

## Chapter 8

# Operations with lists

Lists are a type of calculator's object that can be useful for data processing and in programming. This Chapter presents examples of operations with lists.

## Definitions

A list, within the context of the calculator, is a series of objects enclosed between braces and separated by spaces ( $\overline{\text{SPC}}$ ), in the RPN mode, or commas ( $\overline{\text{,}}$ ), in both modes. Objects that can be included in a list are numbers, letters, character strings, variable names, and/or operators. Lists are useful for manipulating data sets and in some programming applications. Some examples of lists are:

```
{ t 1 }, { "BETA" h2 4 }, { 1 1.5 2.0 },
{ a a a a }, { { 1 2 3 } { 3 2 1 } { 1 2 3 } }
```

In the examples shown below we will limit ourselves to numerical lists.

## Creating and storing lists

To create a list in ALG mode, first enter the braces key ( $\overline{\{ \}}$ ) (associated with the  $\overline{+}$  key), then type or enter the elements of the list, separating them with commas ( $\overline{\text{,}}$ ). The following keystrokes will enter the list {1 2 3 4} and store it into variable L1.

$\overline{\{ \}}$   $\overline{1}$   $\overline{\text{,}}$   $\overline{2}$   $\overline{\text{,}}$   $\overline{3}$   $\overline{\text{,}}$   $\overline{4}$   
 $\overline{\text{STO}}$   $\overline{\text{ALPHA}}$   $\overline{\text{L}}$   $\overline{1}$   $\overline{\text{ENTER}}$

The screen will show the following:

```
{1,2,3,4}►L1
+SKIP+SKIP+ +DEL|DEL+|DEL L|INS =
```

```
{1. 2. 3. 4.}►L1
{1. 2. 3. 4.}
+SKIP+SKIP+ +DEL|DEL+|DEL L|INS =
```

The figure to the left shows the screen before pressing  $\overline{\text{ENTER}}$ , while the one to the right shows the screen after storing the list into L1. Notice that before pressing  $\overline{\text{ENTER}}$  the list shows the commas separating its elements. However, after pressing  $\overline{\text{ENTER}}$ , the commas are replaced with spaces.

Entering the same list in RPN mode requires the following keystrokes:

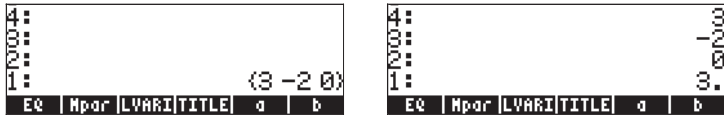
$\overline{\{ \}}$   $\overline{1}$   $\overline{\text{SPC}}$   $\overline{2}$   $\overline{\text{SPC}}$   $\overline{3}$   $\overline{\text{SPC}}$   $\overline{4}$   $\overline{\text{ENTER}}$   
 $\overline{\text{ALPHA}}$   $\overline{\text{L}}$   $\overline{1}$   $\overline{\text{ENTER}}$   $\overline{\text{STO}}$

The figure below shows the RPN stack before pressing the **(STOP)** key:



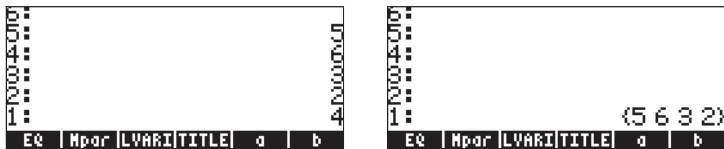
## Composing and decomposing lists

Composing and decomposing lists makes sense in RPN mode only. Under such operating mode, decomposing a list is achieved by using function **OBJ→**. With this function, a list in the RPN stack is decomposed into its elements, with stack level 1: showing the number of elements in the list. The next two screen shots show the stack with a small list before and after application of function **OBJ→**:

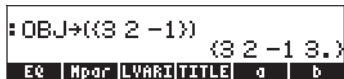


Notice that, after applying **OBJ→**, the elements of the list occupy levels 4: through 2:, while level 1: shows the number of elements in the list.

