Calcul formel et Mathématiques avec la HP49G en mode algébrique

Renée De Graeve
Maître de Conférence à Grenoble I

Remerciements

Je remercie:

- Bernard Parisse pour ses précieux conseils et ses remarques sur ce texte,
- Sylvain Daudé pour sa relecture,
- Jean Tavenas pour l'intérêt porté à l'achèvement de ce guide,
- les élèves de Terminale du lycée Notre-Dame des Victoires de Voiron, ainsi que leur professeur Jean Marc Paucod, pour leur participation au test du sujet de bac avec la HP49.

This document is also available at the following Internet address:

`http://www-fourier.ujf-grenoble.fr/~degraeve/usflan.pdf`

Foreword

Sometimes, I am asked this question: why put symbolic manipulation capabilities into a calculator, while specialized computer software for this is now either cheap, or available for free?

In my opinion, a calculator is the most effective way to integrate calculation aids with the teaching of mathematics, because it is both easy to carry out with you and to use in a classroom.

However, using a symbolic manipulation software package is not as simple as its interface could suggest... Therefore, having an adequate documentation for it is important. The HP49G user guide describes the computer algebra system very briefly, so this manual is its essential complement:

it describes the HP49G from the point of view of someone that "wants to do maths".

In fact, readers interested in maths can read only this book as well, because the author starts with an introduction to the calculator, describes in more detail the computer algebra system commands arranged by purpose (the index allows the reader to find all commands in alphabetical order), and then focuses on programming the calculator in algebraic mode.

Each command is demonstrated by an example, and many of them are leveraged to solve a "baccalaureat" problem. The programming section has several programs, arithmetic programs in particular.

Briefly, this is the manual I should have written if I had enough patience! I thank Renée for carrying it out.

Bernard Parisse
Maître de Conférences à l'Université de Grenoble I
Author of the Computer Algebra System of the HP49G

Getting started

0.1 Introduction

0.1.1 Turning on the calculator

Press the `ON` key.
When the calculator is turned on, the same key exits an application: it acts as either `EXIT` or `CANCEL`.
To turn off the calculator, press `red-shift` and then `ON`.
If the calculator does not respond in despite of several `ON` (`CANCEL`), press both `ON` and `F3` simultaneously to reinitialize it.

0.1.2 What am I looking at?

From top to bottom, you can see:
1. the screen

    1.a the calculator status

    1.b the calculation history

    1.c a menu containing some commands

    2. the keyboard

1. The screen:

    1.a The calculator status describes the current calculator modes:

- `RAD` if the calculator is working in radians, `DEG` if it is working in degrees.
- `XYZ` shows that rectangular coordinates are in use.
- `HEX` shows that binary integers prefixed by `#` are displayed in base 16.
- `R` if the calculator is in REAL mode, `C` if it is in COMPLEX mode.
- `=` if the calculator is in EXACT mode (symbolic calculations), $\sim$ if the calculator is in APPROXIMATE mode (numeric calculations).
- `'X'` denotes the name of the current variable stored in `VX`: usually it is `'X'`.
- `ALG` if the calculator is in ALGEBRAIC mode, `RPN` if it is in RPN mode.
- `{HOME}` or `{HOME ESSAI}` to show the name of the current directory (for example, the main directory HOME or the ESSAI subdirectory).

1.b The calculation history:
General principle: on the screen, the input expression (prefixed by `:`) is displayed left-justified, and the result is right-justified.

1.c The menu:
Menu commands are accessed using the following keys:
`F1 F2 F3 F4 F5 F6`.
When the menu has more than 6 commands, press the `NXT` key to display the next portion of the menu. The menu can also contain subdirectories (which, in turn, contain a set of commands): they can be recognized because their menu item has a small bar across the upper-left corner. To execute a menu command, simply press the corresponding `Fi` key.

2. The keyboard:
You should find:

- the `ON` key, to turn on the calculator, and to interrupt a calculation while it is in progress. To turn off the calculator, press `red-shift` and then `ON`.
- two "shift" keys, one blue and one red; they allow a single key to have more than one function.
- the `ALPHA` key, to enter alphabetical characters (uppercase by default). To keep alphabetic mode active for more than one subsequent keystroke, it is necessary to press `ALPHA` twice. To exit this mode, press `ALPHA`

again. To toggle between uppercase and lowercase alphabetic entry modes, press `blue-shift ALPHA` while the calculator is in alphabetic entry mode.

- the `ENTER` key; it either enters or confirms a command.
- four arrow keys (left, right, up, down); they move the cursor when you are either in the editor or in a command list.

0.2 Calculator modes

The calculator can work in several modes.
You can choose: -algebraic or reverse poland notation mode (`ALG` or `RPN`)
-real or complex mode (`R` or `C`)
-exact or approximate mode (= or $\sim$)
-immediate or step-by-step mode...
WARNING: this book assumes that the calculator is in algebraic, real, exact, immediate mode (`R = ALG`).

Type: `CASCFG` (Computer Algebra System ConFiG) to set the calculator in real, exact, immediate mode. While working, you can type `CASCFG` to restore this configuration (the calculator automatically changes mode -asking you for a permission- when it is appropriate!).

CHECK
Check now that your calculator is indeed in algebraic, real, exact mode. In order to do this:

press `MODE` to check that the operating mode really is `algebraic`; if it is not, select `algebraic` either with the `choos` menu item, or pressing the $^{+}/_{-}$ key.

While you are in the `MODE` screen, activate the `cas` menu item and check that neither `numeric`, nor `approx`, nor `complex` are enabled. If one ore more of these modes are enabled, disable them all using the `chk` menu item.
Notice: for pedagogic applications, it is often interesting to enable the `step/step` mode, to make the calculator execute its calculations step by step.

Now select the `ok` menu item to confirm all changes made in `cas`, and then `ok` again to confirm the choices you made in `MODE`.

Now you are in algebraic, real, exact mode.
WARNING:
THIS BOOK ASSUMES THAT THE CALCULATOR IS IN THIS MODE.

You are now in the HOME directory.

You can simply type in the calculation you want executed, for example: $1 + 1$, and then press ENTER

The result is displayed (right justified), and the input expression $1 + 1$, preceded by :, goes up in the history area (left justified).

It is possible to copy this expression in the command line by pressing the HIST key (the up arrow allows you to select the expression to copy, and the echo menu item copies and simplifies it).

It is also possible to reuse the last result (denoted ANS(1)) using the ANS (red-shift ENTER) key, as well as previous results (denoted ANS(2)) and so on.

You can do both exact calculations and approximate ones; for example:

$\sqrt{2}$ followed by ENTER does not evaluate $\sqrt{2}$ and keeps the result exact, but the same expression followed by red-shift ENTER ($\rightarrow$NUM) returns the approximate value of $\sqrt{2}$ with 12 significant digits, keeping the calculator in exact mode.

Of course, if you want to do only numeric calculations, you can enable approx mode (MODE key, and then cas menu item); in this mode, the ENTER key does the calculation numerically, evaluating both constants and variables.

0.3 Notation

In this book, the four arrow keys are represented by the following four triangles:

$$\triangle \ \triangleleft \ \triangleright \ \triangledown$$

The delete arrow (deletion of the character to the left of the cursor) is represented by:

$$\Leftarrow$$

The red arrow over the 0 (zero) key is represented by:

$$\rightarrow$$

The STO key is represented, in a program, by:

STO $\triangleright$ or $\triangleright$

The carriage return (in red, over the decimal point key), is represented by:

$$\hookleftarrow$$

0.4 Flags

The vary majority of commands takes system flags into account.
Each flag has its own unique number, and has a default value. If you want to change the value of a flag, you can do it by pressing the `MODE` key, then `F1` to select the `flags` menu item and enter the flag management screen.

When you toggle the flag you want changed, its new function appears on the screen.

If you know in advance the flag number, you can also change its value with the `SF` and `CF` commands.

For example, to change flag number −117 (that is, the flag controlling the display of menus), you type:
`SF(-117)` (most menus are now displayed across the bottom of the screen, instead of using pop-up command lists). After this command:
`FS?(-117)` returns `1.` and `FC?(-117)` is `0.`
To have most menus displayed using pop-up command lists again, you type:
`CF(-117)` (`FS?(-117)` is now `0.` and `FC?(-117)` is `1.`).

Chapter 1

Important keys

1.1 The APPS key

This key, when pressed, displays a list of all calculator's application.

1.1.1 Plot functions

This command list has the following items:
`Equation entry`. This item acts the same as the key sequence `blue-shift F1 (Y=3D)`.
`Plot window`. This item acts the same as the key sequence `blue-shift F2 (WIN)`.
`Graph display`. This item acts the same as the key sequence `blue-shift F3 (GRAPH)`.
`Plot setup`. This item acts the same as the key sequence `blue-shift F4 (2D/3D)`.
`Table setup`. This item acts the same as the key sequence `blue-shift F5 (TBLSET)`.
`Table display`. This item acts the same as the key sequence `blue-shift F6 (TABLE)`.
For more information, refer to chapter 3.

1.1.2 I/O functions

This list contains the commands that allow your calculator to interface with a computer.

For example, the fifth item is: `Transfer`.

If you press 5, and then `ok`, the calculator opens the `Transfer` window and displays:
Port : Wire
Type : Kermit (or XModem)
This window allows you to transfer a file interactively. You can do the same thing from the command line, too; for example, these are the steps you should follow to use the Linux `kermit` program:
-Connect both the calculator and the computer to the serial link cable.
-On the computer, type:
`kermit`
and then `serv`
-On the `HP49G` type:
`SEND('NOM')`
to copy the `NOM` variable from your `HP49G` to your computer.
-Or else,
On the `HP49G` type:
`KGET('NOM')`
to copy the `NOM` variable from your computer to your `HP49G`.

1.1.3 Constants library

This item displays a list of 40 physical constants.
These constants are denoted by their symbol and either their name or their value (if the `value` menu item is selected).
They are followed by their measurement units, if the `unit` menu item is selected.
They can also be copied on the command line, by pressing the `->stk` menu key.

1.1.4 Numeric solver

This item acts the same as the sequence of keys: `red-shift 7 (NUM.SLV)`.

1.1.5 Time & date

This item acts the same as the sequence of key: `red-shift 9 (TIME)`.

1.1.6 Equation writer

This item acts the same as the `EQW` key.
See section 2.1 for more details.

1.1.7 File manager

This item acts the same as the sequence of keys: `blue-shift APPS (FILES)`.
See section 2.5 for more details.

1.1.8 Matrix writer

This item acts the same as the sequence of keys: `blue-shift EQW (MTRW)`.
See section 2.2 for more details.

1.1.9 Text editor

This item opens the command line: notice that it is possible to write on more than one line (by pressing `red` − `shift` • (↩) to open a new line).

1.1.10 Math menu

This item acts the same as the sequence of keys: `blue-shift SYMB (MTH)`.

1.1.11 CAS menu

This item opens a command list with the following entries:
`1.ARITHMETIC` corresponding to the `blue-shift 1 (ARIT)` menu
`2.ALGEBRA` corresponding to the `red-shift 4 (ALG)` menu
`3.COMPLEX` corresponding to the `red-shift 1 (CMPLX)` menu
`4.CALCULUS` corresponding to the `blue-shift 4 (CALC)` menu
`5.EXP&LN` corresponding to the `blue-shift 8 (EXP&LN)` menu
`6.SYMBOLIC SOLVER` corresponding to the `blue-shift 7 (S.SLV)` menu
`7.MATRICES` corresponding to the `blue-shift 5 (MATRICES)` menu
`8.CONVERT` corresponding to the `blue-shift 6 (CONVERT)` menu
`9.TRIGONOMETRIC` corresponding to the `red-shift 8 (TRIG)` menu
Refer to chapter 4 for more information.

1.2 The MODE key

This key allows you to tune the operating mode of your calculator: `Algebraic` or `RPN` mode, to examine and change the calculator's `flags` (`F1` key), to tune the `cas` (`F3` key), and to change the way your calculator displays data on the screen with `disp` (`F4` key).

For example (see also page 9) the `flag -117` can be either:
`choose boxes` to have the calculator display its menus using popup windows
or
`soft menu` to have the calculator display its menus across the bottom of the screen.

1.3 The TOOL key

This key displays a menu containing:
`edit`, to edit the first line of the history (or the highlighted line).
`view`, to view the first line of the history (or the highlighted line).
`rcl`, the same as the key sequence `blue` – `shift` `STO` ▷ (`RCL`) (see page 25).
`sto` ▷ the same as the key `STO`▷.
`purge`, the same as the command `PURGE` (see page 25).
`clear`, to delete the current command line, leaving the cursor at the beginning of the line (this isn't the same as `CANCEL` that kills the current command line!). Beware, when the command line is not active, `clear` deletes the whole history; in this case, it is the same as `red` – `shift` ← (`CLEAR`).

1.4 The UNDO key (red-shift HIST)

This key is very useful, because it undoes the last command executed.

1.5 The VAR key

This key displays, across the bottom of the screen, a menu containing the names of all variables in the current directory (press `NXT` to view them all!). See section 2.4 for more information.

1.6 The EQW key

This key invokes the equation editor.
It can always be used, even in the matrix editor.
Also, from the equation editor it is possible to access the history (see 1.11).
For more information, see section 2.1.

1.7 The MTRW key (blue-shift EQW)

This key invokes the matrix editor, to enter a matrix. If you want to enter a vector instead, make sure that the `vect` menu option is selected.

To enter a matrix:
you enter the first line, then you move the cursor to the beginning of the next line; when you finish entering the following lines, the cursor automatically wraps around.

See section 2.2 for more details.

1.8 The SYMB key

This key opens a menu containing the most basic symbolic functions, divided by category.
The sub-menus contain the `cas` functions an high-school student usually needs. These functions, and many others, can be found in the corresponding `cas` menus, too.

Example:
The `SYMBOLIC ARITH MENU` is a portion of the `INTEGER` sub-menu of `ARITH (blue-shift 1)`.

1.9 The MTH (blue-shift SYMB) key

This key opens the mathematics functions menu.
There are:
The hyperbolic functions (sub-menu 4), like:
`SINH ASINH COSH ACOSH TANH ATANH`
The functions:

`EXPM(X)=EXP(X)-1 LNP1(X)=LN(X+1)`
and some functions used on real numbers (sub-menu 5), like:
`FLOOR(X)`, returning the largest integral value not grater than `X`.
`CEIL(X)`, returning the smallest integral value not less than `X`.
`RND(X,n)`, that rounds `X` to `n` decimal digits.
`TRNC(X,n)`, that truncates `X` to `n` decimal digits.

1.10 The UNITS (red-shift 6) key

The `UNITS` menu has 127 measurement units, divided by category.
To use a measurement unit, you must write the unit preceded by `_` (`red-shift -`).
You can convert from one measurement unit to another using the `CONVERT` function (you can find it into the `Tools` sub-menu of the `UNITS` menu).
Example:
Enter:
`CONVERT(12_cm,1_m)`
The result is:
`0.12_m`

1.11 The HIST key

This key allows you to access the history while you are typing a command. The same key allows also to access the history from inside either the equation editor, or the matrix editor.
It is important to know that the object copied from the history is both copied AND evaluated.
If you want to use a result again without reevaluating it, you must use:
`ANS(1)` or `ANS(2)`...(`red-shift ENTER (ANS(1)`).
If you want to reuse a command, you can also press `blue-shift HIST (CMD)`; this key gives you a list containing the last commmands you used.

Chapter 2

Data entry

2.1 The equation editor

2.1.1 Entering the equation writer

The `EQW` key (EQuationWriter) allows you to enter the equation editor at any time, from the command line. It is a very efficient editor useful to write, simplify and work on mathematical expressions.

While you are in the equation editor you can type expressions in, knowing that any operator you type always operates either on the expression next to the cursor, on on the selected expression.

You must not be worried about entering parentheses, you simply select!

You must imagine that mathematical expressions are like a tree, (not necessarily a binary one), and understand that the four arrow keys allow you to visit the tree in a natural way (the right and left keys allow you to go from a sub-tree to another, the up and down keys allow you to go up and down in the tree hierarchy, the "shifted" right and left keys allow you to accomplish various selections; see the second example on page 19).

2.1.2 How to select?

You can enter selection mode in two ways:

- The △ key enters selection mode and selects the expression element next to the cursor.

Then, you can enlarge the selection to the sub-tree located immediately to the right of your present selection, by pressing $\triangleright$.

- The $\triangleright$ key enters selection mode and selects the sub-tree next to the cursor.
- Warning: if you are entering a function with more than one argument, (like, for example, $\sum$, $\int$ or AND), the $\triangleright$ arrow allows you go from one argument to another. Therefore, you must always use the $\triangle$ key to start selection mode in this case (see 2.1.4).

