

```
\def\mylist{10,20,30,40,50,60,70,80,90,100}
%storing transform with 2*x into \myresult
\cttransformlist{\mylist}{2*x}[\myresult]Transform formula 2*x:\\
\myresult\\
%storing transform with x^2-sqrt(x) into \myresultb
\cttransformlist[1]{\mylist}{x^2-sqrt(x)}[\myresultb]Transform formula x\^{}2-sqrt(x):\\
\myresultb
```

Transform formula 2*x:
20,40,60,80,100,120,140,160,180,200
Transform formula x^2-sqrt(x):
96.8,395.5,894.5,1593.7,2492.9,3592.3,4891.6,6391.1,8090.5,9990

4.6 Remove duplicate entries

```
%removing duplicates, only printing
\ctuniqlist*{list}
%removing duplicates with optional sorting [asc] or [des]
\ctuniqlist*[asc]{list}
\ctuniqlist*[des]{list}
%removing duplicates and storing into \macro (\myuniqlist by default)
\ctuniqlist{list}[\macro]
\ctuniqlist[asc]{list}[\macro]
\ctuniqlist[des]{list}[\macro]
```

```
\def\mylist{1,2,3,2,4,3,5,1}
%without sorting, only printing
\ctuniqlist*{\mylist}\\
%with ascending sort, only printing
\ctuniqlist*[asc]{\mylist}\\
%with descending sort, only printing
\ctuniqlist*[des]{\mylist}
```

1,2,3,4,5
1,2,3,4,5
5,4,3,2,1

```
\def\mylist{5,2,9,2,1,5,7,3,1}
%storing without sort into \myuniq
\ctuniqlist{\mylist}[\myuniq]No sort: \myuniq\\
%storing with [asc] into \myuniqasc
\ctuniqlist[asc]{\mylist}[\myuniqasc]With [asc]: \myuniqasc\\
%storing with [des] into \myuniqdes
\ctuniqlist[des]{\mylist}[\myuniqdes]With [des]: \myuniqdes
```

No sort: 5,2,9,1,7,3
With [asc]: 1,2,3,5,7,9
With [des]: 9,7,5,3,2,1

5 With two lists

5.1 Intersection

```
%intersection of two lists, only printing
\ctintersectlists*{list1}{list2}
%intersection with optional sorting [asc] or [des], only printing
\ctintersectlists*[asc]{list1}{list2}
\ctintersectlists*[des]{list1}{list2}
%intersection and storing into \macro (\myintersectlist by default)
\ctintersectlists{list1}{list2}[\macro]
\ctintersectlists[asc]{list1}{list2}[\macro]
\ctintersectlists[des]{list1}{list2}[\macro]
```

```
\def\mylistA{1,3,5,7,9,11,13}
\def\mylistB{3,6,9,12,15,7,1}
%without sorting, only printing
\ctintersectlists*{\mylistA}{\mylistB}\\
%with ascending sort, only printing
\ctintersectlists*[asc]{\mylistA}{\mylistB}\\
%with descending sort, only printing
\ctintersectlists*[des]{\mylistA}{\mylistB}
```

1,3,7,9
1,3,7,9
9,7,3,1

```
\def\mylistA{4,8,2,6,10,3,7}
\def\mylistB{2,5,8,11,6,3}
%storing without sort into \myinter
\ctintersectlists{\mylistA}{\mylistB}[\myinter]No sort: \myinter\\
%storing with [asc] into \myinterasc
\ctintersectlists[asc]{\mylistA}{\mylistB}[\myinterasc]With [asc]: \myinterasc\\
%storing with [des] into \myinterdes
\ctintersectlists[des]{\mylistA}{\mylistB}[\myinterdes]With [des]: \myinterdes
```

No sort: 8,2,6,3
With [asc]: 2,3,6,8
With [des]: 8,6,3,2

5.2 Union (merge)

```
%union of two lists (no duplicates), only printing
\ctmergelists*{list1}{list2}
%union with optional sorting [asc] or [des], only printing
\ctmergelists*[asc]{list1}{list2}
\ctmergelists*[des]{list1}{list2}
%union and storing into \macro (\mymergelist by default)
\ctmergelists{list1}{list2}[\macro]
\ctmergelists[asc]{list1}{list2}[\macro]
\ctmergelists[des]{list1}{list2}[\macro]
```

```
\def\mylistA{1,3,5,7,9}
\def\mylistB{3,6,9,12,15}
%without sorting, only printing
\ctmergelists*{\mylistA}{\mylistB}\\
%with ascending sort, only printing
\ctmergelists*[asc]{\mylistA}{\mylistB}\\
%with descending sort, only printing
\ctmergelists*[des]{\mylistA}{\mylistB}
```