To compose a list in RPN mode, place the elements of the list in the stack, enter the list size, and apply function **→LIST** (select it from the function catalog, as follows: **(P) CAT (P) →**, then use the up and down arrow keys (**▲ ▼**) to locate function **→LIST**). The following screen shots show the elements of a list of size 4 before and after application of function **→LIST**:



**Note:** Function **OBJ→** applied to a list in ALG mode simply reproduces the list, adding to it the list size:



## Operations with lists of numbers

To demonstrate operations with lists of numbers, we will create a couple of other lists, besides list L1 created above: L2={-3,2,1,5}, L3={-6,5,3,1,0,3,-4}, L4={3,-2,1,5,3,2,1}. In ALG mode, the screen will look like this after entering lists L2, L3, L4:

```

: (-3 2 1 5) ► L2
: (-6 5 3 1 0 3 -4) ► L3
: (3 -2 1 5 3 2 1) ► L4
      (-3 2 1 5)
      (-6 5 3 1 0 3 -4)
      (3 -2 1 5 3 2 1)
L4 | L3 | L2 | L1 | TRIAN MES1

```

In RPN mode, the following screen shows the three lists and their names ready to be stored. To store the lists in this case you need to press **STOP** three times.

## Changing sign

The sign-change key (**+/-**), when applied to a list of numbers, will change the sign of all elements in the list. For example:

```

: L1
      (1. 2. 3. 4.)
: -L1
      (-1. -2. -3. -4.)
L2 | L3 | L4 | L1 | TRIAN MES1

```

## Addition, subtraction, multiplication, division

Multiplication and division of a list by a single number is distributed across the list, for example:

```

: -5 L2
      (15 -10 -5 -25)
: L1 / 5
      (.2 .4 .6 .8)
L2 | L3 | L4 | L1 | TRIAN MES1

```

Subtraction of a single number from a list will subtract the same number from each element in the list, for example:

```

: L2
      (-3 2 1 5)
: L2 - 10
      (-13. -8. -9. -5.)
L2 | L3 | L4 | L1 | TRIAN MES1

```

Addition of a single number to a list produces a list augmented by the number, and not an addition of the single number to each element in the list. For example:

```

: L1
      (1 2 3 4)
: L1 + 6
      (1 2 3 4 6)
L2 | L3 | L4 | L1 | TRIAN MES1

```



## ABS

```
:L2
(-3 2 1 5)
:IL2
(3 2 1 5)
```

L2 | L3 | L4 | L1 | TRIAN | MES1

## EXP and LN

```
:e L1
(e1 e2 e3 e4)
:LN(L1)
(0 LN(2) LN(3) 2·LN(2))
```

L2 | L3 | L4 | L1 | TRIAN | MES1

## LOG and ANTILOG

```
:LOG(L1)
(0 LOG(2) LOG(3) LOG(4))
:ALOG(L2)
(1/1000 100 10 100000)
```

L2 | L3 | L4 | L1 | TRIAN | MES1

## SQ and square root

```
:SQ(L1)
(1 4 9 16)
:√L2
(√-1 √3 √2 1 √5)
```

L2 | L3 | L4 | L1 | TRIAN | MES1

## SIN, ASIN

```
:SIN(L1)
(SIN(1) SIN(2) SIN(3) SIN(4))
:ASIN(L2)
(-.304692654015 .20135)
```

L2 | L3 | L4 | L1 | TRIAN | MES1

## COS, ACOS

```
:COS(L2)
(COS(3) COS(2) COS(1) COS(5))
:ACOS(L1)
(1.47062890563 1.36943)
```

L2 | L3 | L4 | L1 | TRIAN | MES1

## TAN, ATAN

```
:TAN(L1)
(TAN(1) TAN(2) TAN(3) TAN(4))
:ATAN(L2)
(-ATAN(3) ATAN(2) π/4 ATAN(4))
```

L2 | L3 | L4 | L1 | TRIAN | MES1

## INVERSE (1/x)

```
:INV(L1)
(1 1/2 1/3 1/4)
```

L2 | L3 | L4 | L1 | TRIAN | MES1

# Real number functions from the MTH menu

Functions of interest from the MTH menu include, from the HYPERBOLIC menu: SINH, ASINH, COSH, ACOSH, TANH, ATANH, and from the REAL menu: %, %CH, %T, MIN, MAX, MOD, SIGN, MANT, XPON, IP, FP, RND, TRNC, FLOOR, CEIL, D→R, R→D. Some of the functions that take a single argument are illustrated below applied to lists of real numbers:

