

commalists-tools-l3

L^AT_EX3

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

Version 0.20e - 2026/07/23

Cédric Pierquet

c pierquet - at - outlook . fr

<https://github.com/cpierquet/latex-packages/tree/main/commalists-tools>

Thanks to Valentin Dao for his comments and improvements to the L^AT_EX3 code.

```
\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	3
2	Global usage	3
2.1	Tools	3
2.2	The macro for printing	3
3	The macros for calculating	4
3.1	Length, minimum, maximum	4
3.2	Mean, sum, prod	4
4	Macros for manipulating	5
4.1	Sorting	5
4.2	Extract element, add/remove element	6
4.3	Reverse	7
4.4	Sub-list	7
4.5	Transform	8
4.6	Remove duplicate entries	9
4.7	Shuffle	10
5	With two lists	10
5.1	Intersection	10
5.2	Union (merge)	11
5.3	Difference	12
5.4	Symmetric difference	13
6	Macros with testing	13
6.1	Value in list ?	13
6.2	Count value	14
6.3	Get index	14
7	History	15
8	The code	15

1 Introduction

In order to load `commalists-tools-13`, simply use:

```
\usepackage{commalists-tools-13}
```

All code is written in $\text{\LaTeX}$ 3, 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!

🔗 Since version 0.20e, the syntax is different for optional arguments that are *not* the separator: rounding precision (`\cttransformlist`) and sorting keys `asc/des` (`\ctuniqlist`, `\ctintersectlists`, `\ctmergelists`, `\ctdifflist`, `\ctsyndifflist`) are no longer given between square brackets `[...]`, but between angle brackets `<...>`. The list separator itself stays between square brackets `[...]`, always in first position (right after the star, if any).

2 Global usage

2.1 Tools

Package `commalists-tools-13` 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 (in & out)
\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
```

```
%just printing, with optional separator in & out
\ctshowlistwithsep[in sep]{list}{out sep}
```

```
\def\mytmplist{12 $ -4$ 5$7 $8$ 9$ 0}
\ctshowlistwithsep[$]{\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

9	10	10	11.5	14	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

15	14	11.5	10	10	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

1	2	4	5	6	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
126157
```

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
10911151014
```

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

```

```

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

```

```

\def\mylist{10,20,30,40,50,60,70,80,90,100}
%even indices (2*x), only printing
\ctslitelist*{\mylist}{2*x}\\
%odd indices (2*x-1), only printing
\ctslitelist*{\mylist}{2*x-1}\\
%formula 3*x+1, only printing
\ctslitelist*{\mylist}{3*x+1}\\
%square indices (x^2), only printing
\ctslitelist*{\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
\ctslitelist{\mylist}{2*x}[\myresult]Slice formula 2*x: \myresult\\
%storing slice with x^2 into \myresultb
\ctslitelist{\mylist}{x^2}[\myresultb]Slice formula x^{2}: \myresultb

```

```

Slice formula 2*x: 20,40,60,80,100
Slice formula x^2: 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
3.162277660168379,4.472135954999579,5.477225575051661
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

```

4.7 Shuffle

```

%shuffle list, only printing
\ctshufflelist*{list}
%shuffle list and storing into \macro (\myshuffledlist by default)
\ctshufflelist{list}[\macro]

```

```

\def\mylist{1,2,3,2,4,3,5,1}
%shuffle, only printing
\ctshufflelist*{\mylist}

```

```

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

```

```

\def\mylist{5,2,9,2,1,5,7,3,1}
%storing into \mytestshuffle
\ctshufflelist{\mylist}[\mytestshuffle]\mytestshuffle

```

```

2,1,1,5,9,7,5,3,2

```

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

```

```

3,9,7,1
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}\myinterNo sort: \myinter\\
%storing with <asc> into \myinterasc
\ctintersectlists<asc>\mylistA\mylistB\myinterascWith <asc>: \myinterasc\\
%storing with <des> into \myinterdes
\ctintersectlists<des>\mylistA\mylistB\myinterdesWith <des>: \myinterdes

```