Equation writer examples:

- Example 1
  Type:

  $$2 \; + \; X \; * \; 3 \; - \; X$$

  You obtain:

  $$2 + X \cdot 3 - X$$

  ENTER ENTER gives the following result:

  $$2 + 2 \cdot X$$

  Type:

  $$2 \; + \; X \; \triangleright \; * 3 \; - \; X$$

  You obtain:

  $$(2 + X) \cdot 3 - X$$

  ENTER ENTER gives the following result:

  $$6 + 2 \cdot X$$

  Type:

  $$2 \; + \; X \; \triangleright \; * 3 \; \triangle \; - \; X$$

  You obtain:

  $$(2 + X) \cdot (3 - X)$$

  ENTER ENTER gives the following result:

  $$-(X^2 - X - 6)$$

- Example 2
  If you want to enter:

  $$\mathtt{X}^2 - 3 \cdot \mathtt{X} + 1$$

  You type:

  $$\mathtt{X}\ \mathtt{y}^{\mathtt{x}}\ 2\ \triangleright\ -\ 3\ \mathtt{X}\ +\ 1$$

- Example 3
  You want to enter:

  $$\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5}$$

  Here, the root of the tree is a $+$, and there are four sub-trees; each sub-tree has a $\div$ as root, and has two leaves.

  First of all, you must press `EQW`, and then you enter the first sub-tree:

  $$1 \div 2$$

  Then, you select this sub-tree with

  $$\triangleright$$

  press

  $$+$$

  and enter the second sub-tree:

  $$1 \div 3$$

  Then, you select this sub-tree with

  $$\triangleright$$

  press

  $$+$$

  and enter the third sub-tree:

  $$1 \div 4$$

  Then, you select this sub-tree with

  $$\triangleright$$

press

$$+$$

and enter the fourth sub-tree:

$$1 \div 5$$

Last, press

$$\triangleright$$

again to select the last sub-tree you entered.

Now, the expression you want:

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5}$$

has been entered into the equation writer, and $\frac{1}{5}$ is selected.

Visit the tree to select:

$$\frac{1}{3} + \frac{1}{4}$$

You must press

$$\triangleleft$$

to select $\frac{1}{4}$; next,

$$\texttt{red} - \texttt{shift} \triangleleft$$

allows you to extend the selection to two contiguous sub-trees, in this example:

$$\frac{1}{3} + \frac{1}{4}$$

Notice that:

You can evaluate the selected portion of the expression with:

$$\texttt{red} - \texttt{shift SYMB (EVAL)}$$

You obtain:

$$\frac{1}{2} + \frac{7}{12} + \frac{1}{5}$$

Now, if you want to evaluate

$$\frac{1}{2} + \frac{1}{5}$$

first of all you must do a permutation in order to make $\frac{1}{2}$ and $\frac{1}{5}$ adjacent, pressing

$$\texttt{blue} - \texttt{shift} \triangleleft$$

thus exchanging the selected element with his left neighbor.

You obtain:

$$\frac{7}{12} + \frac{1}{2} + \frac{1}{5}$$

and $\frac{7}{12}$ is selected. Then,

$$\triangleright \texttt{red} - \texttt{shift} \triangleright$$

selects

$$\frac{1}{2} + \frac{1}{5}$$

You can now do EVAL again.

2.1.3 How to modify an expression

To replace the selection with another expression, you can directly type the new expression.
To remove a selection without deleting the selected expression, press:

$$\Leftarrow$$

To delete the selected expression, press:

$$\texttt{red} - \texttt{shift} \ \Leftarrow \ (\texttt{CLEAR})$$

To delete the unary operator at the root of the selected sub-tree, press:

$$\texttt{blue} - \texttt{shift} \ \Leftarrow \ (\texttt{DEL})$$

For example, to replace $\sin(expr)$ with $\cos(expr)$, you delete sin (selecting $\sin(expr)$ and pressing $\texttt{blue} - \texttt{shift} \ \Leftarrow$), then you enter: cos.

To delete a binary operatoy, you must use the edit menu option, make the correction in the command-line editor, and return to the equation writer with ENTER.

The HIST key (when used inside the equation editor) allows you to enter the history and to copy a history element into the equation writer with the echo menu option.

2.1.4 How to enter AND, $\int$ and $\sum$

To enter AND, you type it in alpha mode and you press $\triangleright$.

To enter the $\int$ symbol, you press:

$$\texttt{red} - \texttt{shift TAN}\ (\int)$$

To enter the $\sum$ symbol, you press:

$$\texttt{red} - \texttt{shift SIN}\ (\sum)$$

The cursor automatically moves where input is required, and you can move it using

$\triangleright$

The expressions you enter follow the selection rules explained above, but you must use $\triangle$ to enter selection mode.

Warning: do not use the index variable i in summations, because i denotes the complex number that solves the equation $x^2 + 1 = 0$.

You must also understand that $\sum$ is able to calculate symbolically the summations of rational fractions, and the hypergeometric series admitting a discrete primitive (starting from ROM version 1.11).

In numeric mode, $\sum$ performs approximate calculations (for example, $\sum_{k=0}^{4} \frac{1}{k!} = 2.70833333334$, instead of $1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} = \frac{65}{24}$) (the ! symbol can be typed in pressing alpha red $-$ shift 2).

2.1.5 Cursor mode

Cursor mode allows you to select a big expression fast:
press red-shift EQW (') to enter cursor mode (or press on the curs menu option). Next, use the arrows to enclose your selection in a box and press ENTER to select the box contents, or CANCEL to cancel the operation.

2.1.6 To view all

Pressing on the big menu option, you make the font used to display the expression either bigger or smaller: sometimes, making the font smaller allows you to view a big expression as a whole on the screen.

If this is not yet enough, select the view option of the TOOL menu.

2.2 The matrix writer

To invoke the matrix writer, press: `blue-shift EQW (MTRW)`.

You can then enter the elements belonging to the first line of the matrix by pressing `ENTER` after each entry (you can use the equation editor to write them, too!). Next, you move the cursor to the beginning of the second line with the arrow keys (the cursor automatically wraps around when you finish entering the following lines).

To enter a negative number, for example -2, enter $^{+}/_{-}$ 2.

If you want to enter a vector, make sure that the `vect` menu option is selected.

Notice that in `Algebraic` mode you must enter the matrix elements one at a time (pressing `ENTER` after each element), but in `RPN` mode you can write more than one element separating them with spaces; pressing `ENTER` then enters them all.

2.3 The text editor

This is the line that opens under the history to type a command in.
It is a full-fledged text editor, where you can: select an expression (with `BEGIN END`), either cut it (`CUT`) or copy it (`COPY`) into a buffer, and then paste it at the current cursor position (`PASTE`).

Notice also that these commands work in `EQW` and `MTRW`, too.

2.3.1 BEGIN END

Move the cursor on the first character of the text you want to select, then press: `red-shift APPS (BEGIN)`.

Then, move the cursor on the character that follows the last character to select, and press:
`red-shift MODE (END)`.

Your selection will be highlighted.

2.3.2 COPY

`red-shift VAR (COPY)` copies the selection into a buffer.

2.3.3 CUT

`red-shift STO (CUT)` copies the selection into a buffer, and deletes it from the command line.

2.3.4 PASTE

`red-shift NXT (PASTE)` pastes the contents of the buffer at the current cursor position (you must have previously done either `COPY`, or `CUT`, to put something into the buffer).

2.4 Variables

You can store objects into variables, and reference them using the variable name.

Be sure to notice the difference between `A` et `'A'` :
`A` is evaluated (it denotes the execution of variable contents), while `'A'` is not (it denotes the variable name).

For example:
`STO(B,'A')`: stores the contents of `B` into `A`.
`STO('B','A')`: means that `B` and `A` will have the same contents from now on.
`VAR` displays a menu listing all variables you have created in the current directory, as well as all its subdirectories (you can distinguish variables from subdirectories because subdirectories have a small bar across the upper-left corner of their menu item).

The `blue-shift APPS (FILES)` application displays the whole variable tree starting from `HOME`, as well as the archive memory, and greatly simplifies variable management.

2.4.1 STO

`STO` allows you to create a variable and to store an object into it.
WARNING: `STO` is prefixed if you type it in alpha mode, and is infixed if you

use the STO key. (from now on, this key will be denoted by either STO▷ or ▷).

Examples:
Type:
STO(1,'A')
or
use the STO▷ key, displayed on screen with ▷ :
to enter:
1 STO▷ A (1 ▷ A).

Notice that, in the latter case, you don't put ' ' around A.

The variable A is created, and it contains 1.

Enter:
≪ 12 ≫ STO▷ P
P is a varialbe containing the program ≪ 12 ≫ , that displays 12.

2.4.2 RCL

RCL takes a variable name, surrounded by ' , as argument, and displays the variable contents.

To recall the contents of a variable, entering the variable name is enough, UNLESS the variable contains a program (because, in the latter case, the program is executed).

In the last examples:
A displays 1 and P displays 12
but:
RCL('A') displays 1 and RCL('P') displays ≪ 12 ≫ .

2.4.3 PURGE

PURGE allows you to delete a variable and its contents.
You can find PURGE in the TOOL menu.

Example:
PURGE('A')

2.4.4 Predefined variables

The name of the current symbolic variable is stored in `VX` (and it will usually be `X`), therefore you should either not use `X` as an ordinary variable, or purge `X` before doing symbolic calculations.

`EPS` holds the value of epsilon used by the `EPSX0` command (see 4.20.1).

`EQ` holds the last equation you plotted.

`MATRIX` holds the last matrix used as argument to either `JORDAN, EGV` or `EGVL`.

`MODULO` holds the value of p when you do symbolic calculations in the $Z/p.Z$ ring.

`PERIOD` holds the period of the function of which you want to calculate the Fourier coefficients (see 4.7.16).

`PRIMIT` holds the antiderivative of the last function you asked the calculator to integrate.

`REALASSUME` holds the names of the symbolic variables you want the calculator assume to be reals (by default, `X, t` and all auxiliary integration variables used).

`SYSTEM` holds the last system of equations used as argument to either `rref` or `RREF`, if the system has at least a parameter.

2.5 Directories

At the beginning, you only have the `HOME` directory; it is the ancestor of any other directory you will create in the future.

2.5.1 Creating a directory

Press `blue-shift APPS (FILES)` to display the tree structure of your directories.

Select the directory you want to be the parent directory (for example HOME) and press ok.

A menu containing edit copy move... is displayed; press NXT and select new (new variable or directory) with F3.

Do not fill the Object field, but fill Name instead (to do this, simply type the name you chose, and then press ok of the menu.

Then, select Directory with F3 (chk), and press ok of the menu.

Last, press CANCEL to return in HOME.

Check, by pressing VAR, that your directory has actually been created.

You can also create a directory with the CRDIR command.

You make the parent directory current, and then type:
CRDIR('NOMREP')
to create a subdirectory named NOMREP.

2.5.2 Working in a directory

Working in a directory is simple: simply press VAR to display the subdirectory names in the menu area, and then open the subdirectory you want by pressing on the Fi corresponding to its name, followed by ENTER.

To climb up in the directory tree, press:
blue-shift VAR (UPDIR)

2.5.3 Deleting, renaming, moving a directory

Press blue-shift APPS (FILES) to display your directory tree.
Select the directory you want to delete, rename, move, and press the ok menu key.

A menu containing edit copy move...purge rename... is displayed.

purge deletes the directory, if it is empty.
rename gives the directory a new name.

`copy` copies the directory (the arrow keys are used to indicate the destination, and `ok` confirms).

`move` moves the directory (the arrow keys are used to indicate the destination, and `ok` confirms).

Chapter 3

Plotting graphs

3.1 Plot windows

3.1.1 Equation entry

This window is activated with the following sequence of keys:
`blue-shift F1 (Y=)`. It allows you to define the equation to be plotted.

3.1.2 Plot window

This window is activated with the following sequence of keys:
`blue-shift F2 (WIN)`.
It allows you to define the plot window and to enter the lower and upper boundaries of the independent plot variable.
If boundaries are set to `Default`, they are assumed to be equal to the horizontal size of the plot window.
To reset a parameter to `Default`, you must press `NXT`, and then press the `reset` menu key.

3.1.3 Graph display

This window is activated with the following sequence of keys:
`blue-shift F3 (GRAPH)`.
It allows you to draw the plot when you have set all its parameters.

3.1.4 Plot setup

This window is activated with the following sequence of keys:
`blue-shift F4 (2D/3D)`.
It allows you to choose the plot type, the equation to be plotted and the plot variables.

3.1.5 Table setup

This window is activated with the following sequence of keys:
`blue-shift F5 (TBLSET)`.
It allows you to initialize a table.

3.1.6 Table display

This window is activated with the following sequence of keys:
`blue-shift F6 (TABLE)`.
It displays the table you initialized with `TBLSET`.

3.2 Plot setup

3.2.1 Plot type

You can select the plot type using the `choos` menu key of the `PLOT SETUP (blue-shift F4 (2D/3D))` window.

Here, the most common plot types will be described, such as:

`Function` to plot a function in cartesian coordinates.

`Polar` to plot a function in polar coordinates.

`Parametric` to plot a parametric function.

`Truth` to plot the solutions of an equation (the pixel at (x,y) is turned on iff `EQ` is true).

`Diff Eq` to plot the solutions of the differential equation $y' = f(x, y)$.
You can plot the solution satisfying $y(x_0) = y_0$ on the interval $[a, b]$.
To do this, put in `H-View` the values of a and b, then x_0 into `Init` and y_0 into `Init-Soln`.

The solution is plotted in two steps: first, set `Final` to b to plot the solution on $[x_0, = b]$, draw the plot, then set `Final` to a to plot the solution on $[a, x_0]$ and draw the plot again.

`Slopefield` to draw the slope field of the differential equation $y' = f(x, y)$.

`Fast3D` to plot a surface defined by $z = f(x, y)$.
The plot can be rotated using `NXT`, `TOOL` and the arrow $\triangle$ $\triangleleft$ $\triangleright$ $\triangledown$ keys, to have a good view of the surface.

3.2.2 The equation

You can enter the equation in many ways:
-you can store it into the `EQ` variable.
-you can enter it in the window opened by `blue-shift F1 (Y=)`.
-you can enter it in the `EQ` field of the `PLOT SETUP` window, opened by `blue-shift F4 (2D/3D)`.
-you can also use the `cas` function `PLOT`. It has a functions as argument, stores it into `EQ` and opens the `PLOT SETUP` window.

Notice that `EQ` can be a list of equations; in this case, all of them will be plotted on the same graphic.

You can also add en equation to the list of equations stored in `EQ`, with the aid of the `cas` function `PLOTADD`.

3.2.3 Independent variable and equation types

The equation type depends on the plot type you have selected and on the independent variable you chose.

Depending on this, you enter an equation of type:
`f(x)` to plot $y = f(x)$ in cartesian coordinates, if `x` is the independent variable and the plot type is `Function`.

`f(t)` to plot $r = f(t)$ in polar coordinates, if `t` is the independent variable and the plot type is `Polar`.

`x(t)` + `i.y(t)` to plot $(x = x(t), y = y(t))$ in parametric coordinates, if `t` is the independent variable and the plot type is `Parametric`.

`f(x,y) > 0` to highlight the corresponding portion of the x, y plane, if `x` and `y` are the independent variables and the plot type is `Truth`.

`f(t,y)` to plot the solutions of the differential equation $y' = f(t, y)$ if `t` is the independent variable, `y` is the solution variable and the plot type is `Diff Eq`.

`f(t,y)` to plot the slope field of the differential equation $y' = f(t, y)$, if `t` is the independent variable, `y` is the solution variable and the plot type is `Slopefield`.

`f(x,y)` to plot the surface defined by $z = f(x, y)$ if `x` and `y` are the independent variables and the plot type is `Fast3D`.

Sometimes, the name of the second independent variable can be changed; by default its name is `y`. This name is always tagged by `Depend`, even if it does correspond to an independent variable! Do not take the word `Depend` into account in this case.

3.3 Drawing the plot

Before drawing a plot, you must set up many parameters.
When you have set all parameters up, to draw a plot press on:
`erase draw` (if tou want to erase the last plot you made) or
`draw` (if you want yo keep the last plot you made).

Using the menu of one of the following windows:
`PLOT SETUP (blue-shift F4 (2D/3D))`
`PLOT (blue-shift F1 (Y=3D))`
`PLOT WINDOW (blue-shift F2 (WIN))`.

You can also press:
`blue-shift F3 (GRAPH)` to draw the new plot without erasing the previous one.

You can review the last plot you made by pressing on $\triangleleft$.

Chapter 4

Symbolic calculations

4.1 Integers (and Gauss integers)

In this chapter, all integers can be freely replaced by Gauss integers, as an argument for all functions described here.

4.1.1 Infinite-precision integers

The calculator can handle infinite-precision integers, for example:

100!

The symbol ! can be obtained either pressing `alpha` `red` − `shift` 2, or using `red-shift CAT (CHARS)`.

In the latter case, you select ! in `CHARS` (with the arrow keys), and then copy it into the command line using the `echo1` menu key.