## SINH, ASINH

```
:SINH(L1)
(SINH(1) SINH(2) SINH(3) S<
:ASINH(L2)
(-.295673047563 .19869)
```

SINH | ASINH | COSH | ACOSH | TANH | ATANH

## COSH, ACOSH

```
:COSH(L2)
(COSH(3) COSH(2) COSH(1) C<
:ACOSH(L1)
(0 ACOSH(2) ACOSH(3) ACOS<
```

SINH | ASINH | COSH | ACOSH | TANH | ATANH

## TANH, ATANH

```

:TANH(L2)
(-TANH(3) TANH(2) TANH(1) )
:ATANH(L1)
(ATANH(1) ATANH(2) ATANH(3) )
SIGN | ASINH | COSH | ACOSH | TANH | ATANH

```

## SIGN, MANT, XPON

```

:SIGN(L1)
(1 1 1 1)
:MANT(100:L2)
(3. 2. 1. 5.)
:XPON(L1:100)
(2. 2. 2. 2.)
ABS | SIGN | MANT | XPON | IP | FP

```

## IP, FP

```

:IP((1.2 2.3 -1.5))
(1. 2. -1.)
:FP((1.2 2.3 -1.5))
(2. 3. -5.)
ABS | SIGN | MANT | XPON | IP | FP

```

## FLOOR, CEIL

```

:FLOOR((1.2 2.3 -1.5))
(1. 2. -2.)
:CEIL((1.2 2.3 -1.5))
(2. 3. -1.)
RND | TRNC | FLOOR | CEIL | D→R | R→D

```

## D→R, R→D

```

:D→R((30 60 90))
(.523598775598 1.04719)
:R→D((1/6 1/3 1/2))
(30. 60. 90.0000000002 90.0)
RND | TRNC | FLOOR | CEIL | D→R | R→D

```

## Examples of functions that use two arguments

The screen shots below show applications of the function % to list arguments. Function % requires two arguments. The first two examples show cases in which only one of the two arguments is a list.

```

:%(10 20 30),1)
(.1 .2 .3)
:%(5,(10 20 30))
(5. 1/10 5. 1/5 5. 3/10)
% | 2CH | 2T | MIN | MAX | MOD

```

The results are lists with the function % distributed according to the list argument. For example,

$$\%(\{10, 20, 30\}, 1) = \{\%(10, 1), \%(20, 1), \%(30, 1)\},$$

while

$$\%(5, \{10, 20, 30\}) = \{\%(5, 10), \%(5, 20), \%(5, 30)\}$$

In the following example, both arguments of function % are lists of the same size. In this case, a term-by-term distribution of the arguments is performed, i.e.,

$$\%(\{10,20,30\},\{1,2,3\}) = \{\%(10,1),\%(20,2),\%(30,3)\}$$

```

: %({10 20 30},{1 2 3})
{10 1 20 2 30 3}
2 | ZCH | ZT | MIN | MAX | MOD

```

This description of function % for list arguments shows the general pattern of evaluation of any function with two arguments when one or both arguments are lists. Examples of applications of function RND are shown next:

```

: RND({1/3 1/6 1/3},2)
{.33 .17 .33}
: RND(1/3,{2 3 4})
{.33 .333 .3333}
INTEG POLY MODUL PERM DIVIS FACTO

```

## Lists of complex numbers

The following exercise shows how to create a list of complex numbers given two lists of the same length, one representing the real parts and one the imaginary parts of the complex numbers. Use L1 ADD i\*L2.

```

RAD MYS HEX C= 'X'      ALG
CHOME3
{1/3 1/6 1/3}
{.33 .17 .33}
: RND(1/3,{2 3 4})
{.33 .333 .3333}
: L1 ADD i*L2
{1+i-3 2+i-2 3+i 4+i-5}
L4 | L3 | L2 | L1 | CASDI

```

Functions such as LN, EXP, SQ, etc., can also be applied to a list of complex numbers, e.g.,

```

: SQ(L5)
{SQ(1+i-3) SQ(2+i-2) SQ(3+i)}
: JL5
{((3+i)√2-2i√5)√1+√10} (1)
L5 | L2 | L3 | L4 | L1 | TRIAN

```

```

: L5
{e 1+i-3 e 2+i-2 e 3+i e 4+i-5}
: LN(L5)
{LN(1+i-3) LN(2+i-2) LN(3+i)}
L5 | L2 | L3 | L4 | L1 | TRIAN