```

No sort: 2,8,6,3
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

0.20e: Improvements with cyclic version
0.20d: Improvements with $\LaTeX$ 3, tks to ankaa3908 again !
0.20c: Shuffle list
0.20b: Extract element with cyclic version of list
0.20a: Sub-list extraction + transform list + merge/intersection + improvements
0.1.9: Improvements with $\LaTeX$ 3, tks to ankaa3908
0.1.8: Get index from value
0.1.7: Improvements with $\LaTeX$ 3
0.1.6: Initial version, code written in $\LaTeX$ 3

8 The code

```
% Author      : C. Pierquet
% Credits     : V. Dao, aka ankaa3908, for improvements with latex3 code
% licence     : Released under the LaTeX Project Public License v1.3c or later, see
               http://www.latex-project.org/lppl.txt

\NeedsTeXFormat{LaTeX2e}
\def\ctfiledate{2026/07/23}%
\def\ctfileversion{0.20e}%
\ProvidesExplPackage{commalists-tools-l3}{\ctfiledate}{\ctfileversion}{Basic operations for numeral comma separated
lists}

%-----History
% 0.20e Bugfix with cycl + separator improvements
% 0.20d Improvements with latex3 (tks to ankaa3908, again !)
% 0.20c Shuffle list
% 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
\seq_new:N \l__commalists_tmpa_seq
\seq_new:N \l__commalists_tmpb_seq
\int_new:N \l__commalists_tmpa_int
\int_new:N \l__commalists_tmpb_int
\fp_new:N \l__commalists_tmpa_fp

%-----Show list
\NewDocumentCommand\ctshowlist{ 0 { , } m }
{
  % #1 = sep (entrée ET sortie)
  % #2 = list
  \seq_set_split:Nne \l__commalists_tmpa_seq { #1 } { #2 }
  \seq_use:Nn \l__commalists_tmpa_seq { #1 }
}

\NewDocumentCommand\ctshowlistwithsep{ 0 { , } m m }
{
  % #1 = sep (entrée)
  % #2 = list
  % #3 = sep (sortie)
  \seq_set_split:Nne \l__commalists_tmpa_seq { #1 } { #2 }
  \seq_use:Nn \l__commalists_tmpa_seq { #3 }
}

%-----Sort list, reverse list
\NewDocumentCommand\ctsortasclist{ s 0 { , } m 0 { \mysortedlist } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
```

```

% #3 = list
% #4 = output macro (non-starred, default \mysortedlist)
\group_begin:
\seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
\seq_sort:Nn \l__commalists_tmpa_seq {
  \fp_compare:nNnTF { ##1 } > { ##2 }
    { \sort_return_swapped: }
    { \sort_return_same: }
}
\IfBooleanTF { #1 }
{ \seq_use:Nn \l__commalists_tmpa_seq { #2 } }
{ \tl_gset:Ne #4 { \seq_use:Nn \l__commalists_tmpa_seq { #2 } } }
\group_end:
}

\NewDocumentCommand\ctsortdeslist{ s O { , } m O { \mysortedlist } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = list
  % #4 = output macro (non-starred, default \mysortedlist)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
  \seq_sort:Nn \l__commalists_tmpa_seq {
    \fp_compare:nNnTF { ##1 } < { ##2 }
      { \sort_return_swapped: }
      { \sort_return_same: }
  }
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_tmpa_seq { #2 } }
  { \tl_gset:Ne #4 { \seq_use:Nn \l__commalists_tmpa_seq { #2 } } }
  \group_end:
}

\NewDocumentCommand\ctreverselist{ s O { , } m O { \reverselist } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = list
  % #4 = output macro (non-starred, default \reverselist)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
  \seq_reverse:N \l__commalists_tmpa_seq
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_tmpa_seq { #2 } }
  { \tl_gset:Ne #4 { \seq_use:Nn \l__commalists_tmpa_seq { #2 } } }
  \group_end:
}

%-----Test in list
\NewDocumentCommand\ctboolvalinlist{ O { , } m m O { \resisinlist } }
{
  % #1 = sep (défaut virgule)
  % #2 = element to test
  % #3 = list
  % #4 = output macro (non-starred, default \resisinlist)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #1 } { #3 }
  \seq_if_in:NnTF \l__commalists_tmpa_seq { #2 }
    { \tl_gset:Nn #4 { 1 } }
    { \tl_gset:Nn #4 { 0 } }
  \group_end:
}

\NewDocumentCommand\cttestifvalinlist{ O { , } m m }
{
  % #1 = sep (défaut virgule)
  % #2 = element to test
  % #3 = list
  \seq_set_split:Nne \l__commalists_tmpa_seq { #1 } { #3 }
  \seq_if_in:NnTF \l__commalists_tmpa_seq { #2 }
    { \use_i:nn }