Since the decimal representation of 100! is very long, you can view the result using the `TOOL` key, followed by the `view` menu key.

The `HIST` and the up arrow keys allow you to climb up through the history, and the `view` menu key allows you to review previous results.

4.1.2 `DEFINE`

Consider the following exercise:
Calculate the first six Fermat numbers $F_k = 2^{2^k} + 1$ for $k = 1..6$, and check

whether they are prime.

Type the expression:

$$2^{2^2}+1$$

to find 17, then invoke the `ISPRIME?()` command with `ANS(1)` as argument.

You can find this command in the `ARITH (blue-shift 1)` menu, submenu `1 INTEGER` (or you can type it in α mode).

The answer is `1.`, meaning `true`.

With the aid of the history, (`HIST`) you can copy the expression $2^{2^2}+1$ into the command line, and modify it to read as:

$$2^{2^3}+1$$

Otherwise, you can type the expression $2^{2^K}+1$ `STO FK`, then do `3 STO K`, and so on...

Otherwise, and it is the better choice, you can define the function `F(K)` with the aid of `DEF (blue-shift 2)`, entering:

$$\texttt{DEFINE}(\texttt{F}(\texttt{K})=2^{2^{\texttt{K}}}+1)$$

The result is `NOVAL` and `F` is added to the variables (press on `VAR` to check this).

For $K=5$ you enter:

$$\texttt{F}(5)$$

obtaining:

$$4294967297$$

You can factorize F_5 with `FACTOR` ; you find it in the `ALG (red-shift 4)` menu.

Type:

$$\texttt{FACTOR}(\texttt{F}(5))$$

You obtain:

$$641\cdot 6700417$$

For `F(6)` you find:

$$18446744073709551617$$

Factorizing the result with `FACTOR`, the result is:

$$274177\cdot 67280421310721$$

Notice the difference in notation between:

$$2.5 = \frac{5}{2}$$

and

$$2 \cdot 5 = 10$$

4.1.3 GCD

GCD denotes the greatest common divisor of two integers (or of two lists of integers with the same size).
Enter:

$$\texttt{GCD}(18, 15)$$

You obtain:

$$3$$

Enter:

$$\texttt{GCD}(\{18, 28\}, \{15, 21\})$$

You obtain:

$$\{3, 7\}$$

because $\texttt{GCD}(18, 15) = 3$ and $\texttt{GCD}(28, 21) = 7$.

4.1.4 LGCD

LGCD denotes the greatest common divisor of a list of integers.
Enter:

$$\texttt{LGCD}(\{18, 15, 21, 36\})$$

You obtain:

$$3$$

4.1.5 SIMP2

SIMP2 has two integers as arguments (or two lists of integers). These integers are assumed to represent a fraction: the first element of the list is the fraction's numerator, the second is the denominator. SIMP2 returns a list of two integers representing, under the same assumptions, the input fraction simplified.
Enter:

$$\texttt{SIMP2}(18, 15)$$

You obtain:

$$\{6, 5\}$$

Enter:

$$\texttt{SIMP2}(\{18, 28\}, \{15, 21\})$$

You obtain:

$$\{6, 5, 4, 3\}$$

4.1.6 LCM

LCM denotes the least common multiple of two integers (or of two lists of integers).
Enter:

$$\texttt{LCM}(18, 15)$$

You obtain:

$$90$$

4.1.7 FACTOR

FACTOR factorizes its argument into a product of prime factors.
Enter:

$$\texttt{FACTOR}(90)$$

You obtain:

$$2.3^2.5$$

4.1.8 FACTORS

FACTORS does the same, but the result is given as a list, containing the prime factors and their exponents.
Type:

$$\texttt{FACTORS}(90)$$

You obtain:

$$\{2, 1., 3, 2., 5, 1.\}$$

4.1.9 DIVIS

DIVIS returns a list containing all divisors of a given integer.
Type:

DIVIS(36)

You obtain:

$\{1, 3, 9, 2, 6, 18, 4, 12, 36\}$

4.1.10 IQUOT

IQUOT returns the integer quotient of the euclidean division of two integers.
Type:

IQUOT(148, 5)

You obtain:

29

4.1.11 IREMAINDER MOD

IREMAINDER returns the integer remainder of the euclidean division of two integers.
You type:

IREMAINDER(148, 5)

or

148 MOD 5

You obtain:

3

The difference between IREMAINDER and MOD is that the former works with both integers and Gauss integers.
Try:

IREMAINDER(148!, 5! + 2)

(you obtain ! with alpha red-shift 2).

4.1.12 IDIV2

IDIV2 returns a list containing the quotient and the remainder of the euclidean division between two integers, in that order.
Type:

IDIV2(148, 5)

You obtain:

$$\{29, 3\}$$

In step-by-step mode, the calculator shows the division process like it is taught at school.

4.1.13 ISPRIME?

`ISPRIME?(N)` returns `1.` (true) if `N` is pseudo-prime, and returns `0.` (false) if `N` is not prime.
Definition: For all integers less than 10^{14} pseudo-primality and primality are the same! ...beyond 10^{14} a pseudo-prime integer has a very high probability to be prime (see the Rabin algorithm in section 7.6).
Type:

```
ISPRIME?(13)
```

You obtain:

```
1.
```

Type:

```
ISPRIME?(14)
```

You obtain:

```
0.
```

4.1.14 NEXTPRIME

`NEXTPRIME(N)` returns the smallest pseudo-prime number greater than `N`.
Type:

```
NEXTPRIME(75)
```

You obtain:

```
79
```

4.1.15 PREVPRIME

`PREVPRIME(N)` returns the largest pseudo-prime number less than `N`.
Type:

```
PREVPRIME(75)
```

You obtain:

```
73
```

4.1.16 IEGCD

`IEGCD(A,B)` returns the extended GCD (Bézout identity) of two integers, that is, `IEGCD(A,B)` returns `{D,U,V}` so that `AU+BV=D` and `D=GCD(A,B)`.
Type:

$$\texttt{IEGCD}(48, 30)$$

You obtain:

$$\{6, 2, -3\}$$

In fact:

$$2 \cdot 48 + (-3) \cdot 30 = 6$$

4.1.17 IABCUV

`IABCUV(A,B,C)` returns `{U ,V}` so that `AU+BV=C`.
`C` must be a multiple of `GCD(A,B)` for a solution to exist.
Type:

$$\texttt{IABCUV}(48, 30, 18)$$

You obtain:

$$\{6, -9\}$$

4.1.18 ICHINREM

`ICHINREM([A,P],[B,Q])` returns a vector `[X, N]` so that:
`X=A (mod P)` and `X=B (mod Q)`.
The solution `X` exists if `P` and `Q` are mutually prime, and all solutions are congruent modulo `N` = `P` · `Q`
Example:
Solve:

$$\begin{cases} X = & 3 \pmod 5 \\ X = & 9 \pmod{13} \end{cases}$$

Type:

$$\texttt{ICHINREM}([3, 5], [9, 13])$$

You obtain:

$$[-147, 65]$$

that is, `X=-147 (mod 65)`

4.1.19 PA2B2

PA2B2 decomposes a prime integer p, congruent to 1 modulo 4, as follows: $p = a^2 + b^2$. The calculator returns the result as $a + b \cdot i$
Type:

PA2B2(17)

You obtain:

4 + i

that is, $17 = 4^2 + 1^2$

4.1.20 EULER

EULER denotes the Euler's totient function of an integer.
EULER(n) is the number of integers less than n and prime with n.
Type:

EULER(21)

you obtain:

12

In fact, the set:
E={2,4,5,7,8,10,11,13,15,16,17,19} is the set of integers less than 21 and prime with 21, and it has 12 elements.

4.2 Rationals

Type:

$$\frac{123}{12} + \frac{57}{21}$$

and then ENTER; the answer is:

$$\frac{363}{28}$$

with red-shift ENTER (→NUM) the answer is:

12.9642857143

If you mix both representations, for example:

$$\frac{1}{2} + 0.5$$

the calculator demands to enable `approx` mode to carry out the calculation; you should answer `yes` to obtain:

1.

Now, return to exact mode (`MODE cas` menu keys, and so on...).

4.2.1 PROPFRAC

`PROPFRAC(A/B)` rewrites the fraction $\frac{A}{B}$ as:

$$Q + \frac{R}{B} \quad with \quad 0 \leq R < B$$

Type:

`PROPFRAC(43 ÷ 12)`

You obtain:

$$3 + \frac{7}{12}$$

4.2.2 FXND

`FXND` has a fraction as argument, and returns a list containing the fraction's numerator and denominator simplified.
Type:

`FXND(42 ÷ 12)`

You obtain:

{7, 2}

4.2.3 SIMP2

`SIMP2` (see 4.1.5) has a list of two integers representing a fraction as argument and, like `FXND`, returns a list containing the fraction's numerator and denominator simplified.
Type:

`SIMP2({42, 12})`

You obtain:

{7, 2}

4.3 Reals

Type:

$$\texttt{EXP}(\pi * \sqrt{20})$$

followed by `ENTER`; the answer is:

$$\texttt{EXP}(2 * \sqrt{5} * \pi)$$

with `red-shift ENTER (→NUM)` the answer is:

$$1263794.7537$$

4.4 Complex numbers

Type:

$$(1 + 2.\texttt{i})^2$$

follower by `ENTER`.

If you aren't in `complex` mode, the calculator asks for a mode change: you should answer `yes` to obtain the answer:

$$-(3 - 4 \cdot \texttt{i})$$

Notice that this expression is not simplified beforehand (all results always put in evidence complex numbers with a positive real part in exact mode).

In the `red-shift 1 (CMPLX)` menu you will find the following functions, having a complex-valued expression as argument:
`ARG` returns the argument of its input.
`ABS` returns the modulo of its argument.
`CONJ` returns the conjugate of its argument.
`RE` returns the real part of its argument.
`IM` returns the imaginary part of its argument.
`NEG` returns the opposite of its argument.
`SIGN` returns the quotient between its argument and its argument's modulo.
For example:
Type:

$$\texttt{ARG}(3 + 4 \cdot \texttt{i})$$

you obtain:

$$\texttt{ATAN}(\frac{4}{3})$$

4.5 Algebraic expressions

4.5.1 FACTOR

FACTOR factorizes the expression given as argument.
Example:
Factorize

$$x^4+1$$

Type:

$$\texttt{FACTOR}(\texttt{X}^4+1)$$

You can find FACTOR in the ALG (red-shift 4) menu (or you can type it in α mode).
In real mode, the answer is:

$$(\texttt{X}^2+\sqrt{2}\cdot\texttt{X}+1)\cdot(\texttt{X}^2-\sqrt{2}\cdot\texttt{X}+1)$$

In complex mode (to enable this mode, press MODE, then the cas menu key, then check the complex field with chk and press ok ok) the answer is:

$$\frac{(2\cdot\texttt{X}+(1+\texttt{i})\cdot\sqrt{2})\cdot(2\cdot\texttt{X}-(1+\texttt{i})\cdot\sqrt{2})\cdot(2\cdot\texttt{X}+(1-\texttt{i})\cdot\sqrt{2})\cdot(2\cdot\texttt{X}-(1-\texttt{i})\cdot\sqrt{2})}{16}$$

4.5.2 EXPAND EVAL

EXPAND and EVAL have an expression as argument; they expand and simplify their input.
Example:
Doing EXPAND(ANS(1)), you obtain again

$$\texttt{X}^4+1.$$

4.5.3 SUBST

SUBST has two arguments: an expression depending on a parameter, and an equality (parameter=substitution value).
SUBST does the commanded substitution in the input expression, and returns the result.
Type:

$$\texttt{SUBST}(\texttt{A}^2+1,\texttt{A}=2)$$

You obtain:

$$2^2+1$$

4.5.4 PREVAL

PREVAL has three arguments: an expression (F(VX)) depending on the variable stored in VX and two expressions: A and B .
PREVAL evaluates F(B)-F(A).
PREVAL is useful to compute a definite integral given an antiderivative: you evaluate the antiderivative between the upper and lower limits of the integral.
Type:

$$\texttt{PREVAL}(\texttt{X}^2+\texttt{X},2,3)$$

You obtain:

$$12-6$$

4.6 Functions

4.6.1 DERVX

Let $f(x)$ be:

$$f(x)=\frac{x}{x^2-1}+\ln(\frac{x+1}{x-1})$$

Determine the derivative of f.

You can find DERVX in the menu:
CALC (blue-shift 4), sub-menu 1.DERIV. & INT..., third position (or you can type it in mode α).

Type either:

$$\texttt{DERVX}(\frac{\texttt{X}}{\texttt{X}^2-1}+\texttt{LN}(\frac{\texttt{X}+1}{\texttt{X}-1}))$$

or, if you have previosuly stored $f(x)$ into F:

$$\texttt{DERVX}(\texttt{F})$$

or, if you have defined $F(X)$ using DEFINE: (DEFINE(F(X) $=\frac{\texttt{X}}{\texttt{X}^2-1}+\texttt{LN}(\frac{\texttt{X}+1}{\texttt{X}-1})$))

$$\texttt{DERVX}(\texttt{F}(\texttt{X}))$$

The result is an involved expression that you can simplify by copying it into the command line ($\triangle$ ENTER ENTER)).
You obtain:

$$-\frac{3\cdot\texttt{X}^2-1}{\texttt{X}^4-2\cdot\texttt{X}^2+1}$$

4.6.2 DERIV

DERIV has two arguments: an expression (or a functions), and a variable (or a vector containing more than one variable name) (see multivariate functions, paragraph 4.16.1).
DERIV returns the derivative of the expression (or function) with respect to the given variable(s) (useful to calculate partial derivatives!).
Example:
Suppose you should calculate:

$$\frac{\partial(x \cdot y^2 \cdot z^3 + x \cdot y)}{\partial z}$$

Type:

$$\texttt{DERIV}(\texttt{X} \cdot \texttt{Y}^2 \cdot \texttt{Z}^3 + \texttt{X} \cdot \texttt{Y}\ ,\ \texttt{Z})$$

You obtain:

$$3 \cdot \texttt{X} \cdot \texttt{Y}^2 \cdot \texttt{Z}^2$$

4.6.3 INTVX

Let $f(x)$ be:

$$f(x) = \frac{x}{x^2 - 1} + \ln(\frac{x+1}{x-1})$$

Determine an antiderivative of f.
You can find INTVX in the CALC (blue-shift 4) menu, 1. DERIV. & INT... sub-menu, eighth position (or you can type it in α mode).
You type either:

$$\texttt{INTVX}(\frac{\texttt{X}}{\texttt{X}^2 - 1} + \texttt{LN}(\frac{\texttt{X}+1}{\texttt{X}-1}))$$

or, if you have previously stored $f(x)$ into F:

$$\texttt{INTVX}(\texttt{F})$$

or, if you have defined $F(X)$ using DEFINE (DEFINE(F(X) $= \frac{\texttt{X}}{\texttt{X}^2-1} + \texttt{LN}(\frac{\texttt{X}+1}{\texttt{X}-1})$):

$$\texttt{INTVX}(\texttt{F}(\texttt{X}))$$

You obtain:

$$\texttt{X} \cdot \texttt{LN}(\frac{\texttt{X}+1}{\texttt{X}-1}) + \frac{3}{2} \cdot \texttt{LN}(|\texttt{X}-1|) + \frac{3=}{2} \cdot \texttt{LN}(|\texttt{X}+1|)$$

Exercise 1
Calculate:

$$\int \frac{2}{x^6 + 2 \cdot x^4 + x^2}\, dx$$

Type:

$$\texttt{INTVX}(\frac{2}{\texttt{X}^6 + 2 \cdot \texttt{X}^4 + \texttt{X}^2})$$

You obtain:

$$-3 \cdot \texttt{ATAN}(\texttt{X}) - \frac{2}{\texttt{X}} - \frac{\texttt{X}}{\texttt{X}^2 + 1}$$

Notice:
You can also enter the expression using the equation writer (`EQW` key):

$$\int_1^{\texttt{X}} \frac{2}{\texttt{X}^6 + 2 \cdot \texttt{X}^4 + \texttt{X}^2}\, \texttt{dX}$$

This gives the same result, barring an integration constant equal to

$$\frac{3 \cdot \pi + 10}{4}$$

Exercise 2
Calculate:

$$\int \frac{1}{\sin(x) + \sin(2 \cdot x)}\, dx$$

Type:

$$\texttt{INTVX}(\frac{1}{\texttt{SIN}(\texttt{X}) + \texttt{SIN}(2 \cdot \texttt{X})})$$

You find:

$$-\frac{1}{6} \cdot \texttt{LN}(|\texttt{COS}(\texttt{X}) - 1|) + \frac{1}{2} \cdot \texttt{LN}(|\texttt{COS}(\texttt{X}) + 1|) + \frac{-2}{3} \cdot \texttt{LN}(|2 \cdot \texttt{COS}(\texttt{X}) + 1|) - \texttt{LN}(2)$$