```

```

:ALOG(L5)
:(ALOG(1+i,-3) ALOG(2+i,2) ▶
:LOG(L5)
:(LOG(1+i,-3) LOG(2+i,2) LO▶
:INV(L5)
:
$$\left\{ \frac{1}{1+i-3} \frac{1}{2+i,2} \frac{1}{3+i} \frac{1}{4+i,5} \right\}$$

L5 L2 L3 L4 L1 TRIAN

```

```

:SIN(L5)
:(SIN(1+i,-3) SIN(2+i,2) SI▶
: SINH(L5)
:(SINH(1+i,-3) SINH(2+i,2) ▶
:ASIN(L5)
:(ASIN(1+i,-3) ASIN(2+i,2) ▶
L5 L2 L3 L4 L1 TRIAN

```

The following example shows applications of the functions RE(Real part), IM(imaginary part), ABS(magnitude), and ARG(argument) of complex numbers. The results are lists of real numbers:

```

:RE(L5)
:IM(L5)
:IL5)
:
$$\left\{ \sqrt{10} \ 2\sqrt{2} \ \sqrt{10} \ \sqrt{41} \right\}$$

L5 L2 L3 L4 L1 TRIAN

```

```

:ARG(L5)
:
$$\left\{ -\text{ATAN}(3) \ \frac{\pi}{4} \ \text{ATAN}\left(\frac{1}{3}\right) \ \text{ATAN}\left(\frac{1}{2}\right) \right\}$$

L5 L2 L3 L4 L1 TRIAN

```

## Lists of algebraic objects

The following are examples of lists of algebraic objects with the function SIN applied to them:

```

:
$$\left\{ \frac{f}{2} \ \alpha-\beta \ \frac{(x-y)^2}{4} \right\}$$

:
$$\left\{ \frac{f}{2} \ \alpha-\beta \ \frac{(x-y)^2}{4} \right\}$$

L5 L2 L3 L4 L1 TRIAN

```

```

:SIN(ANS(1))
:
$$\left\{ \frac{f}{2} \ \alpha-\beta \ \frac{(x-y)^2}{4} \right\}$$

:
$$\left\{ \sin\left(\frac{f}{2}\right) \ \sin(\alpha-\beta) \ \sin\left(\frac{(x-y)^2}{4}\right) \right\}$$

L5 L2 L3 L4 L1 TRIAN

```

## The MTH/LIST menu

The MTH menu provides a number of functions that exclusively to lists. With flag 117 set to CHOOSE boxes:



Next, with system flag 117 set to SOFT menus:



VECTRMATR: LIST MYP REAL BASE

ΔLIST ΣLIST ΠLIST SORT REVL ADD

This menu contains the following functions:

- ΔLIST : Calculate increment among consecutive elements in list
- ΣLIST : Calculate summation of elements in the list
- ΠLIST : Calculate product of elements in the list
- SORT : Sorts elements in increasing order
- REVLIST : Reverses order of list
- ADD : Operator for term-by-term addition of two lists of the same length (examples of this operator were shown above)

Examples of application of these functions in ALG mode are shown next:

```

:L3      (-6 5 3 1 0 3 -4)
:ΔLIST(L3)
      (11 -2 -2 -1 3 -7)
ΔLIST ΣLIST ΠLIST SORT REVL ADD
:L3      (-6 5 3 1 0 3 -4)
:ΣLIST(L3)
      (-6 -4 0 1 3 3 5)
ΔLIST ΣLIST ΠLIST SORT REVL ADD

```

```

:L3      (-6 5 3 1 0 3 -4)
:ΣLIST(L3)
      2
ΔLIST ΣLIST ΠLIST SORT REVL ADD
:L3      (-6 5 3 1 0 3 -4)
:REVLIST(L3)
      (-4 3 0 1 3 5 -6)
ΔLIST ΣLIST ΠLIST SORT REVL ADD

```

SORT and REVLIST can be combined to sort a list in decreasing order:

```

:L3      (-6 5 3 1 0 3 -4)
:REVLIST(SORT(L3))
      (5 3 1 0 -4 -6)
ΔLIST ΣLIST ΠLIST SORT REVL ADD

```

If you are working in RPN mode, enter the list onto the stack and then select the operation you want. For example, to calculate the increment between consecutive elements in list L3, press:



This places L3 onto the stack and then selects the ΔLIST operation from the MTH menu.