```

```

    { \use_ii:nn }
  }

%-----Add, remove
\NewDocumentCommand\ctaddvalinlist{ s O { , } m m }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = value to add
  % #4 = list
  \IfBooleanTF { #1 }
  { #4 #2 #3 }
  { \tl_set:Ne #4 { \tl_use:N #4 #2 #3 } }
}

\NewDocumentCommand\ctremovevalinlist{ s O { , } m m O { \mytmplist } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = value to remove
  % #4 = list
  % #5 = output macro (non-starred, default \mytmplist)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
  \seq_remove_all:Nn \l__commalists_tmpa_seq { #3 }
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_tmpa_seq { #2 } }
  { \tl_gset:Ne #5 { \seq_use:Nn \l__commalists_tmpa_seq { #2 } } }
  \group_end:
}

%-----Count
\NewDocumentCommand\ctcountvalinlist{ s O { , } m m O { \rescount } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = value to count
  % #4 = list
  % #5 = output macro (non-starred, default \rescount)
  \group_begin:
  \int_zero:N \l__commalists_tmpa_int
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
  \seq_map_inline:Nn \l__commalists_tmpa_seq
  { \str_if_eq:nnT { ##1 } { #3 } { \int_incr:N \l__commalists_tmpa_int } }
  \IfBooleanTF{ #1 }
  { \int_use:N \l__commalists_tmpa_int }
  { \tl_gset:Ne #5 { \int_use:N \l__commalists_tmpa_int } }
  \group_end:
}

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

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

\NewDocumentCommand\ctsumoflist{ s O { , } m O { \ressum } }