1,3,5,7,9,6,12,15
1,3,5,6,7,9,12,15
15,12,9,7,6,5,3,1

```
\def\mylistA{4,8,2,6,10}
\def\mylistB{2,5,8,11,6}
%storing without sort into \mymerge
\ctmergelists{\mylistA}{\mylistB}[\mymerge]No sort: \mymerge\\
%storing with [asc] into \mymergeasc
\ctmergelists[asc]{\mylistA}{\mylistB}[\mymergeasc]With [asc]: \mymergeasc\\
%storing with [des] into \mymergedes
\ctmergelists[des]{\mylistA}{\mylistB}[\mymergedes]With [des]: \mymergedes
```

No sort: 4,8,2,6,10,5,11
With [asc]: 2,4,5,6,8,10,11
With [des]: 11,10,8,6,5,4,2

5.3 Difference

```
%difference A\B (elements in A absent from B), only printing
\ctdifflist*{listA}{listB}
%difference with optional sorting [asc] or [des], only printing
\ctdifflist*[asc]{listA}{listB}
\ctdifflist*[des]{listA}{listB}
%difference and storing into \macro (\mydifflist by default)
\ctdifflist{listA}{listB}[\macro]
\ctdifflist[asc]{listA}{listB}[\macro]
\ctdifflist[des]{listA}{listB}[\macro]
```

```
\def\mylistA{1,3,5,7,9,11,13}
\def\mylistB{3,7,11,15,19}
%without sorting, only printing
\ctdifflist*{\mylistA}{\mylistB}\\
%with ascending sort, only printing
\ctdifflist*[asc]{\mylistA}{\mylistB}\\
%with descending sort, only printing
\ctdifflist*[des]{\mylistA}{\mylistB}
```

1,5,9,13
1,5,9,13
13,9,5,1

```
\def\mylistA{4,8,2,6,10,3,7}
\def\mylistB{2,5,8,11,6,3}
%storing without sort into \mydiff
\ctdifflist{\mylistA}{\mylistB}[\mydiff]No sort: \mydiff\\
%storing with [asc] into \mydiffasc
\ctdifflist[asc]{\mylistA}{\mylistB}[\mydiffasc]With [asc]: \mydiffasc\\
%storing with [des] into \mydiffdes
\ctdifflist[des]{\mylistA}{\mylistB}[\mydiffdes]With [des]: \mydiffdes
```

No sort: 4,10,7
With [asc]: 4,7,10
With [des]: 10,7,4

5.4 Symmetric difference

```
%symmetric difference (elements in A or B but not both), only printing
\ctsymdifflist*{listA}{listB}
%symmetric difference with optional sorting [asc] or [des], only printing
\ctsymdifflist*[asc]{listA}{listB}
\ctsymdifflist*[des]{listA}{listB}
%symmetric difference and storing into \macro (\mysymdifflist by default)
\ctsymdifflist{listA}{listB}[\macro]
\ctsymdifflist[asc]{listA}{listB}[\macro]
\ctsymdifflist[des]{listA}{listB}[\macro]
```

```
\def\mylistA{1,3,5,7,9,11,13}
\def\mylistB{3,7,11,15,19}
%without sorting, only printing
\ctsymdifflist*{\mylistA}{\mylistB}\\
%with ascending sort, only printing
\ctsymdifflist*[asc]{\mylistA}{\mylistB}\\
%with descending sort, only printing
\ctsymdifflist*[des]{\mylistA}{\mylistB}
```

1,5,9,13,15,19
1,5,9,13,15,19
19,15,13,9,5,1

```
\def\mylistA{4,8,2,6,10,3,7}
\def\mylistB{2,5,8,11,6,3}
%storing without sort into \mysymdiff
\ctsymdifflist{\mylistA}{\mylistB}[\mysymdiff]No sort: \mysymdiff\\
%storing with [asc] into \mysymdiffasc
\ctsymdifflist[asc]{\mylistA}{\mylistB}[\mysymdiffasc]With [asc]: \mysymdiffasc\\
%storing with [des] into \mysymdiffdes
\ctsymdifflist[des]{\mylistA}{\mylistB}[\mysymdiffdes]With [des]: \mysymdiffdes
```

No sort: 4,10,7,5,11
With [asc]: 4,5,7,10,11
With [des]: 11,10,7,5,4

6 Macros with testing

6.1 Value in list ?

```
%testing if value is in list, with boolean result in \macro (\resisinlist by default)
\ctboolvalinlist{value}{list}[\macro]
%conditionnal test if value is in list, according to xint syntax
\cttestifvalinlist{3}{0,1,2,3}{true}{false}
```

```
%test with xint syntax
\cttestifvalinlist{-1}{0,1,2,3}{true}{false}\\
%test with xint syntax
\cttestifvalinlist{3}{0,1,2,3}{true}{false}\\
%boolean macro
\def\myteslist{0,5,10,5,6,9,7,8}
\ctboolvalinlist{5}{\myteslist}[\resisinlist]\resisinlist
```

false
true
1

6.2 Count value

```
%counting value, only printing
\ctcountvalinlist*{value}{list}
%counting value, with result in \macro (\rescount by default)
\ctcountvalinlist{value}{list}[\macro]
```

```
%only printing
\ctcountvalinlist*{8}{1,2,3,4,5,6,6,7,8,8,8,9}\\
%storing
\def\tmpcountlist{1,2,3,4,5,6,6,7,8,8,8,9}
\ctcountvalinlist{6}{\tmpcountlist}[\rescountsix]\rescountsix\\
\ctcountvalinlist{10}{\tmpcountlist}[\rescountten]\rescountten
```