## Manipulating elements of a list

The PRG (programming) menu includes a LIST sub-menu with a number of functions to manipulate elements of a list. With system flag 117 set to CHOOSE boxes:



Item 1. ELEMENTS.. contains the following functions that can be used for the manipulation of elements in lists:



### List size

Function SIZE, from the PRG/LIST/ELEMENTS sub-menu, can be used to obtain the size (also known as length) of the list, e.g.,

```
:L3
      (-6 5 3 1 0 3 -4)
:SIZE(L3)
      7.
```

L5 | L2 | L3 | L4 | L1 | TRIAN

### Extracting and inserting elements in a list

To extract elements of a list we use function GET, available in the PRG/LIST/ELEMENTS sub-menu. The arguments of function GET are the list and the number of the element you want to extract. To insert an element into a list use function PUT (also available in the PRG/LIST/ELEMENTS sub-menu). The arguments of function PUT are the list, the position that one wants to replace, and the value that will be replaced. Examples of applications of functions GET and PUT are shown in the following screen:

```
:GET(L3,5)
      0
:PUT(L3,5,10)
      (-6 5 3 1 10 3 -4)
```

L5 | L2 | L3 | L4 | L1 | TRIAN

Functions GETI and PUTI, also available in sub-menu PRG/ ELEMENTS/, can also be used to extract and place elements in a list. These two functions, however, are useful mainly in programming. Function GETI uses the same arguments as GET and returns the list, the element location plus one, and the element at the location requested. Function PUTI uses the same arguments as GET and returns the list and the list size.

### Element position in the list

To determine the position of an element in a list use function POS having the list and the element of interest as arguments. For example,

```
:L3
      (-6 5 3 1 0 3 -4)
:POS(L3,5)
      2.
L5 | L2 | L3 | L4 | L1 | TRAIN
```

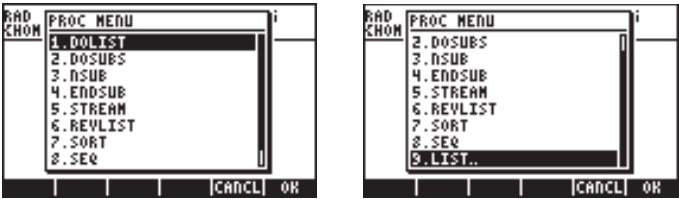
### HEAD and TAIL functions

The HEAD function extracts the first element in the list. The TAIL function removes the first element of a list, returning the remaining list. Some examples are shown next:

```
:L3
      (-6 5 3 1 0 3 -4)
:HEAD(L3)
      -6
:TAIL(L3)
      (5 3 1 0 3 -4)
HEAD | TAIL | | | | LIST
```

### The SEQ function

Item 2. PROCEDURES.. in the PRG/LIST menu contains the following functions that can be used to operate on lists.

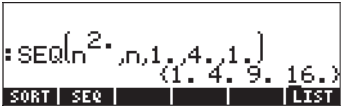


Functions REVLIST and SORT were introduced earlier as part of the MTH/LIST menu. Functions DOLIST, DOSUBS, NSUB, ENDSUB, and STREAM, are designed as programming functions for operating lists in RPN mode. Function

SEQ is useful to produce a list of values given a particular expression and is described in more detail here.

The SEQ function takes as arguments an expression in terms of an index, the name of the index, and starting, ending, and increment values for the index, and returns a list consisting of the evaluation of the expression for all possible values of the index. The general form of the function is SEQ(expression, index, start, end, increment).

In the following example, in ALG mode, we identify *expression* =  $n^2$ , *index* = *n*, *start* = 1, *end* = 4, and *increment* = 1:



The list produced corresponds to the values  $\{1^2, 2^2, 3^2, 4^2\}$ . In RPN mode, you can list the different arguments of the function as follows:



before applying function SEQ.

### The MAP function

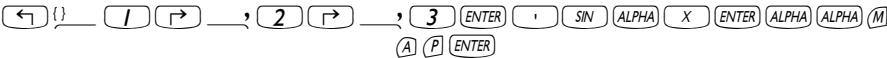
The MAP function, available through the command catalog ( $\boxed{\text{CAT}}$ ), takes as arguments a list of numbers and a function  $f(X)$  or a program of the form  $\ll \rightarrow$  a ...  $\gg$ , and produces a list consisting of the application of that function or program to the list of numbers. For example, the following call to function MAP applies the function  $\text{SIN}(X)$  to the list  $\{1,2,3\}$ :



In ALG mode, the syntax is:



In RPN mode, the syntax is:





In both cases, you can either type out the MAP command (as in the examples above) or select the command from the CAT menu.