```

```

{
% #1 = star (just printing)
% #2 = sep (défaut virgule)
% #3 = list
% #4 = output macro (non-starred, default \ressum)
\group_begin:
\seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
\fp_zero:N \l__commalists_tmpa_fp
\fp_set:Nn \l__commalists_tmpa_fp
{ \exp_args:Ne \fp_eval:n { \seq_use:Nn \l__commalists_tmpa_seq { + } } }
\IfBooleanTF{ #1 }
{ \fp_use:N \l__commalists_tmpa_fp }
{ \tl_gset:Ne #4 { \fp_use:N \l__commalists_tmpa_fp } }
\group_end:
}

\NewDocumentCommand\ctprodoftlist{ s O { , } m O { \resprod } }
{
% #1 = star (just printing)
% #2 = sep (défaut virgule)
% #3 = list
% #4 = output macro (non-starred, default \resprod)
\group_begin:
\seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
\fp_zero:N \l__commalists_tmpa_fp
\fp_set:Nn \l__commalists_tmpa_fp
{ \exp_args:Ne \fp_eval:n { \seq_use:Nn \l__commalists_tmpa_seq { * } } }
\IfBooleanTF{ #1 }
{ \fp_use:N \l__commalists_tmpa_fp }
{ \tl_gset:Ne #4 { \fp_use:N \l__commalists_tmpa_fp } }
\group_end:
}

\NewDocumentCommand\ctmeanoflist{ s O { , } m O { \resmean } }
{
% #1 = star (just printing)
% #2 = sep (défaut virgule)
% #3 = list
% #4 = output macro (non-starred, default \resmean)
\group_begin:
\seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
\fp_zero:N \l__commalists_tmpa_fp
\fp_set:Nn \l__commalists_tmpa_fp
{ \exp_args:Ne \fp_eval:n
{
( \seq_use:Nn \l__commalists_tmpa_seq { + } )
/
\seq_count:N \l__commalists_tmpa_seq
}
}
\IfBooleanTF{ #1 }
{ \fp_use:N \l__commalists_tmpa_fp }
{ \tl_gset:Ne #4 { \fp_use:N \l__commalists_tmpa_fp } }
\group_end:
}

\NewDocumentCommand\ctlentoflist{ s O { , } m O { \resmylen } }
{
% #1 = star (just printing)
% #2 = sep (défaut virgule)
% #3 = list
% #4 = output macro (non-starred, default \myreslen)
\group_begin:
\seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
\IfBooleanTF { #1 }
{ \seq_count:N \l__commalists_tmpa_seq }
{ \tl_gset:Ne #4 { \seq_count:N \l__commalists_tmpa_seq } }
\group_end:
}

%-----Get, index
\NewDocumentCommand\ctgetvaluefromlist{ s O { , } m m O { \resmyelt } }

```

```

{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = list
  % #4 = index
  % #5 = output macro (non-starred, default \resmyelt)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
  \IfBooleanTF { #1 }
  { \seq_item:Nn \l__commalists_tmpa_seq { #4 } }
  { \tl_gset:Ne #5 { \seq_item:Nn \l__commalists_tmpa_seq { #4 } } }
  \group_end:
}

% message d'erreur
\msg_new:nnn { commalist-tools } { index-zero }
{ ctgetvaluefromcyccllist:~index~0~not~available }

\NewDocumentCommand\ctgetvaluefromcyccllist{ s O { , } m m O { \resmyelt } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = list
  % #4 = index
  % #5 = output macro (default \resmyelt)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
  \int_compare:nNnTF { #4 } = { 0 }
  {
    \msg_warning:nn { commalist-tools } { index-zero }
    \tl_set_eq:Nn #5 \c_empty_tl
  }
  {
    \int_set:Nn \l__commalists_tmpa_int
    {
      \int_mod:nn
      { #4 }
      { \seq_count:N \l__commalists_tmpa_seq }
    }
    % si mod = 0 (index multiple de la longueur) dernier élément
    \int_compare:nNnT { \l__commalists_tmpa_int } = { 0 }
    {
      \int_set:Nn \l__commalists_tmpa_int
      { \seq_count:N \l__commalists_tmpa_seq }
    }
    \IfBooleanTF{ #1 }
    {
      \exp_args:NNV
      \seq_item:Nn \l__commalists_tmpa_seq \l__commalists_tmpa_int
    }
    {
      \tl_gset:Ne #5 {
        \exp_args:NNV
        \seq_item:Nn \l__commalists_tmpa_seq \l__commalists_tmpa_int }
    }
  }
  \group_end:
}

\NewDocumentCommand\ctgetindexfromlist{ s O { , } m m O { \resmyindex } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = element
  % #4 = list
  % #5 = output macro (non-starred, default \resmyindex)
  \group_begin:
  \int_zero:N \l__commalists_tmpa_int
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
  \seq_if_in:NnT \l__commalists_tmpa_seq { #3 }
  {
    \seq_map_inline:Nn \l__commalists_tmpa_seq