4.6.4 LIMIT

Find, for $n > 2$, the limit when x approaches 0 of:

$$\frac{n \times \tan(x) - \tan(n \times x)}{\sin(n \times x) - n \times \sin(x)}$$

You can use the `LIMIT` command, found in the menu:
`CALC (blue-shift 4)`, sub-menu 2 `LIMIT & SERIES` (or you can type it in

α mode).
Type:

$$\texttt{LIMIT}\left(\frac{\texttt{N}\cdot\texttt{TAN(X)}-\texttt{TAN(N}\cdot\texttt{X)}}{\texttt{SIN(N}\cdot\texttt{X)}-\texttt{N}\cdot\texttt{SIN(X)}},\texttt{0}\right)$$

You obtain:

$$2$$

Find the limit, when x approaches $+\infty$, of:

$$\sqrt{x+\sqrt{x+\sqrt{x}}}-\sqrt{x}$$

Type:

$$\texttt{LIMIT}(\sqrt{\texttt{X}+\sqrt{\texttt{X}+\sqrt{\texttt{X}}}}-\sqrt{\texttt{X}},+\infty)$$

After a moment, you obtain:

$$\frac{1}{2}$$

Notice that you obtain $+\infty$ pressing:

$$^{+}/_{-}\ \ ^{+}/_{-}\ \ \infty\ (\texttt{blue}-\texttt{shift 0})$$

4.6.5 LIMIT and $\int$

Find the limit, when a approaches $+\infty$, of:

$$\int_2^a (\frac{x}{x^2-1}+\ln(\frac{x+1}{x-1}))\ dx$$

In the `equation writer`, type:

$$\int_2^{+\infty} (\frac{\texttt{X}}{\texttt{X}^2-\texttt{1}}+\texttt{LN}(\frac{\texttt{X}+\texttt{1}}{\texttt{X}-\texttt{1}}))\ \texttt{dX}$$

Notice that tou obtain $+\infty$ pressing:

$$^{+}/_{-}\ \ ^{+}/_{-}\ \ \infty\ (\texttt{blue}-\texttt{shift 0})$$

You obtain:

$$+\infty-\frac{\texttt{7.LN(3)}}{\texttt{2}}$$

and, after simplification:

$$+\infty$$

4.6.6 IBP

IBP has two arguments: an expression you can write as $u(x) \cdot v'(x)$ and $v(x)$.
IBP returns a list containing $u(x) \cdot v(x)$ and $-v(x) \cdot u'(x)$, that is, the two terms you must calculate when doing an integration by parts.
You must then integrate the second term and add the result to the first term to obtain an antiderivative of $u(x) \cdot v'(x)$ (this is handy in RPN mode!).
Type:

$$\texttt{IBP}(\texttt{LN}(\texttt{X}),\texttt{X})$$

You obtain:

$$\{\texttt{X}.\texttt{LN}(\texttt{X}),-1\}$$

Notice: If the first argument of IBP is a list of two elements, IBP only operates on the last element of the list, and adds the integration result to the first element (so that it is easy to invoke IBP multiple times in algebraic mode).

4.6.7 RISCH

RISCH has two arguments: an expression and the name of a variable.
RISCH returns an antiderivative of the first argument with respect to the variable given as the second argument.
Type:

$$\texttt{RISCH}((2\cdot\texttt{X}^2+1)\cdot\texttt{EXP}(\texttt{X}^2+1),\texttt{X})$$

You obtain:

$$\texttt{X}\cdot\texttt{EXP}(\texttt{X}^2+1)$$

4.7 Trigonometric expressions

4.7.1 TEXPAND

TEXPAND has a trigonometric expression as argument.
TEXPAND expands this expression with respect to $\sin(x)$ and $\cos(x)$.
Example 1:
Type:

$$\texttt{TEXPAND}(\texttt{COS}(\texttt{X}+\texttt{Y}))$$

You obtain:

$$\texttt{COS}(\texttt{Y})\cdot\texttt{COS}(\texttt{X})-\texttt{SIN}(\texttt{Y})\cdot\texttt{SIN}(\texttt{X})$$

Example 2:
Type:

$$\texttt{TEXPAND}(\texttt{COS}(3\cdot\texttt{X}))$$

You obtain:

$$4\cdot\texttt{COS}(\texttt{X})^3-3\cdot\texttt{COS}(\texttt{X})$$

Example 3:
Type:

$$\texttt{TEXPAND}(\frac{\texttt{SIN}(3\cdot\texttt{X})+\texttt{SIN}(7\cdot\texttt{X})}{\texttt{SIN}(5\cdot\texttt{X})})$$

You obtain, after one simplification step ($\triangle$ ENTER) :

$$4\cdot\texttt{COS}(\texttt{X})^2-2$$

4.7.2 TLIN

TLIN has a trigonometric expression as argument.
TLIN linearizes this expression in function of $\sin(n\cdot x)$ and $\cos(n\cdot x)$.
Example:
Type:

$$\texttt{TLIN}(\texttt{COS}(\texttt{X})\cdot\texttt{COS}(\texttt{Y}))$$

You obtain:

$$\frac{1}{2}\cdot\texttt{COS}(\texttt{X}-\texttt{Y})+\frac{1}{2}\cdot\texttt{COS}(\texttt{X}+\texttt{Y})$$

Example 2:
Type:

$$\texttt{TLIN}(\texttt{COS}(\texttt{X})^3)$$

You obtain:

$$\frac{1}{4}\cdot\texttt{COS}(3\cdot\texttt{X})+\frac{3}{4}\cdot\texttt{COS}(\texttt{X})$$

Example 3:
Type:

$$\texttt{TLIN}(4 \cdot \texttt{COS}(\texttt{X})^2 - 2)$$

You obtain:

$$2 \cdot \texttt{COS}(2 \cdot \texttt{X})$$

4.7.3 TCOLLECT

TCOLLECT has a trigonometric expression as argument.
TCOLLECT linearizes this expression in function of $\sin(n \cdot x)$ and $\cos(n \cdot x)$, then collects in real mode sines and cosines of the same angle.
Type:

$$\texttt{TCOLLECT}(\texttt{SIN}(\texttt{X}) + \texttt{COS}(\texttt{X}))$$

You obtain:

$$\sqrt{2} \cdot \texttt{COS}(\texttt{X} - \frac{\pi}{4})$$

4.7.4 ACOS2S

ACOS2S has a trigonometric expression as argument.
ACOS2S rewrites this expression replacing $\arccos(x)$ with $\frac{\pi}{2} - \arcsin(x)$.
Type:

$$\texttt{ACOS2S}(\texttt{ACOS}(\texttt{X}) + \texttt{ASIN}(\texttt{X}))$$

You obtain:

$$\frac{\pi}{2}$$

4.7.5 ASIN2C

ASIN2C has a trigonometric expression as argument.
ASIN2C rewrites this expression replacing $\arcsin(x)$ with $\frac{\pi}{2} - \arccos(x)$.
Type:

$$\texttt{ASIN2C}(\texttt{ACOS}(\texttt{X}) + \texttt{ASIN}(\texttt{X}))$$

You obtain:

$$\frac{\pi}{2}$$

4.7.6 ASIN2T

ASIN2T has a trigonometric expression as argument.
ASIN2T rewrites this expression replacing $\arcsin(x)$ with $\arctan(\frac{x}{\sqrt{1-x^2}})$.
Type:

```
ASIN2T(ASIN(X))
```

You obtain:

$$\mathtt{ATAN}(\frac{\mathtt{X}}{\sqrt{1-\mathtt{X}^2}})$$

4.7.7 ATAN2S

ATAN2S has a trigonometric expression as argument.
ATAN2S rewrites this expression replacing $\arctan(x)$ with $\arcsin(\frac{x}{\sqrt{1+x^2}})$.
Type:

```
ATAN2S(ATAN(X))
```

You obtain:

$$\mathtt{ASIN}(\frac{\mathtt{X}}{\sqrt{\mathtt{X}^2+1}})$$

4.7.8 SINCOS

SINCOS accepts as argument an expression containing complex exponentials.
SINCOS rewrites this expression in function of $\sin(x)$ and $\cos(x)$.
Type:

```
SINCOS(EXP(i.X))
```

You obtain:

$$\mathtt{COS(X)+i.SIN(X)}$$

4.7.9 TAN2SC

TAN2SC has a trigonometric expression as argument.
TAN2SC rewrites this expression replacing $\tan(x)$ with $\frac{\sin(x)}{\cos(x)}$.
Type:

```
TAN2SC(TAN(X))
```

You obtain:

$$\frac{\mathtt{SIN(X)}}{\mathtt{COS(X)}}$$

4.7.10 TAN2SC2

TAN2SC2 has a trigonometric expression as argument.
TAN2SC2 rewrites this expression replacing $\tan(x)$ with $\frac{\sin(2\cdot x)}{1+\cos(2\cdot x)}$ (or with $\frac{1-\cos(2\cdot x)}{\sin(2\cdot x)}$ if you prefer sines, that is, when flag -116 is set to `Prefer sin()`; see 0.4 for more details).
Type:

```
TAN2SC2(TAN(X))
```

You obtain:

$$\frac{\mathtt{SIN(2\cdot X)}}{1+\mathtt{COS(2\cdot X)}}$$

4.7.11 HALFTAN

HALFTAN has a trigonometric expression as argument.
HALFTAN rewrites $\sin(x)$, $\cos(x)$ and $\tan(x)$ terms of the expression in function of $\tan(\frac{x}{2})$.
Type:

$$\mathtt{HALFTAN}(\frac{\mathtt{SIN(2\cdot X)}}{1+\mathtt{COS(2\cdot X)}})$$

You obtain, after simplification:

$$\mathtt{TAN(X)}$$

Type:

$$\texttt{HALFTAN(SIN(X)}^2 + \texttt{COS(X)}^2\texttt{)}$$

You obtain ($\texttt{SQ(X)} = \texttt{X}^2$):

$$\left(\frac{2 \cdot \texttt{TAN}(\frac{\texttt{X}}{2})}{\texttt{SQ}(\texttt{TAN}(\frac{\texttt{X}}{2})) + 1}\right)^2 + \left(\frac{1 - \texttt{SQ}(\texttt{TAN}(\frac{\texttt{X}}{2}))}{\texttt{SQ}(\texttt{TAN}(\frac{\texttt{X}}{2})) + 1}\right)^2$$

You obtain, after simplification:

$$1$$

4.7.12 TRIG

TRIG has a trigonometric expression as argument.
TRIG simplifies this expression using the identity: $\sin(x)^2 + \cos(x)^2 = 1$.
Type:

$$\texttt{TRIG(SIN(X)}^2 + \texttt{COS(X)}^2 + 1\texttt{)}$$

You obtain:

$$2$$

4.7.13 TRIGSIN

TRIGSIN has a trigonometric expression as argument.
TRIGSIN simplifies this expression using the identity: $\sin(x)^2 + \cos(x)^2 = 1$, privileging and preserving $\sin(x)$ terms.
Type:

$$\texttt{TRIGSIN(SIN(X)}^4 + \texttt{COS(X)}^2 + 1\texttt{)}$$

You obtain:

$$\texttt{SIN(X)}^4 - \texttt{SIN(X)}^2 + 2$$

4.7.14 TRIGCOS

TRIGCOS has a trigonometric expression as argument.
TRIGCOS simplifies this expression using the identity: $\sin(x)^2 + \cos(x)^2 = 1$, privileging and preserving $\cos(x)$ terms.
Type:

$$\texttt{TRIGCOS}(\texttt{SIN}(\texttt{X})^4 + \texttt{COS}(\texttt{X})^2 + 1)$$

You obtain:

$$\texttt{COS}(\texttt{X})^4 - \texttt{COS}(\texttt{X})^2 + 2$$

4.7.15 TRIGTAN

TRIGTAN has a trigonometric expression as argument.
TRIGTAN simplifies this expression using the identity: $\sin(x)^2 + \cos(x)^2 = 1$, privileging and preserving $\tan(x)$ terms.
Type:

$$\texttt{TRIGTAN}(\texttt{SIN}(\texttt{X})^4 + \texttt{COS}(\texttt{X})^2 + 1)$$

You obtain:

$$\frac{2 \cdot \texttt{TAN}(\texttt{X})^4 + 3 \cdot \texttt{TAN}(\texttt{X})^2 + 2}{\texttt{TAN}(\texttt{X})^4 + 2 \cdot \texttt{TAN}(\texttt{X})^2 + 1}$$

4.7.16 FOURIER

FOURIER has two arguments: an expression $f(x)$ and an integer n.
FOURIER returns the Fourier coefficient c_n of $f(x)$. $f(x)$ is assumed to be a periodic function defined on the interval $[0, T]$, with period T. (T is the current value of the PERIOD variable).
If f is piecewise continuous:

$$f(x) = \sum_{n=-\infty}^{+\infty} c_n e^{\frac{2inx\pi}{T}}$$

Example: Find the Fourier coefficients of the function f; the period of f is $2.\pi$, and f is defined on [0 $2.\pi$[as $f(x) = x^2$.
Type:

$$2 \cdot \pi \ \ \texttt{STO}\triangleright\ \texttt{PERIOD}$$

$$\texttt{FOURIER}(\texttt{X}^2, \texttt{N})$$

You obtain after simplification:

$$\frac{2 \cdot \texttt{i} \cdot \texttt{N} \cdot \pi + 2}{\texttt{N}^2}$$

So, if $n \neq 0$:

$$c_n = \frac{2 \cdot i \cdot N \cdot \pi + 2}{N^2}$$

Then, type:

$$\texttt{FOURIER}(\texttt{X}^2, 0)$$

You obtain:

$$\frac{4 \cdot \pi^2}{3}$$

So, if $n = 0$:

$$c_0 = \frac{4 \cdot \pi^2}{3}$$

4.8 Exponentials and Logarithms

4.8.1 EXPLN

EXPLN has a trigonometric expression as argument.
EXPLN rewrites the trigonometric expression in terms of exponentials and logarithms WITHOUT linearization.
EXPLN demands to put the calculator in complex mode.
Type:

$$\texttt{EXPLN}(\texttt{SIN}(\texttt{X}))$$

You obtain:

$$\frac{\texttt{EXP}(\texttt{i} \cdot \texttt{X}) - \frac{1}{\texttt{EXP}(\texttt{i} \cdot \texttt{X})}}{2 \cdot \texttt{i}}$$

4.8.2 LIN

LIN has an expression containing exponentials and trigonometric functions as argument.
LIN linearizes the expression (that is, it rewrites the expression in terms of $\exp(n \cdot x)$).

LIN demands to put the calculator in complex mode when the input expression contains trigonometric functions.