The following call to function MAP uses a program instead of a function as second argument:

```
: MAP({0,1,2},{x → x 'x  
^2-1' »})  
(-1 0 3)  
CASCH/HELP
```

## Defining functions that use lists

In Chapter 3 we introduced the use of the DEFINE function (  DEF ) to create functions of real numbers with one or more arguments. A function defined with DEF can also be used with list arguments, except that, any function incorporating an addition must use the ADD operator rather than the plus sign (  ). For example, if we define the function  $F(X,Y) = (X-5)*(Y-2)$ , shown here in ALG mode:

```
: DEFINE('F(X,Y)=(X-5.)*(Y-2)  
NOVAL  
F | L2 | L1
```




we can use lists (e.g., variables L1 and L2, defined earlier in this Chapter) to evaluate the function, resulting in:

```
: DEFINE('F(X,Y)=(X-5.)*(Y-2)  
NOVAL  
: F(L1,L2)  
(20. 0. 2. -3.)  
F | L2 | L1
```

Since the function statement includes no additions, the application of the function to list arguments is straightforward. However, if we define the function  $G(X,Y) = (X+3)*Y$ , an attempt to evaluate this function with list arguments (L1, L2) will fail:

```
: DEFINE('G(X,Y)=(X+3.)*Y')  
NOVAL  
G(L1,L2)  
G | F | L2 | L1
```

```
: DE  
: G(L1,L2)  
"Invalid Dimension"  
G | F | L2 | L1
```

To fix this problem we can edit the contents of variable  , which we can list in the stack by using (   ),

```

:DEFINE('G(X,Y)=(X+3.)*Y')
      NOVAL
:G(L1,L2)
  "Invalid Dimension"
  « → X Y '(X+3.)*Y' »

```

G	F	L2	L1	
---	---	----	----	--

to replace the plus sign (+) with ADD:

```

:G(L1,L2)
  "Invalid Dimension"
  « → X Y '(X+3.)*Y' »
  « → X Y '(X ADD 3.)*Y' »
  Y' »
  « → X Y '(X ADD 3.)*Y' »
  »

```

←SKIP←SKIP←←DEL DEL←DEL L Ins				
-------------------------------	--	--	--	--

Next, we store the edited expression into variable ■■■:

```

: « → X Y '(X ADD 3.)*Y' »
Y' »
« → X Y '(X ADD 3.)*Y' »
»
:ANS(1.)▶G
« → X Y '(X ADD 3.)*Y' »
»

```

G	F	L2	L1	
---	---	----	----	--

Evaluating G(L1,L2) now produces the following result:

```

:G(L1,L2)
      (-12. 10. 6. 35.)

```

G	F	L2	L1	
---	---	----	----	--

As an alternative, you can define the function with ADD rather than the plus sign (+), from the start, i.e., use `DEFINE('G(X,Y)=(X ADD 3)*Y')` :

```

:DEFINE('G(X,Y)=(X ADD 3)*Y')
      NOVAL
:G(L1,L2)
      (-12. 10. 6. 35.)

```

G	F	L2	L1	
---	---	----	----	--

You can also define the function as  $G(X,Y) = (X-3)*Y$ .

# Applications of lists

This section shows a couple of applications of lists to the calculation of statistics of a sample. By a sample we understand a list of values, say,  $\{s_1, s_2, \dots, s_n\}$ .

Suppose that the sample of interest is the list

$\langle 1, 5, 3, 1, 2, 1, 3, 4, 2, 1 \rangle$

and that we store it into a variable called S (The screen shot below shows this action in ALG mode, however, the procedure in RPN mode is very similar. Just keep in mind that in RPN mode you place the arguments of functions in the stack before activating the function):

```
: {1. 5. 3. 1. 2. 1. 3. 4. 2. 1.}
  {1. 5. 3. 1. 2. 1. 3. 4. 2. 1.}
+SKIP|SKIP+|+DEL|DEL+|DEL L|INS|
```