```

```

    {
      \int_incr:N \l__commalists_tmpa_int
      \str_if_eq:nnT { ##1 } { #3 }
      { \seq_map_break: }
    }
  }
\IfBooleanTF { #1 }
{ \int_use:N \l__commalists_tmpa_int }
{ \tl_gset:Ne #5 { \int_use:N \l__commalists_tmpa_int } }
\group_end:
}

%-----Sub-list
\seq_new:N \l__commalists_sub_seq
\int_new:N \l__commalists_begin_int
\int_new:N \l__commalists_end_int

\NewDocumentCommand\ctsublist{ s O { , } m m m O { \mysublist } }
{
  % #1 = star (display only)
  % #2 = sep (défaut virgule)
  % #3 = list
  % #4 = begin index (* = start from first)
  % #5 = end index (* = go to last)
  % #6 = output macro (non-starred only, default \mysublist)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
  % --- resolve begin index
  \int_set:Nn \l__commalists_begin_int
  {
    \str_if_eq:nnTF { * } { #4 }
    { 1 }
    { #4 }
  }

  % --- resolve end index
  \int_set:Nn \l__commalists_end_int
  {
    \str_if_eq:nnTF { * } { #5 }
    { \seq_count:N \l__commalists_tmpa_seq }
    { #5 }
  }

  % --- iterate and collect items in [begin, end]
  \seq_map_indexed_inline:Nn \l__commalists_tmpa_seq
  {
    \bool_lazy_and:nnT
    { \int_compare_p:n { ##1 >= \l__commalists_begin_int } }
    { \int_compare_p:n { ##1 <= \l__commalists_end_int } }
    { \seq_put_right:Nn \l__commalists_sub_seq { ##2 } }
  }

  % --- output
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_sub_seq { #2 } }
  { \tl_gset:Ne #6 { \seq_use:Nn \l__commalists_sub_seq { #2 } } }
  \group_end:
}

%-----Slice-list
\seq_new:N \l__commalists_slice_seq
\tl_new:N \l__commalists_slice_formula_tl

\NewDocumentCommand\ctslitelist{ s O { , } m m m O { \myslicelist } }
{
  % #1 = star (display only)
  % #2 = sep (défaut virgule)
  % #3 = list
  % #4 = formula in x (e.g. 2*x, 3*x+1, x^2)
  % #5 = output macro (non-starred only, default \myslicelist)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }

```

```

\seq_map_indexed_inline:Nn \l__commalists_tmpa_seq
{
  \tl_set:Nn \l__commalists_slice_formula_tl { #4 }
  \tl_replace_all:Nnn \l__commalists_slice_formula_tl { x } { ##1 }
  \int_set:Nn \l__commalists_tmpa_int
    { \fp_eval:n { \l__commalists_slice_formula_tl } }
  \int_compare:nNnTF
    { \l__commalists_tmpa_int }
    <
    { \seq_count:N \l__commalists_tmpa_seq +1 }
    {
      \seq_put_right:Ne \l__commalists_slice_seq
        { \seq_item:Nn \l__commalists_tmpa_seq
          { \l__commalists_tmpa_int }
        }
    }
  { \seq_map_break: }
}
% --- output
\IfBooleanTF { #1 }
{ \seq_use:Nn \l__commalists_slice_seq { #2 } }
{ \tl_gset:Ne #5 { \seq_use:Nn \l__commalists_slice_seq { #2 } } }
\group_end:
}

%-----Transform-list
\seq_new:N \l__commalists_transform_seq
\tl_new:N \l__commalists_transform_formula_tl

\NewDocumentCommand\cttransformlist{ s O { , } d < > m m O { \mytransformlist } }
{
  % #1 = star (display only)
  % #2 = sep (défaut virgule)
  % #3 = round (optionnel, entre < >, nombre de décimales)
  % #4 = list
  % #5 = formula in x (e.g. 2*x+1, x^2, sqrt(x))
  % #6 = output macro (non-starred only, default \mytransformlist)
  \seq_clear:N \l__commalists_transform_seq
  \group_begin:
  % --- sécurité : [] explicite ne redonne pas le défaut O{,} en xparse
  \tl_if_empty:nTF { #2 }
  { \seq_set_split:Nne \l__commalists_tmpa_seq { , } { #4 } }
  { \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 } }
  % --- iterate over each element, apply formula
  \seq_map_inline:Nn \l__commalists_tmpa_seq
  {
    % --- substitute x by current element value in formula
    \tl_set:Nn \l__commalists_transform_formula_tl { #5 }
    \tl_replace_all:Nnn \l__commalists_transform_formula_tl { x } { ##1 }
    % --- evaluate and collect result
    \IfNoValueTF{ #3 }
    {
      \seq_put_right:Ne \l__commalists_transform_seq
        { \fp_eval:n { \l__commalists_transform_formula_tl } }
    }
    {
      \seq_put_right:Ne \l__commalists_transform_seq
      {
        \fp_eval:n {
          round( \l__commalists_transform_formula_tl, #3 )
        }
      }
    }
  }
}

% --- output
\tl_if_empty:nTF { #2 }
{
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_transform_seq { , } }
  { \tl_gset:Ne #6 { \seq_use:Nn \l__commalists_transform_seq { , } } }
}