3
2
0

6.3 Get index

```
%index value (first occurency), only printing
\ctgetindexfromlist*{value}{list}
%index value (first occurency), with result in \macro (\resmyindex by default)
\ctgetindexfromlist{value}{list}[\macro]
```

```
%only printing
\ctgetindexfromlist*{8}{1,2,3,4,5,6,6,7,8,8,8,9}\\
%storing
\def\tmpindexlist{1,2,3,4,5,6,6,7,8,8,8,9}
\ctgetindexfromlist{6}{\tmpindexlist}[\resindexsix]\resindexsix\\
\ctgetindexfromlist{10}{\tmpindexlist}[\resindexten]\resindexten
```

9
6
0

7 History

```
0.20b: Extract element with cyclic version of list
0.20a: Sub-list extraction + transform list + merge/intersection + improvements
0.1.9: Improvements with LaTeX3, tks to ankaa3908
0.1.8: Get index from value
0.1.7: Improvements with LaTeX3
0.1.6: Initial version,code written in LaTeX3
```

8 The code

```
% Author     : C. Pierquet
% licence    : Released under the LaTeX Project Public License v1.3c or later, see
     http://www.latex-project.org/lppl.txt

\NeedsTeXFormat{LaTeX2e}
\ProvidesExplPackage{commalists-tools-l3}{2026-03-24}{0.20b}{Basic operations for numeral comma separated lists}

%------History
% 0.20b  Extract element with cyclic version of list
% 0.20a  New commands (sublist / transform / slice / merge / intersection / ...)
% 0.1.9  Improvements with latex3 (tks to ankaa3908)
% 0.1.8  Get index
% 0.1.7  Improvements with latex3
% 0.1.6  Initial version (prefixed macros due to legacy package)

%------Variables
\clist_new:N \l__commalists_var_clist
\clist_new:N \l__commalists_varb_clist
\int_new:N   \l__commalists_tmpc_int
\fp_new:N    \l__commalists_tmpa_fp

%------Show list
\NewDocumentCommand\ctshowlist{ O { , } m }
% #1 = sep
% #2 = list
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \clist_use:Nn \l__commalists_var_clist { #1 }
}

%------Sort list, reverse list
\NewDocumentCommand\ctsortasclist{ s m O { \mysortedlist } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \mysortedlist)
{
  \clist_set:Ne \l__commalists_var_clist {#2}
  \clist_sort:Nn \l__commalists_var_clist {
    \fp_compare:nNnTF { ##1 } > { ##2 }
      { \sort_return_swapped: }
      { \sort_return_same: }
  }
  \IfBooleanTF { #1 } %if star := just printing // if not, storing
    {
      \clist_use:Nn \l__commalists_var_clist { , }
    }
    {
      \tl_gset:Ne \g_tmpa_tl { \l__commalists_var_clist }
      \tl_gset:NV #3 \g_tmpa_tl
    }
}

\NewDocumentCommand\ctsortdeslist{ s m O { \mysortedlist } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \mysortedlist)
```

```
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \clist_sort:Nn \l__commalists_var_clist {
    \fp_compare:nNnTF { ##1 } < { ##2 }
      { \sort_return_swapped: }
      { \sort_return_same: }
  }
  \IfBooleanTF { #1 } %if star := just printing // if not, storing
    {
      \clist_use:Nn \l__commalists_var_clist { , }
    }
    {
      \tl_gset:Ne \g_tmpa_tl { \l__commalists_var_clist }
      \tl_gset:NV #3 \g_tmpa_tl
    }
}

\NewDocumentCommand\ctreverselist{ s m O { \reverselist } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \reverselist)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \IfBooleanTF { #1 } %if star := just printing // if not, storing
  {
    \clist_reverse:N \l__commalists_var_clist
    \clist_use:Nn \l__commalists_var_clist { , }
  }
  {
    \clist_set_eq:NN #3 \l__commalists_var_clist
    \clist_greverse:N #3
  }
}

%------Test in list
\NewDocumentCommand\ctboolvalinlist{ m m O { \resisinlist } }
% #1 = element to test
% #2 = list
% #3 = output macro (non-starred, default \resisinlist)
{
  \clist_set:Ne \l__commalists_var_clist {#2}

  \tl_gset:Nn #3 { 0 }

  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \tl_if_eq:neT { ##1 } { #1 }
    {
      \tl_gset:Nn #3 { 1 }
      \clist_map_break:
    }
  }
}

\NewDocumentCommand\cttestifvalinlist{ m m m m }
% #1 = element to test
% #2 = list
% #3 = true branch
% #4 = false branch
{
  \clist_set:Ne \l__commalists_var_clist { #2 }

  \bool_set_false:N \l_tmpa_bool

  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \tl_if_eq:neT { ##1 } { #1 }
    {
      \bool_set_true:N \l_tmpa_bool
      \clist_map_break:
    }
  }