## Harmonic mean of a list

This is a small enough sample that we can count on the screen the number of elements ( $n=10$ ). For a larger list, we can use function SIZE to obtain that number, e.g.,

```
: {1. 5. 3. 1. 2. 1. 3. 4.}
{1. 5. 3. 1. 2. 1. 3. 4. 2.}
: SIZE(S)
10.
S | G | F | L2 | L1 |
```

Suppose that we want to calculate the harmonic mean of the sample, defined as

$$s_h = \frac{1}{\frac{1}{n} \sum_{k=1}^n \frac{1}{s_n}} = \frac{1}{\frac{1}{n} \left( \frac{1}{s_1} + \frac{1}{s_2} + \dots + \frac{1}{s_n} \right)}$$

To calculate this value we can follow this procedure:

- 1. Apply function INV () to list S:

```
: {1. 5. 3. 1. 2. 1. 3. 4.}
{1. 5. 3. 1. 2. 1. 3. 4. 2.}
: SIZE(S)
10.
: INV(S)
{1. .2 .3333333333333333 1.}
S | G | F | L2 | L1 |
```

- 2. Apply function ΣLIST() to the resulting list in 1.

```

{1. 5. 3. 1. 2. 1. 3. 4. 2}
:SIZE(S)
10.
:INV(S)
{1. 2. 333333333333 1.}
:ΣLIST(ANS(1.))
6.11666666666
◀LIST|ELIST|TLIST|SORT|REVLI|ADD

```

3. Divide the result above by  $n = 10$ :

```

:INV(S)
{1. 2. 333333333333 1.}
:ΣLIST(ANS(1.))
6.11666666666
:ANS(1.)
10.
.611666666666
S | G | F | L2 | L1 |

```

4. Apply the INV() function to the latest result:

```

:ΣLIST(ANS(1.))
6.11666666666
:ANS(1.)
10.
.611666666666
:INV(ANS(1.))
1.6348773842
S | G | F | L2 | L1 |

```

Thus, the harmonic mean of list S is  $s_h = 1.6348\dots$

## Geometric mean of a list

The geometric mean of a sample is defined as

$$x_g = \sqrt[n]{\prod_{k=1}^n x_k} = \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$$

To find the geometric mean of the list stored in S, we can use the following procedure:

1. Apply function TLIST() to list S:

```

:ANS(1.)
10.
.611666666666
:INV(ANS(1.))
1.6348773842
:TLIST(S)
720.
S | G | F | L2 | L1 |

```

2. Apply function XROOT(x,y), i.e., keystrokes  $\boxed{\rightarrow} \boxed{\sqrt[y]{x}}$ , to the result in 1:



```

10.
: INV(ANS(1.)) .611666666666
: TLIST(S) 1.6348773842
: XROOT(ANS(1.),10) 720.
S G F L2 L1

```

```

: INV(ANS(1.)) .611666666666
: TLIST(S) 1.6348773842
: ANS(1.) 720.
: XROOT(ANS(1.),10) 1.00320315402
S G F L2 L1

```

Thus, the geometric mean of list S is  $s_g = 1.003203\dots$

## Weighted average

Suppose that the data in list S, defined above, namely:

$$S = \langle 1, 5, 3, 1, 2, 1, 3, 4, 2, 1 \rangle$$

is affected by the weights,

$$W = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle$$

If we define the weight list as  $W = \{w_1, w_2, \dots, w_n\}$ , we notice that the  $k$ -th element in list W, above, can be defined by  $w_k = k$ . Thus we can use function SEQ to generate this list, and then store it into variable **W** as follows:

```

: ANS(1.) 720.
: XROOT(ANS(1.),10) 1.00320315402
: SEQ(k,k,1,10,1) (1. 2. 3. 4. 5. 6. 7. 8. 9)
: ANS(1.) W (1. 2. 3. 4. 5. 6. 7. 8. 9)
W

```

Given the data list  $\{s_1, s_2, \dots, s_n\}$ , and the weight list  $\{w_1, w_2, \dots, w_n\}$ , the weighted average of the data in S is defined as

$$s_w = \frac{\sum_{k=1}^n w_k \cdot s_k}{\sum_{k=1}^n w_k}$$

To calculate the weighted average of the data in list S with the weights in list W, we can use the following steps:

1. Multiply lists S and W:

```

: SEQ(k,k,1,10,1) 1.00320315402
: ANS(1.) W (1. 2. 3. 4. 5. 6. 7. 8. 9)
: ANS(1.) W (1. 2. 3. 4. 5. 6. 7. 8. 9)
: S*W (1. 10. 9. 4. 10. 6. 21. )
W S G F L2 L1