```

```

{
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_transform_seq { #2 } }
  { \tl_gset:Ne #6 { \seq_use:Nn \l__commalists_transform_seq { #2 } } }
}
\group_end:
}

%-----Unique list
\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 { , } d < > m O { \myuniqlist } }
{
  % #1 = star (display only)
  % #2 = sep (défaut virgule)
  % #3 = options (asc, des), entre < >
  % #4 = list
  % #5 = output macro (non-starred only, default \myuniqlist)
  \group_begin:
  % parse keys
  \IfNoValueF { #3 } { \keys_set:nn { commalists / uniq } { #3 } }

  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
  \seq_remove_duplicates:N \l__commalists_tmpa_seq

  % --- sort if requested
  \bool_case:n {
    { \l__commalists_uniq_asc_bool }
    {
      \seq_sort:Nn \l__commalists_tmpa_seq
      {
        \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
      }
    }
    { \l__commalists_uniq_des_bool }
    {
      \seq_sort:Nn \l__commalists_tmpa_seq
      {
        \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
      }
    }
  }

  % --- output
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_tmpa_seq { #2 } }
  { \tl_gset:Ne #5 { \seq_use:Nn \l__commalists_tmpa_seq { #2 } } }
  \group_end:
}

%-----Intersection
\seq_new:N \l__commalists_short_sequence_seq
\seq_new:N \l__commalists_long_sequence_seq
\seq_new:N \l__commalists_intersect_seq
\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 0 { , } d < > m m 0 { \myintersectlist }
}
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = options (asc, des), entre < >
  % #4 = list 1
  % #5 = list 2
  % #6 = output macro (non-starred, default \myintersectlist)
  \group_begin:
  % parse keys
  \IfNoValueF { #3 } { \keys_set:nn { commalists / intersect } { #3 } }
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
  \seq_set_split:Nne \l__commalists_tmpb_seq { #2 } { #5 }
  % --- keep the shortest list for iteration
  \int_set:Nn \l__commalists_tmpa_int { \seq_count:N \l__commalists_tmpa_seq }
  \int_set:Nn \l__commalists_tmpb_int { \seq_count:N \l__commalists_tmpb_seq }
  \int_compare:nNnTF { \l__commalists_tmpa_int } < { \l__commalists_tmpb_int }
  {
    \seq_set_eq:NN \l__commalists_short_sequence_seq \l__commalists_tmpa_seq
    \seq_set_eq:NN \l__commalists_long_sequence_seq \l__commalists_tmpb_seq
  }
  {
    \seq_set_eq:NN \l__commalists_short_sequence_seq \l__commalists_tmpb_seq
    \seq_set_eq:NN \l__commalists_long_sequence_seq \l__commalists_tmpa_seq
  }
  \seq_map_inline:Nn \l__commalists_short_sequence_seq
  {
    \seq_if_in:NnT \l__commalists_long_sequence_seq { ##1 }
    { \seq_put_right:Nn \l__commalists_intersect_seq { ##1 } }
  }
  \seq_remove_duplicates:N \l__commalists_intersect_seq
  % --- sort if requested
  \bool_case:n {
    { \l__commalists_intersect_asc_bool }
    {
      \seq_sort:Nn \l__commalists_intersect_seq
      {
        \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
      }
    }
    { \l__commalists_intersect_des_bool }
    {
      \seq_sort:Nn \l__commalists_intersect_seq
      {
        \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
      }
    }
  }
  % --- output
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_intersect_seq { #2 } }
  { \tl_gset:Ne #6 { \seq_use:Nn \l__commalists_intersect_seq { #2 } } }
  \group_end:
}

%-----Union
\seq_new:N \l__commalists_merge_seq
\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 { , } d < > m m O { \mymergelist } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = options (asc, des), entre < >
  % #4 = list 1
  % #5 = list 2
  % #6 = output macro (non-starred, default \mymergelist)
  \group_begin:
  % keys reading
  \IfNoValueF { #3 } { \keys_set:nn { commalists / merge } { #3 } }
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
  \seq_set_split:Nne \l__commalists_tmpb_seq { #2 } { #5 }
  \seq_concat:NNN \l__commalists_merge_seq
    \l__commalists_tmpa_seq
    \l__commalists_tmpb_seq
  \seq_remove_duplicates:N \l__commalists_merge_seq
  % --- sort if requested
  \bool_case:n {
    { \l__commalists_merge_asc_bool }
    {
      \seq_sort:Nn \l__commalists_merge_seq
      {
        \fp_compare:nNnTF { ##1 } > { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
      }
    }
    { \l__commalists_merge_des_bool }
    {
      \seq_sort:Nn \l__commalists_merge_seq
      {
        \fp_compare:nNnTF { ##1 } < { ##2 }
        { \sort_return_swapped: }
        { \sort_return_same: }
      }
    }
  }
  % --- output
  \IfBooleanTF { #1 }
  { \seq_use:Nn \l__commalists_merge_seq { #2 } }
  { \tl_gset:Ne #6 { \seq_use:Nn \l__commalists_merge_seq { #2 } } }
  \group_end:
}

%-----Difference

\seq_new:N \l__commalists_diff_seq
\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 { , } d < > m m O { \mydifflist } }
{
  % #1 = star (display only)
  % #2 = sep (défaut virgule)