  \bool_if:NTF \l_tmpa_bool { #3 } { #4 }
}

%------Add, remove
\NewDocumentCommand\ctaddvalinlist{ s m m }
% #1 = star (just printing)
% #2 = value to add
% #3 = list
{
  \IfBooleanTF { #1 }
  {
    #3, #2
  }
  {
    \tl_gset:Ne #3 { \tl_use:N #3 , #2 }
  }
}

\NewDocumentCommand\ctremovevalinlist{ s m m O { \mytmplist } }
% #1 = star (just printing)
% #2 = value to remove
% #3 = list
% #4 = output macro (non-starred, default \mytmplist)
{
  \clist_set:Ne \l__commalists_var_clist { #3 }

  \clist_clear_new:N \l_tmpb_clist

  \clist_map_inline:Nn \l__commalists_var_clist
  {

    \tl_if_eq:neF { ##1 } { #2 }
    {
      \clist_put_right:Nn \l_tmpb_clist { ##1 }
    }
  }

  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l_tmpb_clist { , }
  }
  {
    \tl_gset:Ne \g_tmpa_tl { \clist_use:Nn \l_tmpb_clist { , } }
    \tl_gset:NV #4 \g_tmpa_tl
  }
}

%------Count
\NewDocumentCommand\ctcountvalinlist{ s m m O { \rescount } }
% #1 = star (just printing)
% #2 = value to count
% #3 = list
% #4 = output macro (non-starred, default \rescount)
{
  \clist_set:Ne \l__commalists_var_clist { #3 }

  \int_zero:N \l_tmpa_int
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \tl_if_eq:neT { ##1 } { #2 }
    {
      \int_incr:N \l_tmpa_int
    }
  }
  \IfBooleanTF { #1 }
  {
    \int_use:N \l_tmpa_int
  }
  {
    \tl_gset:NV #4 \l_tmpa_int
  }
}
```

```
%------Calculus
\NewDocumentCommand\ctminoflist{ s m O { \resmin } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \resmin)
{
  \IfBooleanTF { #1 }
  {
    \fp_eval:n { min(#2) }
  }
  {
    \tl_gset:Ne #3 { \fp_eval:n { min(#2) } }
  }
}

\NewDocumentCommand\ctmaxoflist{ s m O { \resmax } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \resmax)
{
  \IfBooleanTF { #1 }
  {
    \fp_eval:n { max(#2) }
  }
  {
    \tl_gset:Ne #3 { \fp_eval:n { max(#2) } }
  }
}

\NewDocumentCommand\ctsumoflist{ s m O { \ressum } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \ressum)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \fp_set:Nn \l__commalists_tmpa_fp { 0 }
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \fp_set:Nn \l__commalists_tmpa_fp
      { \fp_eval:n { \l__commalists_tmpa_fp + (##1) } }
  }
  \IfBooleanTF { #1 }
  {
    \fp_use:N \l__commalists_tmpa_fp
  }
  {
    \tl_gset:Ne #3 { \fp_use:N \l__commalists_tmpa_fp }
  }
}

\NewDocumentCommand\ctprodoflist{ s m O { \resprod } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \resprod)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \fp_set:Nn \l__commalists_tmpa_fp { 1 }
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \fp_set:Nn \l__commalists_tmpa_fp
      { \fp_eval:n { \l__commalists_tmpa_fp * (##1) } }
  }
  \IfBooleanTF { #1 }
  {
    \fp_use:N \l__commalists_tmpa_fp
  }
  {
    \tl_gset:Ne #3 { \fp_use:N \l__commalists_tmpa_fp }
  }
}
```

```
\NewDocumentCommand\ctmeanoflist{ s m O { \resmean } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \resmean)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \fp_set:Nn \l__commalists_tmpa_fp { 0 }
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \fp_set:Nn \l__commalists_tmpa_fp
      { \fp_eval:n { \l__commalists_tmpa_fp + (##1) } }
  }
  \fp_set:Nn \l__commalists_tmpa_fp
    {
      \fp_eval:n {
        ( \l__commalists_tmpa_fp )
        /
        ( \clist_count:N \l__commalists_var_clist )
      }
    }
  \IfBooleanTF { #1 }
  {
    \fp_use:N \l__commalists_tmpa_fp
  }
  {
    \tl_gset:Ne #3 { \fp_use:N \l__commalists_tmpa_fp }
  }
}

\NewDocumentCommand\ctlenoflist{ s m O { \resmylen } }
% #1 = star (just printing)
% #2 = list
% #3 = output macro (non-starred, default \myreslen)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \IfBooleanTF { #1 }
  {
    \clist_count:N \l__commalists_var_clist
  }
  {
    \tl_gset:Ne #3 { \clist_count:N \l__commalists_var_clist }
  }
}

%------Get, index
\NewDocumentCommand\ctgetvaluefromlist{ s m m O { \resmyelt } }
% #1 = star (just printing)
% #2 = list
% #3 = index
% #3 = output macro (non-starred, default \resmyelt)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \IfBooleanTF { #1 }
  {
    \clist_item:Nn \l__commalists_var_clist { #3 }
  }
  {