Example 1 :
Type:

$$\texttt{LIN}((\texttt{SIN}(\texttt{X}))$$

You obtain:

$$-(\frac{\texttt{i}}{2}\cdot\texttt{EXP}(\texttt{i}\cdot\texttt{X}))+\frac{\texttt{i}}{2}\cdot\texttt{EXP}(-(\texttt{i}\cdot\texttt{X}))$$

Example 2 :
Type:

$$\texttt{LIN}((\texttt{COS}(\texttt{X})^2)$$

You obtain:

$$-(\frac{1}{4}\cdot\texttt{EXP}(2\cdot\texttt{i}\cdot\texttt{X}))+\frac{1}{2}+\frac{1}{4}\cdot\texttt{EXP}(-(2\cdot\texttt{i}\cdot\texttt{X}))$$

Example 3 :
Type:

$$\texttt{LIN}((\texttt{EXP}(\texttt{X})+1)^3)$$

You obtain:

$$3\cdot\texttt{EXP}(\texttt{X})+1+3\cdot\texttt{EXP}(2\cdot\texttt{X})+\texttt{EXP}(3\cdot\texttt{X})$$

4.8.3 LNCOLLECT

LNCOLLECT has an expression containing logarithms as argument.
LNCOLLECT collects the logarithmic terms. Therefore, it is better to use it on a factorized expression (using FACTOR beforehand).
Type:

$$\texttt{LNCOLLECT}(\texttt{LN}(\texttt{X}+1)+\texttt{LN}(\texttt{X}-1))$$

You obtain:

$$\texttt{LN}((\texttt{X}+1)(\texttt{X}-1))$$

4.8.4 TSIMP

TSIMP has an expression as argument; it simplifies the expression rewriting it in function of complex exponentials (enabling complex mode in the process), and then reducing the number of variables as returned by LVAR (see section 4.20.2).
Use TSIMP only as a last resort.
Type:

$$\texttt{TSIMP}(\frac{\texttt{SIN}(3\cdot\texttt{X})+\texttt{SIN}(7\cdot\texttt{X})}{\texttt{SIN}(5\cdot\texttt{X})})$$

You obtain after simplification (that is, after copying the result 2 times):

$$\frac{\texttt{EXP}(\texttt{i}\cdot\texttt{X})^4+1}{\texttt{EXP}(\texttt{i}\cdot\texttt{X})^2}$$

4.9 Polynomials

4.9.1 GCD

GCD returns the gcd (greatest common divisor) of two polynomials (or of two lists of polynomials with the same length).
Type:

$$\texttt{GCD}(\texttt{X}^2+2\cdot\texttt{X}+1,\texttt{X}^2-1)$$

You obtain:

$$\texttt{X}+1$$

Type:

$$\texttt{GCD}(\{\texttt{X}^2+2\cdot\texttt{X}+1,\texttt{X}^3-1\},\{\texttt{X}^2-1,\texttt{X}^2+\texttt{X}-2\})$$

You obtain:

$$\{\texttt{X}+1,\texttt{X}-1\}$$

4.9.2 LGCD

LGCD denotes the gcd (greatest common divisor) of a list of polynomials.
LGCD returns a list containing the given list of polynomials and the GCD of all polynomials of the list.
Type:

$$\texttt{LGCD}(\{\texttt{X}^2+2\cdot\texttt{X}+1,\texttt{X}^3+1,\texttt{X}^2-1,\texttt{X}^2+\texttt{X}\})$$

You obtain:

$$\{\{X^2+2\cdot X+1, X^3+1, X^2-1, X^2+X\}\ ,\ X+1\}$$

4.9.3 SIMP2

SIMP2 has two polynomials (or two lists of polynomials with the same length) as arguments. These two polynomials are considered as representing a rational fraction. SIMP2 returns the simplified rational fraction, represented as a list of two polynomials.
Type:

$$\texttt{SIMP2}(X^3-1, X^2-1)$$

You obtain:

$$\{X^2+X+1, X+1\}$$

4.9.4 LCM

LCM returns the lcm (least common multiple) of two polynomials (or of two lists of polynomials with the same length).
Type:

$$\texttt{LCM}(X^2+2\cdot X+1, X^2-1)$$

You obtain:

$$(X^2+2\cdot X+1)\cdot(X-1)$$

4.9.5 FACTOR

FACTOR has either a polynomal or a list of polynomials as argument.
FACTOR factors its input.
Type:

$$\texttt{FACTOR}(X^2+2\cdot X+1)$$

You obtain:

$$(X+1)^2$$

Type:

$$\texttt{FACTOR}(X^4-2.X^2+1)$$

You obtain:

$$(X-1)^2.(X+1)^2$$

Type:

$$\texttt{FACTOR}(\{X^3-2.X^2+1, X^2-X\})$$

You obtain:

$$\{\frac{(\mathtt{X}-1)\cdot(2\cdot\mathtt{X}+-1+\sqrt{5})\cdot(2\cdot\mathtt{X}-(1+\sqrt{5}))}{4}\ ,\ \mathtt{X}\cdot(\mathtt{X}-1)\}$$

4.9.6 FACTORS

FACTORS has either a polynomial or a list of polynomials as argument.
FACTORS returns a list containing the factors of the polynomial and their exponents.
Type:

$$\mathtt{FACTORS}(\mathtt{X}^2+2\cdot\mathtt{X}+1)$$

you obtain:

$$\{\mathtt{X}+1, 2.\}$$

Type:

$$\mathtt{FACTORS}(\mathtt{X}^4-2\cdot\mathtt{X}^2+1)$$

You obtain:

$$\{\mathtt{X}-1, 2.\ ,\ \mathtt{X}+1, 2.\}$$

Type:

$$\mathtt{FACTORS}(\{\mathtt{X}^3-2\cdot\mathtt{X}^2+1, \mathtt{X}^2-\mathtt{X}\})$$

You obtain:

$$\{\{\mathtt{X}-1, 1.\ ,\ 2\cdot\mathtt{X}+-1+\sqrt{5}, 1.\ ,\ 2\cdot\mathtt{X}-(1+\sqrt{5}), 1.\ ,\ 4, -1.\},$$

$$\{\mathtt{X}, 1.\ ,\ \mathtt{X}-1, 1.\}\}$$

4.9.7 DIVIS

DIVIS has either a polynomial or a list of polynomials as argument, and returns the list of its divisors.
Type:

$$\mathtt{DIVIS}(\mathtt{X}^4-1)$$

You obtain:

$$\{1, \mathtt{X}^2+1, \mathtt{X}-1, \mathtt{X}^3-\mathtt{X}^2+\mathtt{X}-1, \mathtt{X}+1, \mathtt{X}^3+\mathtt{X}^2+\mathtt{X}+1, \mathtt{X}^2-1, \mathtt{X}^4-1\}$$

4.9.8 QUOT

QUOT returns the quotient of the division between two polynomials.
Type:

$$\texttt{QUOT}(\texttt{X}^2 + 2 \cdot \texttt{X} + 1, \texttt{X})$$

You obtain:

$$\texttt{X} + 2$$

4.9.9 REMAINDER

REMAINDER returns the remainder of the division between two polynomials.
Type:

$$\texttt{REMAINDER}(\texttt{X}^3 - 1, \texttt{X}^2 - 1)$$

you obtain:

$$\texttt{X} - 1$$

4.9.10 DIV2

Returns a list containing both the quotient and the remainder of the division between two polynomials.
Type:

$$\texttt{DIV2}(\texttt{X}^2 + 2 \cdot \texttt{X} + 1, \texttt{X})$$

You obtain:

$$\{\texttt{X} + 2, 1\}$$

The step-by-step mode can be of interest here, because it displays the intermediate steps of the division process.

4.9.11 EGCD

This command applies the Bézout identity (Extended Greatest Common Divisor). EGCD(A[X], B[X]) returns {D[X], U[X], V[X]}, where D, U, V satisfy the following relation:

$$\texttt{D[X]} = \texttt{U[X]} * \texttt{A[X]} + \texttt{V[X]} * \texttt{B[X]}$$

Type:

$$\texttt{EGCD}(\texttt{X}^2 + 2 \cdot \texttt{X} + 1, \texttt{X}^2 - 1)$$

You obtain:

$$\{2 \cdot \texttt{X} + 2, 1, -1\}$$

4.9.12 ABCUV

This command applies the Bézout identity like EGCD but, now, the arguments are three polynomials, A, B, C (C must be a multiple of GCD(A,B)): ABCUV(A[X], B[X], C[X]) returns {U[X], V[X]}, where U, V satisfy the following:

$$\mathtt{C[X]} = \mathtt{U[X]} * \mathtt{A[X]} + \mathtt{V[X]} * \mathtt{B[X]}$$

Type:

$$\mathtt{ABCUV}(\mathtt{X}^2 + 2 \cdot \mathtt{X} + 1, \mathtt{X}^2 - 1, \mathtt{X} + 1)$$

You obtain:

$$\{\frac{1}{2}, \frac{-1}{2}\}$$

Type:

$$\mathtt{ABCUV}(\mathtt{X}^2 + 2 \cdot \mathtt{X} + 1, \mathtt{X}^2 - 1, \mathtt{X}^3 + 1)$$

You obtain:

$$\{\frac{\mathtt{X}^2 - \mathtt{X} + 1}{2}, -\frac{\mathtt{X}^2 - \mathtt{X} + 1}{2}\}$$

4.9.13 HORNER

HORNER has two arguments: a polynomial $P[X]$ and a number a; it returns a list containing $Q[X]$ (quotient of $P[X]$ divided by $X - a$), a, and $P[a]$.
Type:

$$\mathtt{HORNER}(\mathtt{X}^4 + 2 \cdot \mathtt{X}^3 - 3 \cdot \mathtt{X}^2 + \mathtt{X} - 2, 1)$$

You obtain:

$$\{\mathtt{X}^3 + 3 \cdot \mathtt{X}^2 + 1 \ , \ 1 \ , \ -1\}$$

4.9.14 PTAYL

Rewrites a polynomial $P[X]$ in function of the powers of $X - a$.
PTAYL has two arguments: a polynomial P and a number a.
Type:

$$\mathtt{PTAYL}(\mathtt{X}^2 + 2 \cdot \mathtt{X} + 1, 2)$$

you obtain the polynomial $Q[X]$:

$$\mathtt{X}^2 + 6 \cdot \mathtt{X} + 9$$

Warning, notice that:

$$\mathtt{P(X)} = \mathtt{Q(X - 2)}$$

4.9.15 ZEROS

ZEROS has two arguments: a polynomial P and a variable name.
ZEROS returns a list containing the zeros of P with respect to the given variable, WITHOUT their multiplicity.
Type:

$$\texttt{ZEROS}(\texttt{X}^4 - 1, \texttt{X})$$

You obtain:
-in real mode

$$\{-1\ ,\ 1\}$$

-in complex mode

$$\{-1\ ,\ 1\ ,\ -\texttt{i}\ ,\ \texttt{i}\}$$

4.9.16 PROOT

PROOT is the numeric command of the HP48.
PROOT has a vector containing the coefficients of a monovariate polynomial (ordered by decreasing powers of the polynomial's variable) as argument.
PROOT returns a vector whose elements are the roots of the polynomial.
To find the roots of $P[x] = x^5 - 2 \cdot x^4 + x^3$, type:

$$\texttt{PROOT}([1, -2, 1, 0, 0, 0])$$

You obtain:

$$[0., 0., 0., 1., 1.]$$

The result means that 0 is a triple root, and 1 is a double root of $P[x]$.

4.9.17 FROOTS

FROOTS has a rational function $F[x]$ as argument.
FROOTS returns a vector whose components are the roots and the poles of $F[x]$, followed by their multiplicity.
Type:

$$\texttt{FROOTS}(\frac{\texttt{X}^5 - 2 \cdot \texttt{X}^4 + \texttt{X}^3}{\texttt{X} - 2})$$

You obtain:

$$[2, -1., 0, 3., 1, 2.]$$

The result means that: 2 is a pole of order 1, 0 is a triple root, and 1 is a double root of $F[x] = \frac{x^5 - 2 \cdot x^4 + x^3}{x-2}$.

4.9.18 PCOEF

PCOEF is the numeric command of the HP48.
PCOEF has a vector containing the roots of a polynomial $P[x]$ as argument.
PCOEF returns a vector whose components are the coefficients of the polynomial $P[x]$ (ordered by decreasing powers of the polynomial's variable).
Type:

$$\texttt{PCOEF}([1,2,0,0,3])$$

You obtain:

$$[1.,-6.,11.,-6.,0.,0.]$$

This means that $P[x]=(x-1)\cdot(x-2)\cdot x\cdot x\cdot(x-3)$ is equal to:
$x^5-6\cdot x^4+11\cdot x^3-6\cdot x^2$.

4.9.19 FCOEF

FCOEF has as argument a vector whose components are the roots and poles of a rational function $F[x]$, followed by their multiplicity.
FCOEF returns the rational function $F[x]$.
Type:

$$\texttt{FCOEF}([1,2,0,3,2,-1])$$

You obtain:

$$\frac{\texttt{X}^5-2\cdot\texttt{X}^4+\texttt{X}^3}{\texttt{X}-2}$$

since $(x-1)^2\cdot x^3=x^5-2\cdot x^4+x^3$

4.9.20 CHINREM

CHINREM has two vectors as arguments; each vector has two polynomials as components.
CHINREM returns a vector with two polynomials as components.
CHINREM([A[X],R[X]],[B[X],Q[X]]) finds the polyonimals P[X] and S[X] satisfying the following relations:
S[X] = R[X] · Q[X],
P[X] = A[X](modR[X]) and P[X] = B[X](modQ[X]).
There always is a solution P[X] if R[X] and Q[X] are mutually primes, and all solutions are congruent modulo S[X] = R[X] · Q[X].
Find the solutions $P[X]$ of:

$$\begin{cases} P[X] = & X \pmod{X^2+1} \\ P[X] = & X-1 \pmod{X^2-1} \end{cases}$$

Type:

$$\texttt{CHINREM}([\texttt{X}, \texttt{X}^2+1], [\texttt{X}-1, \texttt{X}^2-1])$$

You obtain:

$$[-\frac{\texttt{X}^2-2\cdot\texttt{X}+1}{2}, -\frac{\texttt{X}^4-1}{2}]$$

that is, $P[X] = -\frac{X^2-2\cdot X+1}{2} \pmod{-\frac{X^4-1}{2}}$

4.9.21 TRUNC

TRUNC truncates a polynomial to a given order.
TRUNC has two arguments: a polynomial and X^n.
TRUNC returns the polynomial truncated to order $n-1$ (no terms of order $\geq X^n$).
Type:

$$\texttt{TRUNC}((1+\texttt{X}+\frac{1}{2}\cdot\texttt{X}^2)^3\ ,\ \texttt{X}^4)$$

You obtain:

$$4\cdot\texttt{X}^3+\frac{9}{2}\cdot\texttt{X}^2+3\cdot\texttt{X}+1$$

4.9.22 LAGRANGE

LAGRANGE has as argument a matrix with two rows and n columns:
the first row corresponds to the abscissa values x_i, and the second row corresponds to ordinate values y_i $(i = 1..n)$.
LAGRANGE returns the polynomial P of degree $n-1$, so that $P(x_i) = y_i$.
Type:

$$\texttt{LAGRANGE}([[1,3],[0,1]])$$

You obtain:

$$\frac{\texttt{X}-1}{2}$$

in fact $\frac{x-1}{2} = 0$ for $x = 1$ and $\frac{x-1}{2} = 1$ for $x = 3$

4.9.23 LEGENDRE

LEGENDRE has as argument an integer value n.
LEGENDRE returns the non trivial polynomial solution of the differential equation:

$$(x^2-1)\cdot y'' - 2\cdot x\cdot y' - n(n+1)\cdot y = 0$$

Type:

LEGENDRE(4)

You obtain:

$$\frac{35 \cdot X^4 - 30 \cdot X^2 + 3}{8}$$

4.9.24 HERMITE

HERMITE has as argument an integer value n.
HERMITE returns the Hermite polynomial of degree n.
Type:

HERMITE(6)

You obtain:

$$64 \cdot X^6 - 480 \cdot X^4 + 720 \cdot X^2 - 120$$

4.9.25 TCHEBYCHEFF

TCHEBYCHEFF has as argument an integer value n.
If $n > 0$, TCHEBYCHEFF returns the polynomial T_n:

$$T_n[x] = \cos(n \cdot \arccos(x))$$

If $n < 0$ TCHEBYCHEFF returns the the polynomial T_n:

$$T_n[x] = \frac{\sin(n \cdot \arccos(x))}{\sin(\arccos(x))}$$

Type:

TCHEBYCHEFF(4)

You obtain:

$$8 \cdot X^4 - 8 \cdot X^2 + 1$$

in fact:
$\cos(4 \cdot x) = Re((\cos(x) + i \cdot \sin(x))^4)$
$\cos(4 \cdot x) = \cos(x)^4 - 6 \cdot \cos(x)^2 \cdot (1 - \cos(x)^2) + ((1 - \cos(x)^2)^2$.
$\cos(4 \cdot x) = T_4(\cos(x))$.
Type:

TCHEBYCHEFF(−4)

You obtain:

$$8 \cdot X^3 - 4 \cdot X$$

in fact:
$\sin(4 \cdot x) = \sin(x) \cdot (8 \cdot \cos(x)^3 - 4 \cdot \cos(x))$.