```

2. Use function  $\Sigma$ LIST in this result to calculate the numerator of  $s_w$ :

```

(1. 2. 3. 4. 5. 6. 7. 8. 9)
:ANS(1.)▶W
(1. 2. 3. 4. 5. 6. 7. 8. 9)
:S.W
(1. 10. 9. 4. 10. 6. 21. ▶
:ΣLIST(ANS(1.))
121.
ΣLIST ΣLIST nLIST SORT REVL1 ADD

```

3. Use function  $\Sigma$ LIST, once more, to calculate the denominator of  $s_w$ :

```

(1. 2. 3. 4. 5. 6. 7. 8. 9)
:S.W
(1. 10. 9. 4. 10. 6. 21. ▶
:ΣLIST(ANS(1.))
121.
:ΣLIST(W)
55.
W S G F L2 L1

```

4. Use the expression  $\text{ANS}(2)/\text{ANS}(1)$  to calculate the weighted average:

```

:ΣLIST(ANS(1.))
121.
:ΣLIST(W)
55.
:ANS(2.)
:ANS(1.)
2.2
W S G F L2 L1

```

Thus, the weighted average of list S with weights in list W is  $s_w = 2.2$ .

**Note:**  $\text{ANS}(1)$  refers to the most recent result (55), while  $\text{ANS}(2)$  refers to the previous to last result (121).

## Statistics of grouped data

Grouped data is typically given by a table showing the frequency ( $w$ ) of data in data classes or bins. Each class or bin is represented by a class mark ( $s$ ), typically the midpoint of the class. An example of grouped data is shown next:

	Class	Frequency
Class	mark	count
boundaries	$s_k$	$w_k$
0 - 2	1	5
2 - 4	3	12
4 - 6	5	18
6 - 8	7	1
8 - 10	9	3

The class mark data can be stored in variable S, while the frequency count can be stored in variable W, as follows:

```

:SEQ(2-k-1,k,1,5,1)
      (1 3 5 7 9)
:ANS(1)►S
      (1 3 5 7 9)
:(5 12 18 1 3)►W
      (5 12 18 1 3)
W | S | G | F | L2 | L1

```

Given the list of class marks  $S = \{s_1, s_2, \dots, s_n\}$ , and the list of frequency counts  $W = \{w_1, w_2, \dots, w_n\}$ , the weighted average of the data in S with weights W represents the mean value of the grouped data, that we call  $\bar{s}$ , in this context:

$$\bar{s} = \frac{\sum_{k=1}^n w_k \cdot s_k}{\sum_{k=1}^n w_k} = \frac{\sum_{k=1}^n w_k \cdot s_k}{N}$$

where  $N = \sum_{k=1}^n w_k$  represents the total frequency count.

The mean value for the data in lists S and W, therefore, can be calculated using the procedure outlined above for the weighted average, i.e.,

```

:ΣLIST(W*S)
:ΣLIST(W)
      55
      13
:÷NUM(ANS(1))
      4.23076923077
W | S | G | F | L2 | L1

```

We'll store this value into a variable called XBAR:

```

:ΣLIST(W)
      55
      13
:÷NUM(ANS(1))
      4.23076923077
:ANS(1)►XBAR
      4.23076923077
XBAR | W | S | G | F | L2

```

The variance of this grouped data is defined as

$$V = \frac{\sum_{k=1}^n w_k \cdot (s_k - \bar{s})^2}{\sum_{k=1}^n w_k} = \frac{\sum_{k=1}^n w_k \cdot (s_k - \bar{s})^2}{N}$$

To calculate this last result, we can use the following:

```

: ANS(1) ► XBAR
      4.23076923077
: ΣLIST(W(S-XBAR)^2)
      156.923076923
: ΣLIST(W)
      39
XBAR | W | S | G | F | L2

```

```

: ANS(1) ► XBAR
      4.23076923077
: ΣLIST(W(S-XBAR)^2)
      156.923076923
: ΣLIST(W)
      39
XBAR | W | S | G | F | L2

```

The standard deviation of the grouped data is the square root of the variance:

```

: ΣLIST(W)
      39
: ANS(2)
: ANS(1)
      4.02366863905
: √ANS(1)
      2.00590843237
XBAR | W | S | G | F | L2

```