    \tl_gset:Ne #4 { \clist_item:Nn \l__commalists_var_clist { #3 } }
  }
}

\NewDocumentCommand\ctgetvaluefromcycllist{ s m m O { \resmyelt } }
  % #1 = star (just printing)
  % #2 = list
  % #3 = index
  % #4 = output macro (default \resmyelt)
{
    \clist_set:Ne \l__commalists_var_clist { #2 }
    \int_set:Nn \l_tmpa_int { \clist_count:N \l__commalists_var_clist }
    %\int_set:Ne \l_tmpb_int { #3 }
    \int_set:Nn \l_tmpb_int { \int_eval:n { #3 } }
    % test indice 0
    \int_compare:nNnTF { \l_tmpb_int } = { 0 }
      {
        \msg_warning:nn { commalist-tools } { index-zero }
        \tl_gset:Nn #4 { }
      }
      {
        % modulo positif
        \int_compare:nNnTF { \l_tmpb_int } > { 0 }
          { \int_set:Nn \l_tmpb_int { \l_tmpb_int - 1 } }
          { }
        \int_set:Nn \l__commalists_tmpc_int
          { \int_mod:nn { \l_tmpb_int } { \l_tmpa_int } }
        \int_compare:nNnT { \l__commalists_tmpc_int } < { 0 }
          { \int_add:Nn \l__commalists_tmpc_int { \l_tmpa_int } }
        \int_add:Nn \l__commalists_tmpc_int { 1 }
        % récupération
        \IfBooleanTF { #1 }
          {
            \clist_item:Nn \l__commalists_var_clist { \l__commalists_tmpc_int }
          }
          {
            \tl_gset:Ne #4
              {
                \clist_item:Nn
                  \l__commalists_var_clist
                  { \l__commalists_tmpc_int }
              }
          }
      }
}
% message d'erreur
\msg_new:nnn { commalist-tools } { index-zero }
  { ctgetvaluefromcycllist:~index~0~not~available }

\NewDocumentCommand\ctgetindexfromlist{ s m m O { \resmyindex } }
% #1 = star (just printing)
% #2 = list
% #3 = element
% #3 = output macro (non-starred, default \resmyindex)
{
  \IfBooleanTF { #1 }
  {
    \ct_get_index:nn { #3 } { #2 }
  }
  {
    \ct_get_index:nnN { #3 } { #2 } #4
  }
}

\cs_new:Npn \ct_get_index:nn #1 #2
{
  \int_zero:N \l_tmpa_int
  \bool_set_false:N \l_tmpa_bool
  \tl_set:Ne \l_tmpb_tl { #2 }
  \exp_args:No \clist_map_inline:nn { #1 }
  {
    \int_incr:N \l_tmpa_int
    \str_if_eq:nVT { ##1 } \l_tmpb_tl
    {
      \bool_set_true:N \l_tmpa_bool
      \clist_map_break:
    }
  }
  \bool_if:NTF \l_tmpa_bool
  { \int_use:N \l_tmpa_int }
  { 0 }
}

\cs_new:Npn \ct_get_index:nnN #1 #2 #3
{
  \int_zero:N \l_tmpa_int
  \bool_set_false:N \l_tmpa_bool
  \tl_set:Ne \l_tmpb_tl { #2 }
  \exp_args:No \clist_map_inline:nn { #1 }
  {
    \int_incr:N \l_tmpa_int
    \str_if_eq:nVT { ##1 } \l_tmpb_tl
    {
      \bool_set_true:N \l_tmpa_bool
      \clist_map_break:
    }
  }
  \bool_if:NTF \l_tmpa_bool
  { \tl_gset:Ne #3 { \int_use:N \l_tmpa_int } }
  { \tl_gset:Nn #3 { 0 } }
}

%------Sub-list
\clist_new:N \l__commalists_sub_clist
\int_new:N \l__commalists_begin_int
\int_new:N \l__commalists_end_int
\int_new:N \l__commalists_idx_int

\NewDocumentCommand\ctsublist{ s m m m O { \mysublist } }
% #1 = star (display only)
% #2 = list
% #3 = begin index (* = start from first)
% #4 = end   index (* = go to last)
% #5 = output macro (non-starred only, default \mysublist)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \clist_clear:N \l__commalists_sub_clist

  % --- resolve begin index
  \tl_if_eq:nnTF { #3 } { * }
  { \int_set:Nn \l__commalists_begin_int { 1 } }
  { \int_set:Nn \l__commalists_begin_int { #3 } }

  % --- resolve end index
  \tl_if_eq:nnTF { #4 } { * }
  { \int_set:Nn \l__commalists_end_int
    { \clist_count:N \l__commalists_var_clist }
  }
  { \int_set:Nn \l__commalists_end_int { #4 } }

  % --- iterate and collect items in [begin, end]
  \int_set:Nn \l__commalists_idx_int { 0 }
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \int_incr:N \l__commalists_idx_int
    \int_compare:nTF
    { \l__commalists_idx_int >= \l__commalists_begin_int }
    {
       \int_compare:nTF
        { \l__commalists_idx_int <= \l__commalists_end_int }
        {
          \clist_put_right:Nn \l__commalists_sub_clist { ##1 }
        }
        { \clist_map_break: }  % early exit : idx > end, stop
    }
    { }
  }

  % --- output
  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l__commalists_sub_clist { , }
  }
  {
    \tl_gset:Ne #5 { \clist_use:Nn \l__commalists_sub_clist { , } }