4.9.26 REORDER

REORDER has two arguments: an expression and a vector containing an ordered list of variables.
REORDER reorders the input expression following the order of variables given by its second argument.
Type:

$$\texttt{REORDER}(\texttt{X}^2 + 2 \cdot \texttt{X} \cdot \texttt{A} + \texttt{A}^2 + \texttt{Z}^2 - \texttt{X} \cdot \texttt{Z}, [\texttt{A}, \texttt{X}, \texttt{Z}])$$

You obtain:

$$\texttt{A}^2 + 2 \cdot \texttt{X} \cdot \texttt{A} + \texttt{X}^2 - \texttt{Z} \cdot \texttt{X} + \texttt{Z}^2$$

4.10 Rational fractions

4.10.1 FXND

FXND has a rational fraction as argument, and returns a list containing the simplified numerator and denominator of this fraction.
Type:

$$\texttt{FXND}(\frac{\texttt{X}^2 - 1}{\texttt{X} - 1})$$

You obtain:

$$\{\texttt{X} + 1, 1\}$$

4.10.2 SIMP2

SIMP2 has two polynomials (or two lists of polynomials with the same length) as arguments. These two polynomials are considered as representing a rational fraction.
SIMP2 simplifies the rational fraction and returns the result as a list of two polynomials.
Type:

$$\texttt{SIMP2}(\texttt{X}^3 - 1, \texttt{X}^2 - 1)$$

You obtain:

$$\{\texttt{X}^2 + \texttt{X} + 1, \texttt{X} + 1\}$$

4.10.3 PROPFRAC

PROPFRAC has a rational fraction as argument.
PROPFRAC rewrites the rational fraction to put its integer part in evidence and returns the result. In other words, PROPFRAC(A(X)/B(X)) rewrites the rational fraction $\frac{A[X]}{B[X]}$ as:

$$Q[X] + \frac{R[X]}{B[X]}$$

where $R[X] = 0$ or $0 \leq deg(R[X]) < deg(B[X])$.
Type:

$$\texttt{PROPFRAC}(\frac{(5 \cdot \texttt{X} + 3) \cdot (\texttt{X} - 1)}{\texttt{X} + 2})$$

You obtain:

$$5 \cdot \texttt{X} - 12 + \frac{21}{\texttt{X} + 2}$$

4.10.4 PARTFRAC

To decompose into partial fractions a rational fraction like

$$\frac{x^5 - 2 \times x^3 + 1}{x^4 - 2 \times x^3 + 2 \times x^2 - 2 \times x + 1}$$

you can use the PARTFRAC command.
You can find this command in the ARITH (blue-shift 1) menu, sub-menu 2.POLYNOMIAL..., position 14 (or you can type it in α mode).
Type:

$$\texttt{PARTFRAC}(\frac{\texttt{X}^5 - 2 * \texttt{X}^3 + 1}{\texttt{X}^4 - 2 * \texttt{X}^3 + 2 * \texttt{X}^2 - 2 * \texttt{X} + 1})$$

In real mode, you obtain:

$$\texttt{X} + 2 + \frac{\frac{-1}{2}}{\texttt{X} - 1} + \frac{\frac{\texttt{X}-3}{2}}{\texttt{X}^2 + 1}$$

In complex mode, you obtain instead:

$$\texttt{X} + 2 + \frac{\frac{1-3.\texttt{i}}{4}}{\texttt{X} + \texttt{i}} + \frac{\frac{-1}{2}}{\texttt{X} - 1} + \frac{\frac{1+3.\texttt{i}}{4}}{\texttt{X} - \texttt{i}}$$

4.11 Modular calculations

You can do calculations "modulo p", that is, in either Z/pZ or $Z/pZ[X]$.
Warning: for some commands, p must be a prime number.
The calculator uses the symmetrical representation of elements (-1 instead of 6 modulo 7).
The value of p must be stored into the MODULO variable in the HOME directory.
ALL THE FOLLOWING EXAMPLES ASSUME THAT p=13.

4.11.1 MODSTO

To store into MODULO the value of p (for example p=13) you can use either: MODE cas MODULO ..., or 13 STO▷ MODULO (if the current directory is HOME), or MODSTO(13).
MODSTO allows you to change the value of the MODULO variable in the HOME directory.
For example, you will type: MODSTO(5) or 5 STO▷ MODULO to do your calculations modulo 5.
ALL THE FOLLOWING EXAMPLES ASSUME THAT p=13.

4.11.2 ADDTMOD

ADDTMOD performs an addition in $Z/pZ[X]$.
Type:

ADDTMOD(11X + 5, 8X + 6)

You obtain:

6X − 2

4.11.3 SUBTMOD

SUBTMOD peforms a subtraction in $Z/pZ[X]$.
Type:

SUBTMOD(11X + 5, 8X + 6)

You obtain:

3X − 1

4.11.4 MULTMOD

MULTMOD performs a multiplication in $Z/pZ[X]$.
Type:

$$\texttt{MULTMOD}(\texttt{11X}+\texttt{5},\texttt{8X}+\texttt{6})$$

You obtain:

$$-(\texttt{3X}^2-\texttt{2X}-\texttt{4})$$

4.11.5 DIV2MOD

The arguments of DIV2MOD are two polynomials $A[X]$ and $B[X]$. The result is a list containing both the quotient and the remainder of the euclidean division of $A[X]$ by $B[X]$ in $Z/pZ[X]$.
Type:

$$\texttt{DI2VMOD}(\texttt{X}^3+\texttt{X}^2+\texttt{1},\texttt{2X}^2+\texttt{4})$$

Since:

$$X^3+X^2+1=(2X^2+4)\cdot(\frac{X+1}{2})+\frac{5X-4}{4}$$

you obtain:

$$\{\frac{\texttt{X}+\texttt{1}}{\texttt{2}}\ ,\ \frac{\texttt{5X}-\texttt{4}}{\texttt{4}}\}$$

Then, with

$$\texttt{EXPANDMOD}(\{\frac{\texttt{X}+\texttt{1}}{\texttt{2}}\ ,\ \frac{\texttt{5X}-\texttt{4}}{\texttt{4}}\})$$

you obtain:

$$\{-(\texttt{6X}+\texttt{6})\ ,\ -(\texttt{2X}+\texttt{1})\}$$

4.11.6 DIVMOD

The arguments of DIVMOD are two polynomials $A[X]$ and $B[X]$. The result is the rational fraction $\frac{A[X]}{B[X]}$ simplified in $Z/pZ[X]$.
Type:

$$\texttt{DIVMOD}(\texttt{2X}^2+\texttt{5},\texttt{5X}^2+\texttt{2X}-\texttt{3})$$

You obtain:

$$\frac{\texttt{5X}+\texttt{3}}{\texttt{6X}+\texttt{6}}$$

4.11.7 POWMOD

POWMOD(X,N) returns X raised to N in $Z/pZ[X]$.
The current value p of MODULO must be a prime number less than 100.
Type:

$$\texttt{POWMOD}(2\texttt{X}+1,5)$$

You obtain:

$$6\cdot\texttt{X}^5+2\cdot\texttt{X}^4+2\cdot\texttt{X}^3+\texttt{X}^2-3\cdot\texttt{X}+1$$

because:
$10 = -3 \pmod{13}$ $40 = 1 \pmod{13}$ $80 = 2 \pmod{13 =}$ $32 = 6 \pmod{13}$.

4.11.8 INVMOD

INVMOD has an integer as argument.
INVMOD returns the reciprocal of this integer in Z/pZ.
Type:

$$\texttt{INVMOD}(5)$$

You obtain (since $5 \times -5 = -25 = 1 \pmod{13}$) :

$$-5$$

4.11.9 GCDMOD

GCDMOD has two polynomials as arguments.
GCDMOD returns the GCD of the polynomials in $Z/pZ[X]$.
Type:

$$\texttt{GCDMOD}(2\texttt{X}^2+5,5\texttt{X}^2+2\texttt{X}-3)$$

You obtain:

$$-(4\texttt{X}-5)$$

4.11.10 EXPANDMOD

EXPANDMOD has a polynomial expression as argument.
EXPANDMOD expands this expression in $Z/pZ[X]$.
Type:

$$\texttt{EXPANDMOD}((2\texttt{X}^2+12).(5\texttt{X}-4))$$

You obtain:

$$-(3\texttt{X}^3-5\texttt{X}^2+5\texttt{X}-4)$$

4.11.11 FACTORMOD

FACTORMOD has a polynomial as argument.
FACTORMOD factorizes the polynomial in $Z/pZ[X]$ if $p \leq 97$ and p is prime.
Type:

$$\texttt{FACTORMOD}(-(3\texttt{X}^3 - 5\texttt{X}^2 + 5\texttt{X} - 4))$$

You obtain:

$$-((3\texttt{X} - 5)(\texttt{X}^2 + 6))$$

4.11.12 RREFMOD

RREFMOD solves a system of linear equations $AX = B$ in Z/pZ.
The argument is the matrix A augmented with the vector B as its rightmost column. The result is a matrix composed of $A1$ and $B1$, where $A1$ has zeros both above and under its principal diagonal, and the system $A1X = B1$ is equivalent to $AX = B$.
Type:

$$\texttt{RREFMOD}([[1, 2, 9][3, 10, 0]])$$

to solve

$$\begin{cases} x + 2 \cdot y = 9 \\ 3 \cdot x + 10 \cdot y = 0 \end{cases}$$

You obtain:

$$\begin{bmatrix} 2 & 0 & 6 \\ 0 & 4 & -1 \end{bmatrix}$$

that is, $2.\texttt{X} = 6$ and $4.\texttt{Y} = -1$ or, which is the same, $\texttt{X} = 3 \;\; \texttt{Y} = 3$ (since $-4 * 3 = 1 \pmod{13}$).

4.12 Limited and asymptotic expansions

4.12.1 DIVPC

DIVPC has three arguments: two polynomials A(X), B(X) (with $\texttt{B}(0) \neq 0$) and an integer n.
DIVPC returns the quotient Q(X) of the division of A(X) by B(X), with $\deg(\texttt{Q}) \leq \texttt{n}$ or $\texttt{Q} = 0$.
That is, $Q[X]$ is the series expansion, limited to order n, of $\frac{A[X]}{B[X]}$ about $X = 0$.
Type:

$$\texttt{DIVPC}(1 + \texttt{X}^2 + \texttt{X}^3, 1 + \texttt{X}^2, 5)$$

You obtain:

$$1 + X^3 - X^5$$

Warning: the calculator demands to enable the "increasing powers" mode; answer `yes` to proceed.

4.12.2 TAYLOR0

`TAYLOR0` has only one argument: a function of x to be expanded, and returns its Taylor expansion, limited to relative order 4, about $x = 0$ (where x is the current CAS variable).
Type:

$$\texttt{TAYLOR0}(\frac{\texttt{TAN}(\texttt{P}\cdot\texttt{X})-\texttt{SIN}(\texttt{P}\cdot\texttt{X})}{\texttt{TAN}(\texttt{Q}\cdot\texttt{X})-\texttt{SIN}(\texttt{Q}\cdot\texttt{X})})$$

You obtain:

$$\frac{\texttt{P}^5-\texttt{Q}^2\cdot\texttt{P}^3}{4\cdot\texttt{Q}^3}\cdot\texttt{X}^2+\frac{\texttt{P}^3}{\texttt{Q}^3}$$

Warning: "relative order 4" means to expand up to relative order 4 both numerator and denominator (here, absolute order 5 for both numerator and denominator); this gives an expansion of order 2 (5-3), because X^3 can be factored out in both the expanded numerator and denominator.

4.12.3 TAYLR

Determine a limited Taylor expansion of order 2 near $x = 0$ of:

$$\frac{3\times\tan(x)-\tan(3\times x)}{x\times(1-\cos(3\times x))}$$

Since the denominator has order 3, to obtain a limited expansion of order 2 about $x = 0$, it is necessary to expand the numerator near $x = 0$ up to order 5. To do this, use the `TAYLR` command; you can find it in the `CALC` (`blue-shift 4`) menu, sub-menu `2. LIMITS & SER...` at position 5 (or you can type it in α mode).
`TAYLR` is compatible with the HP48.
Type:

$$\texttt{TAYLR}(\frac{3\cdot\texttt{TAN}(\texttt{X})-\texttt{TAN}(3\cdot\texttt{X})}{\texttt{X}\cdot(1-\texttt{COS}(3\cdot\texttt{X}))},\texttt{X},5)$$

You obtain:

$$-\frac{16}{9}\cdot(1+\frac{19}{4}\cdot\texttt{X}^2)$$

4.12.4 SERIES

- expansion about x=a
  Example:
  Determine a series expansion limited to order 4 about $x = \frac{\pi}{6}$ of $\cos(2 \times x)^2$.

  You use the SERIES command; you can find it in the CALC (blue-shift 4) menu, position 8 (or you can type it in α mode).

  Type:

  $$\texttt{SERIES}(\texttt{COS}(2 \cdot \texttt{X})^2, \texttt{X} = \frac{\pi}{6}, 4)$$

  You obtain:

  $$\{\{\texttt{Limit} : \frac{1}{4} \quad \texttt{Equiv} : \frac{1}{4}$$

  $$\texttt{Expans} : \ (-\frac{8}{3}\texttt{h}^4 + \frac{8\sqrt{3}}{3}\texttt{h}^3 + 2\texttt{h}^2 - \sqrt{=}3\texttt{h} + \frac{1}{4})$$

  $$\texttt{Remain} : \ \frac{\texttt{h}^5}{4}\} \ \texttt{h} = \texttt{X} - \frac{\pi}{6}\}$$

- expansion about x=+∞ or x=-∞
  Example 1:
  Determine an expansion of $\arctan(x)$, of order 5, about x=+∞, assuming $h = \frac{1}{x}$.
  Type:

  $$\texttt{SERIES}(\texttt{ATAN}(\texttt{X}), \texttt{X} = +\infty, 5)$$

  You obtain:

  $$\{\{\texttt{Limit} : \frac{\pi}{2} \ \texttt{Equiv} : \frac{\pi}{2}$$

  $$\texttt{Expans} : \ (\frac{\pi}{2} - \texttt{h} + \frac{\texttt{h}^3}{3} - \frac{\texttt{h}^5}{5}) \ \texttt{Remain} : \frac{\pi\texttt{h}^6}{2}\} \ \texttt{h} = \frac{1}{\texttt{X}}\}$$

  Example 2 :
  Determine an expansion of $(2x - 1)e^{\frac{1}{x-1}}$, of order 2, about x=+$\infty$, assuming $h = \frac{1}{x}$.
  Type:

  $$\texttt{SERIES}((2\texttt{X} - 1) \cdot \texttt{EXP}(\frac{1}{\texttt{X} - 1}), \texttt{X} = +\infty, 3)$$

  You obtain:

  $$\{\{\texttt{Limit} : +\infty \ \texttt{Equiv} : \frac{2}{\texttt{h}}$$

$$\texttt{Expans}: \ (\frac{2+\texttt{h}+2\texttt{h}^2+\frac{17\texttt{h}^3}{6}}{\texttt{h}}) \ \texttt{Remain}: 2\texttt{h}^3\} \ \texttt{h} = \frac{1}{\texttt{X}}\}$$

Example 3 :
Determine en expansion of $(2x-1)e^{\frac{1}{x-1}})$, of order 2, about x=-∞, assuming $h = -\frac{1}{x}$.
Type:

$$\texttt{SERIES}((2\texttt{X}-1)\cdot\texttt{EXP}(\frac{1}{\texttt{X}-1}), \texttt{X} = -\infty, 3)$$

You obtain:

$$\{\{\texttt{Limit}: -\infty \ \texttt{Equiv}: -\frac{2}{\texttt{h}}$$

$$\texttt{Expans}: \ (\frac{-2+\texttt{h}-2\texttt{h}^2+\frac{17\texttt{h}^3}{6}}{\texttt{h}}) \ \texttt{Remain}: -2\texttt{h}^3\} \ \texttt{h} = -\frac{1}{\texttt{X}}\}$$

- unidirectional expansions
  You must give a positive real number (for example 4.) as the order to do an unidirectional expansion about the point $x = a$ with $x > a$, and a negative real number (for example -4.) to do an unidirectional expansion about $x = a$ with $x < a$.
  Example 1 :
  Determine an expansion of $\frac{(1+X)^{\frac{1}{X}}}{X^3}$, of order 2, about $X = 0^+$.
  Type:

$$\texttt{SERIES}(\frac{(1+\texttt{X})^{\frac{1}{\texttt{X}}}}{\texttt{X}^3}, \texttt{X}, 2.)$$

You obtain:

$$\{\{\texttt{Limit}: +\infty \ \texttt{Equiv}: \frac{\texttt{e}}{\texttt{h}^3} \ \texttt{Expans}: \ (-\frac{\texttt{e}\cdot\texttt{h}-2\cdot\texttt{e}}{2\cdot\texttt{h}^3}) \ \texttt{Remain}: \frac{\texttt{e}}{\texttt{h}}\} \ \texttt{h} = \texttt{X}\}$$

Example 2:
Determine an expansion of $\frac{(1+X)^{\frac{1}{X}}}{X^3}$, of order 2, about $X = 0^-$.
Type:

$$\texttt{SERIES}(\frac{(1+\texttt{X})^{\frac{1}{\texttt{X}}}}{\texttt{X}^3}, \texttt{X}, -2.)$$

You obtain:

$$\{\{\texttt{Limit}: -\infty \ \texttt{Equiv}: \frac{\texttt{e}}{\texttt{h}^3} \ \texttt{Expans}: \ (-\frac{\texttt{e}\cdot\texttt{h}-2\cdot\texttt{e}}{2\cdot\texttt{h}^3}) \ \texttt{Remain}: \frac{\texttt{e}}{\texttt{h}}\} \ \texttt{h} = \texttt{X}\}$$

Example 3 :

Determine an expansion of $\frac{(1+X)^{\frac{1}{X}}}{X^3}$, of order 2, about $X = 0$.
Type:

$$\texttt{SERIES}(\frac{(1+\texttt{X})^{\frac{1}{\texttt{X}}}}{\texttt{X}^3}, \texttt{X}, 2)$$

You obtain:

$$\{\{\texttt{Limit} : \infty \ \texttt{Equiv} : \frac{\texttt{e}}{\texttt{h}^3} \ \texttt{Expans} : \ (-\frac{\texttt{e}\cdot\texttt{h}-2\cdot\texttt{e}}{2\cdot\texttt{h}^3}) \ \texttt{Remain} : \frac{\texttt{e}}{\texttt{h}}\} \ \texttt{h} = \texttt{X}\}$$

4.12.5 LIMIT

LIMIT has as arguments an expression depending on a variable, and an equality (the variable equated to the value at which you want to calculate the limit of the expression).