```

```

% #3 = options (asc, des), entre < >
% #4 = list A
% #5 = list B
% #6 = output macro (non-starred, default \mydifflist)
\group_begin:
% parse keys
\IfNoValueF { #3 } { \keys_set:nn { commalists / diff } { #3 } }
% --- iterate A, keep elements absent from B
\seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
\seq_set_split:Nne \l__commalists_tmpb_seq { #2 } { #5 }
\seq_map_inline:Nn \l__commalists_tmpa_seq
{
  \seq_if_in:NnF \l__commalists_tmpb_seq { ##1 }
  { \seq_put_right:Nn \l__commalists_diff_seq { ##1 } }
}
\seq_remove_duplicates:N \l__commalists_diff_seq
% --- sort if requested
\bool_case:n {
  { \l__commalists_diff_asc_bool }
  {
    \seq_sort:Nn \l__commalists_diff_seq
    {
      \fp_compare:nNnTF { ##1 } > { ##2 }
      { \sort_return_swapped: }
      { \sort_return_same: }
    }
  }
  { \l__commalists_diff_des_bool }
  {
    \seq_sort:Nn \l__commalists_diff_seq
    {
      \fp_compare:nNnTF { ##1 } < { ##2 }
      { \sort_return_swapped: }
      { \sort_return_same: }
    }
  }
}
% --- output
\IfBooleanTF { #1 }
{ \seq_use:Nn \l__commalists_diff_seq { #2 } }
{ \tl_gset:Ne #6 { \seq_use:Nn \l__commalists_diff_seq { #2 } } }
\group_end:
}

%-----Symmetric difference
\seq_new:N \l__commalists_syndiff_seq
\bool_new:N \l__commalists_syndiff_asc_bool
\bool_new:N \l__commalists_syndiff_des_bool

\keys_define:nn { commalists / syndiff }
{
  asc .bool_set:N = \l__commalists_syndiff_asc_bool,
  asc .default:n = true,
  des .bool_set:N = \l__commalists_syndiff_des_bool,
  des .default:n = true,
}

\NewDocumentCommand\ctsyndifflist{ s O { , } d < > m m O { \mysyndifflist } }
{
  % #1 = star (display only)
  % #2 = sep (défaut virgule)
  % #3 = options (asc, des), entre < >
  % #4 = list A
  % #5 = list B
  % #6 = output macro (non-starred, default \mysyndifflist)
  \group_begin:
  % parse keys
  \IfNoValueF { #3 } { \keys_set:nn { commalists / syndiff } { #3 } }
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #4 }
  \seq_set_split:Nne \l__commalists_tmpb_seq { #2 } { #5 }
  % --- elements in A absent from B
  \seq_map_inline:Nn \l__commalists_tmpa_seq