  }
}

%------Slice-list
```

```
\clist_new:N \l__commalists_slice_clist
\int_new:N \l__commalists_slice_x_int
\int_new:N \l__commalists_slice_idx_int
\int_new:N \l__commalists_slice_len_int
\tl_new:N \l__commalists_slice_formula_tl

\NewDocumentCommand\ctslicelist{ s m m O { \myslicelist } }
% #1 = star (display only)
% #2 = list
% #3 = formula in x (e.g. 2*x, 3*x+1, x^2)
% #4 = output macro (non-starred only, default \myslicelist)
{
  \clist_set:Ne \l__commalists_var_clist { #2 }
  \clist_clear:N \l__commalists_slice_clist
  \int_set:Nn \l__commalists_slice_len_int
    { \clist_count:N \l__commalists_var_clist }
  \int_set:Nn \l__commalists_slice_x_int { 1 }

  % --- iterate x = 1..len, evaluate formula, collect item if index in range
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    % substitute x by current value in formula, then evaluate
    \tl_set:Nn \l__commalists_slice_formula_tl { #3 }
    \tl_replace_all:Nne \l__commalists_slice_formula_tl { x }
      { \int_use:N \l__commalists_slice_x_int }
    \int_set:Nn \l__commalists_slice_idx_int
      { \fp_eval:n { \l__commalists_slice_formula_tl } }

    % stop as soon as the computed index goes out of range
    \int_compare:nTF
      { \int_use:N \l__commalists_slice_idx_int >= 1 }
    {
      \int_compare:nTF
        { \int_use:N \l__commalists_slice_idx_int
          <=
          \int_use:N \l__commalists_slice_len_int }
      {
        \clist_put_right:Ne \l__commalists_slice_clist
          { \clist_item:Nn \l__commalists_var_clist
            { \l__commalists_slice_idx_int }
          }
        \int_incr:N \l__commalists_slice_x_int
      }
      {
        \clist_map_break:
      }
    }
    {
      \clist_map_break:
    }
  }

  % --- output
  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l__commalists_slice_clist { , }
  }
  {
    \tl_gset:Ne #4 { \clist_use:Nn \l__commalists_slice_clist { , } }
  }
}

%------Transform-list
\clist_new:N \l__commalists_transform_clist
\tl_new:N \l__commalists_transform_formula_tl

\NewDocumentCommand\cttransformlist{ s o m m O { \mytransformlist } }
% #1 = star (display only)
% #2 = round
% #3 = list
% #4 = formula in x (e.g. 2*x+1, x^2, sqrt(x))
% #5 = output macro (non-starred only, default \mytransformlist)
```

```
{
  \clist_set:Ne \l__commalists_var_clist { #3 }
  \clist_clear:N \l__commalists_transform_clist
  % --- iterate over each element, apply formula
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    % --- substitute x by current element value in formula
    \tl_set:Nn \l__commalists_transform_formula_tl { #4 }
    \tl_replace_all:Nnn \l__commalists_transform_formula_tl { x } { ##1 }

    % --- evaluate and collect result
    \IfNoValueTF { #2 }
    {
      \clist_put_right:Ne \l__commalists_transform_clist
      { \fp_eval:n { \l__commalists_transform_formula_tl } }
    }
    {
      \clist_put_right:Ne \l__commalists_transform_clist
      { \fp_eval:n { round( \l__commalists_transform_formula_tl , #2 ) } }
    }
  }

  % --- output
  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l__commalists_transform_clist { , }
  }
  {
    \tl_gset:Ne #5 { \clist_use:Nn \l__commalists_transform_clist { , } }
  }
}

%------Unique list
\clist_new:N \l__commalists_uniq_clist
\bool_new:N \l__commalists_uniq_asc_bool
\bool_new:N \l__commalists_uniq_des_bool

\keys_define:nn { commalists / uniq }
{
  asc .bool_set:N   = \l__commalists_uniq_asc_bool,
  asc .default:n    = true,
  des .bool_set:N   = \l__commalists_uniq_des_bool,
  des .default:n    = true,
}

\NewDocumentCommand\ctuniqlist{ s O { } m O { \myuniqlist } }
% #1 = star (display only)
% #2 = options (asc, des)
% #3 = list
% #4 = output macro (non-starred only, default \myuniqlist)
{
  % reset booleans
  \bool_set_false:N \l__commalists_uniq_asc_bool
  \bool_set_false:N \l__commalists_uniq_des_bool

  % parse keys
  \keys_set:nn { commalists / uniq } { #2 }

  \clist_set:Ne \l__commalists_var_clist { #3 }
  \clist_clear:N \l__commalists_uniq_clist

  % --- iterate and collect unique elements
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \clist_if_in:NnF \l__commalists_uniq_clist { ##1 }