Often, it is better to surrond the expression with quotes, to avoid a preliminary transformation of the expression into normal form (thus, avoiding a rational simplification) before the limit computation actually takes place.
For example, type:

$$\texttt{LIMIT}('(2\texttt{X}-1)\cdot\texttt{EXP}(\frac{1}{\texttt{X}-1})', \texttt{X} = +\infty)$$

You obtain:

$$+\infty$$

4.13 Matrices

4.13.1 TRAN

TRAN has a matrix A as argument.
TRAN returns the input matrix A transposed.
Type:

$$\texttt{TRAN}(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix})$$

You obtain:

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

4.13.2 TRN

TRN has a matrix A as argument.
TRN returns the adjoint of A (transpose of the conjugate) of A (this is the HP48 command).
Type:

$$\texttt{TRN}(\begin{bmatrix} i & 1+i \\ 1 & 1-i \end{bmatrix})$$

After simplification, you obtain:

$$\begin{bmatrix} -i & 1 \\ 1-i & 1+i \end{bmatrix}$$

4.13.3 MAD

MAD has a square matrix A, of order n, as argument.
MAD returns a list containing the determinant of A, the inverse of A, a list containing the matrix coefficients of a polynomial Q, and of the characteristic poloynomial P of A.
We have:

$$P(x) = (-1)^n \cdot \det(A - x \cdot I)$$

The polynomial with matrix coefficients $P(A) - P(x) \cdot I$ is divisible by $A - x \cdot I$ (since its value is zero for $x = A$). Let $Q(x)$ be their quotient.
Since $P(A) = 0$, we have $P(A) - P(x) \cdot I = -P(x) \cdot I = (A - x \cdot I) \cdot Q(x)$.
Therefore, $Q(x)$ is also the co-matrix of $A - x \cdot I$ and the following holds:
$Q(x) = I \cdot x^{n-1} + ... + B_0$, where B_0 is the co-matrix of A (with the sign exchanged if n is even!).
Type:

$$\texttt{MAD}(\begin{bmatrix} 4 & 1 & -2 \\ 1 & 2 & -1 \\ 2 & 1 & 0 \end{bmatrix})$$

You obtain:

$$\{8, \begin{bmatrix} \frac{1}{8} & -\frac{1}{4} & \frac{3}{8} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ -\frac{3}{8} & -\frac{1}{4} & \frac{7}{8} \end{bmatrix},$$

$$\{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -2 & 1 & -2 \\ 1 & -4 & -1 \\ 2 & 1 & -6 \end{bmatrix}, \begin{bmatrix} 1 & -2 & 3 \\ -2 & 4 & 2 \\ -3 & -2 & 7 \end{bmatrix}\},$$

$$\texttt{X}^3 - 6 \cdot \texttt{X}^2 + 12 \cdot \texttt{X} - 8\}$$

4.13.4 HADAMARD

HADAMARD has two matrices A and B, with the same size, as arguments.
HADAMARD returns the element-by-element product between A and B.
Type:

$$\texttt{HADAMARD}(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix})$$

You obtain:

$$\begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}$$

4.13.5 AXM

If given a symbolic matrix as argument, AXM returns an equivalent (but approximate) numeric matrix, and vice versa.
Type:

AXM([[1/2, 2], [3, 4]])

You obtain:

[[0.5, 2], [3, 4]]

4.13.6 AXL

When AXL is given a matrix as argument, it returns the same matrix rewritten as a list of lists. Vice versa, AXL transforms a list of lists into a matrix.
Type:

AXL([[1, 2], [3, 4]])

You obtain:

{{1, 2}{3, 4}}

Type:

AXL({{1, 2}{3, 4}})

You obtain:

[[1, 2], [3, 4]]

4.13.7 EGVL

EGVL has a matrix A, of order n, as argument.
EGVL returns a vector containing the n eigenvalues of A.
NOTICE: If A is a symbolic matrix, you will obtain only the eigenvalues that the CAS is able to determine (because it is necessary to symbolically factorize

the characteristic polynomial!)
Type:

$$\texttt{EGV}(\begin{bmatrix} 4 & 1 & -2 \\ 1 & 2 & -1 \\ 2 & 1 & 0 \end{bmatrix})$$

You obtain:

$$[2, 2, 2]$$

4.13.8 EGV

EGV has a matrix A, of order n, as argument.
EGV returns a list containing the matrix of the n column eigenvectors of A and the vector of the n eigenvalues of A (the same notice given for EGVL is valid here, too).
Type:

$$\texttt{EGV}(\begin{bmatrix} 4 & 1 & -2 \\ 1 & 2 & -1 \\ 2 & 1 & 0 \end{bmatrix})$$

You obtain:

$$\{\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 1 & 2 & 0 \end{bmatrix}, [2, 2, 2]\}$$

4.13.9 PCAR

PCAR has a matrix A, of order n, as argument.
PCAR returns the characteristic polynomial P of A ($P[x] = (-1)^n \cdot \det(A - x \cdot I)$)
Type:

$$\texttt{PCAR}(\begin{bmatrix} 4 & 1 & -2 \\ 1 & 2 & -1 \\ 2 & 1 & 0 \end{bmatrix})$$

You obtain:

$$\texttt{X}^3 - 6 \cdot \texttt{X}^2 + 12 \cdot \texttt{X} - 8$$

4.13.10 JORDAN

JORDAN has a matrix A, of order n, as argument.
JORDAN returns a list composed by the minimal polynomial M of A, the characteristic polynomial P of A, the list of the eigenvectors and characteristic vectors (each vector is preceded by its characteristic value), and the vector of

the n eigenvalues of A.
Type:

$$\texttt{JORDAN}(\begin{bmatrix} 4 & 1 & -2 \\ 1 & 2 & -1 \\ 2 & 1 & 0 \end{bmatrix})$$

You obtain:

$$\{\texttt{X}^3 - 6\texttt{X}^2 + 12\texttt{X} - 8, \texttt{X}^3 - 6\texttt{X}^2 + 12\texttt{X} - 8,$$
$$\{\texttt{Char} : 2 : [1,0,0], \texttt{Char} : 2 : [2,1,2], \texttt{Eigen} : 2 : [1,0,1]\}, [2,2,2]\}$$

4.13.11 HILBERT

HILBERT has an integer n as argument.
HILBERT returns the square Hilbert matrix of order n whose elements are given by:

$$a_{i,j} = \frac{1}{i+j-1}$$

Type:

$$\texttt{HILBERT}(4)$$

You obtain:

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

4.13.12 VANDERMONDE

VANDERMONDE has as argument a vector whose components are denoted by x_i.
VANDERMONDE returns the corresponding Vandermonde matrix (the k-th row of the matrix is the vector whose components are x_i^{k-1}).
Type:

$$\texttt{VANDERMONDE}([\texttt{A}, \texttt{B}, \texttt{C}])$$

You obtain:

$$\begin{bmatrix} 1 & 1 & 1 \\ \texttt{A} & \texttt{B} & \texttt{C} \\ \texttt{A}^2 & \texttt{B}^2 & \texttt{C}^2 \end{bmatrix}$$

4.13.13 LCXM

LCXM has as arguments two integers, n and p, and a program accepting as arguments i (a row number) and j (a column number) and yielding the value of $a_{i,j}$.
LCXM returns a $n \cdot p$ matrix having coefficients $a_{i,j}$. Type:

$$\texttt{LCXM}(2, 3, \ll \rightarrow \texttt{I J} \ \ll \texttt{I} + \texttt{J} \gg \gg)$$

You obtain:

$$\begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

4.14 Vectors

In the blue-shift SYMB (MTH) menu, you can find the functions to compute:
-the absolute value of a vector: ABS
-the dot product of two vectors: DOT
-the cross product of two vectors: CROSS

4.15 Quadratic forms

4.15.1 QXA

QXA has two arguments: a quadratic form q and a vector whose components are the form's variables.
QXA returns a list of two elements: the matrix A associated with q and the vector denoting the variables of the quadratic form.
Type:

$$\texttt{QXA}(2 \cdot \texttt{X} \cdot \texttt{Y} \ , \ [\texttt{X}, \texttt{Y}])$$

You obtain:

$$\{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \ , \ [\texttt{X}, \texttt{Y}]\}$$

4.15.2 AXQ

AXQ has two arguments: a symmetric matrix A representing a quadratic form q and a vector whose components are the quadratic form's variables.

AXQ returns a list of two elements: the quadratic form q and the vector denoting the form's variables.
Type:

$$\texttt{AXQ}([[0,1],[1,0]]\ ,\ [\texttt{X},\texttt{Y}])$$

You obtain:

$$\{2\cdot\texttt{X}\cdot\texttt{Y}\ ,\ [\texttt{X},\texttt{Y}]\}$$

4.15.3 GAUSS

GAUSS has two arguments: a quadratic form q and a vector whose components are the quadratic form's variables.
GAUSS returns a list of 4 elements: the diagonal elements of a diagonal matrix B (obtained expressing q as a sum of squares), the matrix of change of base Q, q expressed as a sum of squares, and the vector denoting the form's variables.
We have (if we denote as A the matrix associated with q) :

$$^tQ\cdot B\cdot Q=A$$

Type:

$$\texttt{GAUSS}(2\cdot\texttt{X}\cdot\texttt{Y}\ ,\ [\texttt{X},\texttt{Y}])$$

You obtain:

$$\{[\frac{1}{2},-2]\ ,\ \begin{bmatrix}1 & 1\\ -\frac{1}{2} & \frac{1}{2}\end{bmatrix}\ ,\ -2.(\frac{\texttt{Y}-\texttt{X}}{2})^2+\frac{1}{2}.(\texttt{Y}+\texttt{X})^2\ ,\ [\texttt{X},\texttt{Y}]\}$$

4.15.4 SYLVESTER

SYLVESTER has one argument: a symmetric matrix representing a quadratic form q.
SYLVESTER returns a list of two elements: the diagonal elements of the diagonal matrix B (obtained expressing q as a sum of squares) and the matrix of change of base Q.
We have:

$$^tQ\cdot B\cdot Q=A$$

Type:

$$\texttt{SYLVESTER}([[0,1],[1,0]])$$

You obtain:

$$\{[\frac{1}{2},-2]\ ,\ \begin{bmatrix}1 & 1\\ -\frac{1}{2} & \frac{1}{2}\end{bmatrix}\}$$

4.16 Functions of multiple variables

4.16.1 DERIV

DERIV has two arguments: an application F from R^n in R and a vector of R^n denoting the variable names.
DERIV returns the gradient of F ($[\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}]$ if $n = 3$).
Type:

$$\texttt{DERIV}(2 \cdot \texttt{X}^2 \cdot \texttt{Y} - \texttt{X} \cdot \texttt{Z}^3, [\texttt{X}, \texttt{Y}, \texttt{Z}])$$

After simplification, you obtain:

$$[4 \cdot \texttt{Y} \cdot \texttt{X} - \texttt{Z}^3, 2 \cdot \texttt{X}^2, -(3 \cdot \texttt{Z}^2 \cdot \texttt{X})]$$

4.16.2 LAPL

LAPL has two arguments: an application F from R^n in R and a vector of R^n denoting the variable names.
LAPL returns the laplacian of F ($\frac{\partial^2 F}{\partial X^2} + \frac{\partial^2 F}{\partial Y^2} + \frac{\partial^2 F}{\partial Z^2}$ if $n = 3$).
Type:

$$\texttt{LAPL}(2.\texttt{X}^2.\texttt{Y} - \texttt{X}.\texttt{Z}^3 \ , \ [\texttt{X}, \texttt{Y}, \texttt{Z}])$$

You obtain:

$$4.\texttt{Y} - 6.\texttt{X}.\texttt{Z}$$

4.16.3 HESS

HESS has two arguments: an applications F from R^n in R and a vector of R^n denoting the variable names.
HESS returns a list containing the hessian of F, the gradient of F and the vector of the variable names.
Type:

$$\texttt{HESS}(2.\texttt{X}^2.\texttt{Y} - \texttt{X}.\texttt{Z} \ , \ [\texttt{X}, \texttt{Y}, \texttt{Z}])$$

You obtain:

$$\{ \begin{bmatrix} 4.\texttt{Y} & 4.\texttt{X} & -1 \\ 4.\texttt{X} & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \ , \ [4.\texttt{X}.\texttt{Y} - \texttt{Z}, 2.\texttt{X}^2, -\texttt{X}] \ , \ [\texttt{X}, \texttt{Y}, \texttt{Z}] \}$$

Now, to obtain the critical points of F, in RPN mode you can type:

SOLVE

directly, because on the stack you have: $[4.X.Y - Z, 2.X^2, -X]$ and $[X, Y, Z]$
In ALGEBRAIC mode, you must enter instead:

$$\texttt{SOLVE}([4.X.Y - Z, 2.X^2, -X]\ ,\ [X, Y, Z])$$

4.16.4 DIV

DIV has two arguments: a vectorial function F (application from R^n in R^n) and a vector of R^n denoting the variable names.
DIV returns the divergence of F.

$$\texttt{DIV}([A, B, C], [X, Y, Z]) = \frac{\partial A}{\partial X} + \frac{\partial B}{\partial Y} + \frac{\partial C}{\partial Z} \text{ (here } n = 3)$$

Type:

$$\texttt{DIV}([X \cdot Z, -Y^2, 2 \cdot X^Y], [X, Y, Z])$$

You obtain:

$$Z - 2 \cdot Y$$

4.16.5 CURL

Here $n = 3$.
CURL has two arguments: a vectorial function F (application from R^3 in R^3) and a vector of R^3 denoting the variable names.
CURL returns the rotor of F.

$$\texttt{CURL}([A, B, C], [X, Y, Z]) = [\frac{\partial C}{\partial Y} - \frac{\partial B}{\partial Z}, \frac{\partial A}{\partial Z} - \frac{\partial C}{\partial X}, \frac{\partial B}{\partial X} - \frac{\partial A}{\partial Y}]$$

Type:

$$\texttt{CURL}([X.Z, -Y^2, 2.X^Y], [X, Y, Z])$$

You obtain:

$$[2.X^2, X - 2.Y.2X, 0]$$

4.17 Equations

4.17.1 EXLR

EXLR has an equation as argument.
EXLR returns a list containing the left and right hand sides of the equation.

Type:

$$\texttt{EXLR}(\texttt{A} = \texttt{B})$$

You obtain:

$$\{\texttt{A}, \texttt{B}\}$$

4.17.2 SOLVEVX

SOLVEVX has as argument either an equation between two expression of the variable stored in VX, or an expression (=0 is assumed then).
SOLVEVX solves the equation.
Example 1:
Type:

$$\texttt{SOLVEVX}(\texttt{X}^4 - 1 = 3)$$

In real mode, you obtain:

$$\{\texttt{X} = -\sqrt{2},\ \texttt{X} = \sqrt{2}\}$$

In complex mode, you obtain:

$$\{\texttt{X} = -\sqrt{2},\ \texttt{X} = \sqrt{2},\ \texttt{X} = -(\texttt{i} \cdot \sqrt{2}),\ \texttt{X} = \texttt{i} \cdot \sqrt{2}\}$$

Example 2:
Type:

$$\texttt{SOLVEVX}((\texttt{X} - 2).\texttt{SIN}(\texttt{X}))$$

In real mode, you obtain:

$$\{\texttt{X} = -(2 \cdot \pi \cdot \texttt{n}_1),\ \texttt{X} = 2 \cdot \pi \cdot \texttt{n}_1,\ \texttt{X} = 2\}$$

4.17.3 SOLVE

SOLVE has as argument either an equation between two expressions or an expression (=0 is assumed then), and the name of a variable.
SOLVE can also solve a system of equations: to do this, put the equations into a vector and the variable names into another vector.
SOLVE solves either the equation or the system of equations.
Example 1:

Type:

$$\texttt{SOLVE}(\texttt{X}^4 - 1 = 3, \texttt{X})$$

In real mode, you obtain:

$$\{\texttt{X} = -\sqrt{2},\ \texttt{X} = \sqrt{2}\}$$

In complex mode, you obtain:

$$\{\texttt{X} = -\sqrt{2},\ \texttt{X} = \sqrt{2},\ \texttt{X} = -(\texttt{i} \cdot \sqrt{2}),\ \texttt{X} = \texttt{i} \cdot \sqrt{2}\}$$

Example 2:
Type:

$$\texttt{SOLVE}([\texttt{X} + \texttt{Y} = 1, \texttt{X} - \texttt{Y}], [\texttt{X}, \texttt{Y}])$$

You obtain:

$$\{[\texttt{X} = \frac{1}{2},\ \texttt{Y} = \frac{1}{2}]\}$$

4.17.4 ISOL

ISOL isolates a variable in an expression or equation (the variable must appear only once). This command is the same as the HP48 one.
ISOL has two arguments: either an expression or an equation, and the name of the variable to isolate.
Type:

$$\texttt{ISOL}(\texttt{X}^4 - 1 = 3, \texttt{X})$$

In real mode, you obtain:

$$\{\texttt{X} = -\sqrt{2},\ \texttt{X} = \sqrt{2}\}$$

In complex mode, you obtain:

$$\{\texttt{X} = -\sqrt{2},\ \texttt{X} = \sqrt{2},\ \texttt{X} = -(\texttt{i} \cdot \sqrt{2}),\ \texttt{X} = \texttt{i} \cdot \sqrt{2}\}$$

Warning: if flag 01 (`Principal value`) is set, ISOL always returns only one solution.