```

```

{
  \seq_if_in:NnF \l__commalists_tmpb_seq { ##1 }
  {
    \seq_if_in:NnF \l__commalists_syndiff_seq { ##1 }
    { \seq_put_right:Nn \l__commalists_syndiff_seq { ##1 } }
  }
}
% --- elements in B absent from A
\seq_map_inline:Nn \l__commalists_tmpb_seq
{
  \seq_if_in:NnF \l__commalists_tmpa_seq { ##1 }
  {
    \seq_if_in:NnF \l__commalists_syndiff_seq { ##1 }
    { \seq_put_right:Nn \l__commalists_syndiff_seq { ##1 } }
  }
}
% --- sort if requested
\bool_case:n {
  { \l__commalists_syndiff_asc_bool }
  {
    \seq_sort:Nn \l__commalists_syndiff_seq
    {
      \fp_compare:nNnTF { ##1 } > { ##2 }
      { \sort_return_swapped: }
      { \sort_return_same: }
    }
  }
  { \l__commalists_syndiff_des_bool }
  {
    \seq_sort:Nn \l__commalists_syndiff_seq
    {
      \fp_compare:nNnTF { ##1 } < { ##2 }
      { \sort_return_swapped: }
      { \sort_return_same: }
    }
  }
}
}
% --- output
\IfBooleanTF { #1 }
{ \seq_use:Nn \l__commalists_syndiff_seq { #2 } }
{ \tl_gset:Ne #6 { \seq_use:Nn \l__commalists_syndiff_seq { #2 } } }
\group_end:
}

%-----Shuffle list
\NewDocumentCommand\ctshufflelist{ s O { , } m O { \myshuffledlist } }
{
  % #1 = star (just printing)
  % #2 = sep (défaut virgule)
  % #3 = list (content)
  % #4 = output macro (non-starred, default \myshuffledlist)
  \group_begin:
  \seq_set_split:Nne \l__commalists_tmpa_seq { #2 } { #3 }
  \seq_shuffle:N \l__commalists_tmpa_seq
  % --- output
  \IfBooleanTF{ #1 }
  { \seq_use:Nn \l__commalists_tmpa_seq { #2 } }
  { \tl_gset:Ne #4 { \seq_use:Nn \l__commalists_tmpa_seq { #2 } } }
  \group_end:
}

\endinput

```