    {
      \clist_put_right:Nn \l__commalists_uniq_clist { ##1 }
    }
  }

  % --- sort if requested
  \bool_if:NT \l__commalists_uniq_asc_bool
  {
    \clist_sort:Nn \l__commalists_uniq_clist
    {
      \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }
  \bool_if:NT \l__commalists_uniq_des_bool
  {
    \clist_sort:Nn \l__commalists_uniq_clist
    {
      \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }

  % --- output
  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l__commalists_uniq_clist { , }
  }
  {
    \tl_gset:Ne #4 { \clist_use:Nn \l__commalists_uniq_clist { , } }
  }
}

%------Intersection
\clist_new:N \l__commalists_intersect_clist
\bool_new:N \l__commalists_intersect_asc_bool
\bool_new:N \l__commalists_intersect_des_bool

\keys_define:nn { commalists / intersect }
{
  asc .bool_set:N   = \l__commalists_intersect_asc_bool,
  asc .default:n    = true,
  des .bool_set:N   = \l__commalists_intersect_des_bool,
  des .default:n    = true,
}

\NewDocumentCommand\ctintersectlists{ s O { } m m O { \myintersectlist } }
% #1 = star (just printing)
% #2 = options (asc, des)
% #3 = list 1
% #4 = list 2
% #5 = output macro (non-starred, default \myintersectlist)
{
  % set lists
  \clist_set:Ne \l__commalists_var_clist { #3 }
  \clist_set:Ne \l__commalists_varb_clist { #4 }

  % reset booleans
  \bool_set_false:N \l__commalists_intersect_asc_bool
  \bool_set_false:N \l__commalists_intersect_des_bool

  % parse keys
  \keys_set:nn { commalists / intersect } { #2 }

  % clear res list
  \clist_clear:N \l__commalists_intersect_clist

  % --- iterate and collect common elements
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \clist_if_in:NnT \l__commalists_varb_clist { ##1 }
    {
      % if ok, add
      \clist_if_in:NnF \l__commalists_intersect_clist { ##1 }
      {
        \clist_put_right:Nn \l__commalists_intersect_clist { ##1 }
      }
    }
  }

  % --- sort if asked
  \bool_if:NT \l__commalists_intersect_asc_bool
  {
    \clist_sort:Nn \l__commalists_intersect_clist
    {
      \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }
  \bool_if:NT \l__commalists_intersect_des_bool
  {
    \clist_sort:Nn \l__commalists_intersect_clist
    {
      \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }

  % --- output
  \IfBooleanTF { #1 }
  {
    \clist_if_empty:NTF \l__commalists_intersect_clist
    { }
    { \clist_use:Nn \l__commalists_intersect_clist { , } }
  }
  {
    \tl_gset:Ne #5 { \clist_use:Nn \l__commalists_intersect_clist { , } }
  }
}

%------Union
\clist_new:N \l__commalists_merge_clist
\bool_new:N \l__commalists_merge_asc_bool
\bool_new:N \l__commalists_merge_des_bool

\keys_define:nn { commalists / merge }
{
  asc .bool_set:N   = \l__commalists_merge_asc_bool,
  asc .default:n    = true,
  des .bool_set:N   = \l__commalists_merge_des_bool,
  des .default:n    = true,
}

\NewDocumentCommand\ctmergelists{ s O { } m m O { \mymergelist } }
% #1 = star (just printing)
% #2 = options (asc, des)
% #3 = list 1
% #4 = list 2
% #5 = output macro (non-starred, default \mymergelist)
{
  % reinit of booleans
  \bool_set_false:N \l__commalists_merge_asc_bool
  \bool_set_false:N \l__commalists_merge_des_bool

  % keys reading
  \keys_set:nn { commalists / merge } { #2 }

  % set lists
  \clist_set:Ne \l__commalists_var_clist { #3 }
  \clist_set:Ne \l__commalists_varb_clist { #4 }

  % init merge list
  \clist_clear:N \l__commalists_merge_clist

  % add element list1, w/o doublons
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \clist_if_in:NnF \l__commalists_merge_clist { ##1 }
    {
      \clist_put_right:Nn \l__commalists_merge_clist { ##1 }
    }
  }

  % add element list2, w/o doublons
  \clist_map_inline:Nn \l__commalists_varb_clist
  {
    \clist_if_in:NnF \l__commalists_merge_clist { ##1 }
    {
      \clist_put_right:Nn \l__commalists_merge_clist { ##1 }
    }
  }

  % sort if asked
  \bool_if:NT \l__commalists_merge_asc_bool
  {
    \clist_sort:Nn \l__commalists_merge_clist
    {
      \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }
  \bool_if:NT \l__commalists_merge_des_bool
  {
    \clist_sort:Nn \l__commalists_merge_clist
    {
      \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }

  % --- Output
  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l__commalists_merge_clist { , }
  }
  {
    \tl_gset:Ne #5 { \clist_use:Nn \l__commalists_merge_clist { , } }
  }
}

%------Difference
\clist_new:N \l__commalists_diff_clist
\bool_new:N \l__commalists_diff_asc_bool
\bool_new:N \l__commalists_diff_des_bool

\keys_define:nn { commalists / diff }
{
  asc .bool_set:N   = \l__commalists_diff_asc_bool,
  asc .default:n    = true,
  des .bool_set:N   = \l__commalists_diff_des_bool,
  des .default:n    = true,
}

\NewDocumentCommand\ctdifflist{ s O { } m m O { \mydifflist } }
% #1 = star (display only)
% #2 = options (asc, des)
% #3 = list A
% #4 = list B
% #5 = output macro (non-starred, default \mydifflist)
{
  % set lists
  \clist_set:Ne \l__commalists_var_clist  { #3 }
  \clist_set:Ne \l__commalists_varb_clist { #4 }

  % reset booleans
  \bool_set_false:N \l__commalists_diff_asc_bool
  \bool_set_false:N \l__commalists_diff_des_bool
  % parse keys
  \keys_set:nn { commalists / diff } { #2 }

  % clear result list
  \clist_clear:N \l__commalists_diff_clist

  % --- iterate A, keep elements absent from B
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \clist_if_in:NnF \l__commalists_varb_clist { ##1 }
    {
      % avoid duplicates in result
      \clist_if_in:NnF \l__commalists_diff_clist { ##1 }
      {
        \clist_put_right:Nn \l__commalists_diff_clist { ##1 }
      }
    }
  }

  % --- sort if asked
  \bool_if:NT \l__commalists_diff_asc_bool
  {
    \clist_sort:Nn \l__commalists_diff_clist
    {
      \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }
  \bool_if:NT \l__commalists_diff_des_bool
  {
    \clist_sort:Nn \l__commalists_diff_clist
    {
      \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }

  % --- output
  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l__commalists_diff_clist { , }
  }
  {
    \tl_gset:Ne #5 { \clist_use:Nn \l__commalists_diff_clist { , } }
  }
}

%------Symmetric difference
\clist_new:N \l__commalists_symdiff_clist
\bool_new:N \l__commalists_symdiff_asc_bool
\bool_new:N \l__commalists_symdiff_des_bool

\keys_define:nn { commalists / symdiff }
{
  asc .bool_set:N   = \l__commalists_symdiff_asc_bool,
  asc .default:n    = true,
  des .bool_set:N   = \l__commalists_symdiff_des_bool,
  des .default:n    = true,
}

\NewDocumentCommand\ctsymdifflist{ s O { } m m O { \mysymdifflist } }
% #1 = star (display only)
% #2 = options (asc, des)
% #3 = list A
% #4 = list B
% #5 = output macro (non-starred, default \mysymdifflist)
{
  % set lists
  \clist_set:Ne \l__commalists_var_clist  { #3 }
  \clist_set:Ne \l__commalists_varb_clist { #4 }

  % reset booleans
  \bool_set_false:N \l__commalists_symdiff_asc_bool
  \bool_set_false:N \l__commalists_symdiff_des_bool

  % parse keys
  \keys_set:nn { commalists / symdiff } { #2 }

  % clear result list
  \clist_clear:N \l__commalists_symdiff_clist

  % --- elements in A absent from B
  \clist_map_inline:Nn \l__commalists_var_clist
  {
    \clist_if_in:NnF \l__commalists_varb_clist { ##1 }
    {
      \clist_if_in:NnF \l__commalists_symdiff_clist { ##1 }
      {
        \clist_put_right:Nn \l__commalists_symdiff_clist { ##1 }
      }
    }
  }

  % --- elements in B absent from A
  \clist_map_inline:Nn \l__commalists_varb_clist
  {
    \clist_if_in:NnF \l__commalists_var_clist { ##1 }
    {
      \clist_if_in:NnF \l__commalists_symdiff_clist { ##1 }
      {
        \clist_put_right:Nn \l__commalists_symdiff_clist { ##1 }
      }
    }
  }

  % --- sort if asked
  \bool_if:NT \l__commalists_symdiff_asc_bool
  {
    \clist_sort:Nn \l__commalists_symdiff_clist
    {
      \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }
  \bool_if:NT \l__commalists_symdiff_des_bool
  {
    \clist_sort:Nn \l__commalists_symdiff_clist
    {
      \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
    }
  }

  % --- output
  \IfBooleanTF { #1 }
  {
    \clist_use:Nn \l__commalists_symdiff_clist { , }
  }
  {
    \tl_gset:Ne #5 { \clist_use:Nn \l__commalists_symdiff_clist { , } }
  }
}

\endinput
```