4.18 Linear systems

In this paragraph, we call "augmented matrix" of the system $A \cdot X = B$ (or matrix "representing" the system $A \cdot X = B$), the matrix obtained augmenting the matrix A to the right with the column vector B.

4.18.1 REF

REF solves a linear system of equations written in matrix form:

$$\mathtt{A} \cdot \mathtt{X} = \mathtt{B}$$

The REF command is in the MATRICES (blue-shift 5) menu, 5 LINEAR SYST... sub-menu.
The argument of REF is the augmented matrix of the system (the matrix obtained augmenting matrix A to the right with the column vector B).
The result is a matrix [A1,B1] : A1 has zeros under its principal diagonal, and the solutions of:

$$\mathtt{A1} \cdot \mathtt{X} = \mathtt{B1}$$

are the same as:

$$\mathtt{A} \cdot \mathtt{X} = \mathtt{B}$$

For example, to solve the system:

$$\begin{cases} 3 \cdot x + y & = & -2 \\ 3 \cdot x + 2 \cdot y & = & 2 \end{cases}$$

Type (using blue-shift EQW (MTRW) to enter the matrix):

$$\mathtt{REF}([[3, 1, -2][3, 2, 2]])$$

You obtain:

$$\begin{bmatrix} 1 & \frac{1}{3} & \frac{-2}{3} \\ 0 & 1 & 4 \end{bmatrix}$$

4.18.2 rref

rref solves a linear system of equations written in matrix form:

$$\mathtt{A} \cdot \mathtt{X} = \mathtt{B}$$

The rref command is in the MATRICES (blue-shift 5) menu, 5 LINEAR SYST... sub-menu.

The argument of `rref` is the augmented matrix of the system (the matrix obtained augmenting matrix `A` to the right with the column vector `B`).
The result is a list containing the list of pivot elements used by the command and a matrix `[A1,B1]` : `A1` has zeros both above and under its principal diagonal, and the solutions of:

$$\texttt{A1} \cdot \texttt{X} = \texttt{B1}$$

are the same as:

$$\texttt{A} \cdot \texttt{X} = \texttt{B}$$

It is interesting to use `rref` in step-by-step mode, setting the `Step/Step (MODE cas chk)` flag.
For example, to solve the system:

$$\begin{cases} 3 \cdot x + y & = & -2 \\ 3 \cdot x + 2 \cdot y & = & 2 \end{cases}$$

Type (using `blue-shift EQW (MTRW)` to enter the matrix) :

$$\texttt{rref}([[3,1,-2][3,2,2]])$$

You obtain:

$$\left\{ \texttt{Pivots : \{1 1.\}} \begin{bmatrix} 3 & 0 & -6 \\ 0 & 1 & 4 \end{bmatrix} \right\}$$

4.18.3 RREF

`RREF` is the same as `rref`, but it does not return the pivots.
`RREF` is the HP48 command; this is its result in step-by-step mode, on the HP49, to solve the system:

$$\begin{cases} 3 \cdot x + y & = & -2 \\ 3 \cdot x + 2 \cdot y & = & 2 \end{cases}$$

Type (using `blue-shift EQW (MTRW)` to enter the matrix):

$$\texttt{RREF}([[3,1,-2][3,2,2]])$$

You obtain:

$\mathtt{L_2} = \mathtt{L_2} - \mathtt{L_1}$

$$\begin{bmatrix} 3 & 1 & -2 \\ 3 & 2 & 2 \end{bmatrix}$$

after `ok`:

$L_1 = L_1 - L_2$

$$\begin{bmatrix} 3 & 1 & -2 \\ 0 & 1 & 4 \end{bmatrix}$$

after `ok`:

Reduction result:

$$\begin{bmatrix} 3 & 0 & -6 \\ 0 & 1 & 4 \end{bmatrix}$$

after `ok` the result (with all 1 on the diagonal of $A1$) is stored in the history:

$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 4 \end{bmatrix}$$

4.18.4 LINSOLVE

`LINSOLVE` solves a linear system of equations.
The `LINSOLVE` command is in the `MATRICES (blue-shift 5)`, `5.LINEAR SYST...` sub-menu, position 1.
Type:

`LINSOLVE()`

Then, enter `MATRIXWRITER` by pressing `blue-shift EQW (MTRW)` (with the cursor positioned between the two parentheses of `LINSOLVE`).

Set the `vect` menu option (if it isn't already set), and enter the equations (possibly with the help of `EQW`).

$2 \cdot X + Y + Z = 1$ `ENTER`

$X + Y + 2 \cdot Z = 1$ `ENTER`

$X + 2 \cdot Y + Z = 4$ `ENTER`

`ENTER`
Then, enter the unknown variables:

`[X Y Z]`

and ENTER

If step-by-step mode is enabled (MODE cas Step/Step), you obtain:
L2=2L2-L1
$$\begin{bmatrix} 2 & 1 & 1 & -1 \\ 1 & 1 & 2 & -1 \\ 1 & 2 & 1 & -4 \end{bmatrix}$$
after ok:
L3=2L3-L1
$$\begin{bmatrix} 2 & 1 & 1 & -1 \\ 0 & 1 & 3 & -1 \\ 1 & 2 & 1 & -4 \end{bmatrix}$$
an so on... Last, you obtain: Result
$$\begin{bmatrix} 8 & 0 & 0 & 4 \\ 0 & 8 & 0 & -20 \\ 0 & 0 & 8 & -4 \end{bmatrix}$$
after ok,

$$\{\mathtt{X} = -\frac{1}{2}\ \mathtt{Y} = \frac{5}{2}\ \mathtt{Z} = -\frac{1}{2}\}$$

is stored into the history.

4.19 Differential equations

4.19.1 LDEC

LDEC directly solves linear differential equations; in order to do this, LAP and ILAP (see 4.19.3) are used internally.

For second order linear equations, the arguments are the second member of the equation and the characteristic equation.

For first order systems of linear equations, the arguments are the second member of the equation (a vector) and the matrix of the system.
Example 1:

Solve:

$$y'' - 6 \cdot y' + 9 \cdot y \ = \ x \cdot e^{3 \cdot x}$$

Type:

$$\texttt{LDEC}(\texttt{X}\cdot\texttt{EXP}(3\cdot\texttt{X}),\texttt{X}^2-6\cdot\texttt{X}+9)$$

You obtain:

$$(\frac{\texttt{X}^3}{6}-(3\cdot\texttt{C0}-\texttt{C1})\cdot\texttt{X}+\texttt{C0})\cdot\texttt{EXP}(3\cdot\texttt{X})$$

C0 and C1 are arbitrary integration constants ($\texttt{y}(0)=\texttt{C0},\ \texttt{y}\prime(0)=\texttt{C1}$).
Example 2:
Solve:

$$Z'=\begin{bmatrix}0 & 1\\ -9 & 6\end{bmatrix}.Z+\begin{bmatrix}0\\ X.EXP(3.X)\end{bmatrix}$$

This is the same example as before, with $Z=[y,y\prime]$.
Type:

$$\texttt{LDEC}([0,\texttt{X}\cdot\texttt{EXP}(3\cdot\texttt{X})],[[0,1][-9,6]])$$

You obtain:

$$[(\frac{\texttt{X}^3}{6}-(3\cdot\texttt{V1}-\texttt{V2})\cdot\texttt{X}+\texttt{V1})\cdot\texttt{EXP}(3\cdot\texttt{X}),$$
$$(\frac{\texttt{X}^3}{2}+\frac{\texttt{X}^2}{2}-(9\cdot\texttt{V1}-3\cdot\texttt{V2})\cdot\texttt{X}+\texttt{V2})\cdot\texttt{EXP}(3\cdot\texttt{X})]$$

V1 and V2 are arbitrary integration constants ($\texttt{Z}(0)=[\texttt{V1},\texttt{V2}]$).

4.19.2 DESOLVE and SUBST

The DESOLVE command is in the S.SLV (blue-shift 7) menu, at position 1, or in the CALC (blue-shift 4) menu, DIFFERENTIAL EQNS... sub-menu.
DESOLVE solves other types of differential equations.
Its arguments are: the differential equation (here $y\prime$ is written as d1Y(X) and y is written as Y(X)).
Example 1 :
Solve:

$$y''+y=\cos(x)\ \ y(0)=c_0\ \ y'(0)=c_1$$

Type:

$$\texttt{DESOLVE}(\texttt{d1d1Y}(\texttt{X})+\texttt{Y}(\texttt{X})=\texttt{COS}(\texttt{X}),\texttt{Y}(\texttt{X}))$$

You obtain:

$$\texttt{Y}(\texttt{X})=\texttt{C0}\cdot\texttt{COS}(\texttt{X})+\frac{\texttt{X}+2\cdot\texttt{C1}}{2}\cdot\texttt{SIN}(\texttt{X})$$

Now, you can assign a value to the integration constants, using the SUBST command; you can find it in the ALG (red-shift 4) menu, at position 6. If you want the solutions that satisfy $y(0) = 1$, you type:

$$\texttt{SUBST}(\texttt{Y(X)} = \texttt{C0} \cdot \texttt{COS(X)} + \frac{\texttt{X} + 2 \cdot \texttt{C1}}{2} \cdot \texttt{SIN(X)}, \texttt{C0} = 1)$$

You obtain:

$$\texttt{Y(X)} = \frac{2 \cdot \texttt{COS(X)} + (\texttt{X} + 2 \cdot \texttt{C1}) \cdot \texttt{SIN(X)}}{2}$$

Example 2 :
Solve:

$$y'' + y = \cos(x) \;\; y(0) = 1 \;\; y'(0) = c_1$$

To have the solutions satisfying $y(0) = 1$ you can also directly type:

$$\texttt{DESOLVE}([\texttt{d1d1Y(X)} + \texttt{Y(X)} = \texttt{COS(X)}, \texttt{Y(0)} = 1], \texttt{Y(X)})$$

In this case, you obtain:

$$\texttt{Y(X)} = \texttt{COS(X)} + \frac{\texttt{X} + 2 \cdot \texttt{C1}}{2} \cdot \texttt{SIN(X)}$$

4.19.3 LAP ILAP

You can find these commands in the CALC (blue-shift 4) menu, 3.DIFFERENTIAL... sub-menu, at positions 2 and 3. Laplace tranform (LAP) and inverse Laplace transform (ILAP) are useful to solve linear differential equations with constant coefficients, for example:

$$y'' + p.y' + q.y \;\; = \;\; f(x) \;\; y(0) = a \;\; y'(0) = b$$

The following relations hold:

$$\texttt{LAP(Y)(P)} = \int_0^{+\infty} \texttt{e}^{-\texttt{P.X}} \texttt{Y(X)dX}$$

$$\texttt{ILAP(F)(T)} = \frac{1}{2.\texttt{i}.\pi} \int_{\texttt{C}} \texttt{e}^{\texttt{Z.T}} \texttt{dZ}$$

where C is a closed contour enclosing the poles of F
The following property is used:

$$\texttt{LAP(Y')(P)} = -\texttt{Y(0)} + \texttt{P} \cdot \texttt{LAP(Y)(P)}$$

The solution is then:

$$\texttt{ILAP}(\frac{\texttt{LAP(F(X))}+(\texttt{X}+\texttt{P})\cdot\texttt{A}+\texttt{B}}{\texttt{X}^2+\texttt{P}\cdot\texttt{X}+\texttt{Q}})$$

Example:
Solve:

$$y'' - 6\cdot y' + 9\cdot y \;=\; x\cdot e^{3\cdot x}$$

$$y(0) = a$$

$$y'(0) = b$$

Type:

$$\texttt{LAP(X}\cdot\texttt{EXP(3}\cdot\texttt{X))}\ \texttt{ENTER}$$

You obtain:

$$\frac{1}{\texttt{X}^2-6\cdot\texttt{X}+9}$$

Type:

$$\texttt{ILAP}(\frac{\texttt{ANS(1)}+(\texttt{X}-6)\cdot\texttt{A}+\texttt{B}}{\texttt{X}^2-6\cdot\texttt{X}+9})$$

You obtain the solution y :

$$(\frac{\texttt{X}^3}{6}-(3\cdot\texttt{A}-\texttt{B})\cdot\texttt{X}+\texttt{A})\cdot\texttt{EXP(3}\cdot\texttt{X)}$$

4.20 Other functions

4.20.1 EPSX0

EPSX0 has as argument an expression; in the expression, it replaces all numeric values whose magnitude is smaller than EPS with zero and returns the result.
Type:

$$\texttt{EPSX0(0.001+X)}$$

You obtain (when EPS=0.01) :

$$0+\texttt{X}$$

You obtain (when EPS=0.0001) :

$$.001+\texttt{X}$$

4.20.2 LVAR

LVAR has an expression as argument, and returns a list containing the expression and a vector whose components are the independent variables of the expression.
Type:

LVAR(X.Y.SIN(X))

You obtain:

{X.Y.SIN(X), [SIN(X), X, Y]}

4.20.3 LNAME

LNAME has an expression as argument, and returns a vector whose components are the symbolic variable names the expression contains.
Type:

LNAME(X.Y.SIN(X))

You obtain:

[X, Y]

4.20.4 XNUM

XNUM has either an expression or an array as argument.
XNUM enables approximate mode and returns the numeric approximation of its argument.
Type:

XNUM($\sqrt{2}$)

You obtain:

1.41421356237

4.20.5 XQ

XQ has a real numeric expression as argument.
XQ enables exact mode and returns either a rational or a real approximation of the expression.
Type:

XQ(1.41422)

You obtain:

$$\frac{66441}{46981}$$

Type:

XQ(1.414213562)

You obtain:

$$\sqrt{2}$$

4.21 New commands

4.21.1 qr

qr has a square matrix as argument.
qr factors the matrix as Q*R, where Q is an orthogonal matrix and R is a triangular matrix.
For example, enter:

$$\texttt{qr}(\begin{bmatrix} 3 & 5 \\ 4 & 5 \end{bmatrix})$$

You obtain:

$$\{\begin{bmatrix} \frac{3}{5} & \frac{4}{5} \\ \frac{4}{5} & -\frac{3}{5} \end{bmatrix}, \begin{bmatrix} 5 & 7 \\ 0 & 1 \end{bmatrix}\}$$

4.21.2 GRAMSCHMIDT

GRAMSCHMIDT has two arguments: a vector representing a base of a vectorial space and the definition of a scalar product in that space, expressed as a function.
GRAMSCHMIDT returns an orthonormal base of the vectorial space with respect to the given scalar product.
Example:
In the vectorial space of polynomials of degree less than 2, we consider the scalar product defined by:

$$P \cdot Q = \int_{-1}^{1} P(x) \cdot Q(x) dx$$

We first express the scalar product as a function named PS :
$\ll\!\!\rightarrow$ P Q $\ll$ PREVAL(INTVX(P * Q), −1, 1) $\gg \gg$ STO $\triangleright$ PS
Next, we type:

GRAMSCHMIDT([1, 1 + X], RCL('PS'))

We obtain:

$$[\frac{1}{\sqrt{2}}, \frac{\sqrt{6} \cdot \texttt{X}}{2}]$$

4.21.3 SYST2MAT

SYST2MAT has two arguments: a vector containing a system of linear equations, and a vector whose elements are the system's variables.
SYST2MAT rewrites the system in matrix notation, and returns the matrix.
For example, type:

$$\texttt{SYST2MAT}([\texttt{X}+\texttt{Y}, \texttt{X}-\texttt{Y}=2], [\texttt{X}, \texttt{Y}])$$

You obtain:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & -2 \end{bmatrix}$$

Warning: you must purge the system's variables (X and Y in this example) beforehand.

4.21.4 CHOLESKY

CHOLESKY has as argument a square matrix M, positive by definition.
CHOLESKY returns an upper triangular matrix P so that:
${}^{t}\texttt{P} * \texttt{P} = \texttt{M}$
For example, type:

$$\texttt{CHOLESKY}(\begin{bmatrix} 1 & 1 \\ 1 & 5 \end{bmatrix})$$

You obtain:

$$\begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$$

4.21.5 DIAGMAP

DIAGMAP has two arguments: a diagonalizable matrix and a function defining an holomorphic operator.
DIAGMAP applies the operator to the matrix, and returns the result.
Warning! The matrix must be diagonalizable.
For example, we define the function PH as follows :
$\ll\!\!\rightarrow \texttt{M} \ll \texttt{EXP}(\texttt{M}) \gg \gg \texttt{STO} \triangleright \texttt{PH}$
Next, we type:

$$\texttt{DIAGMAP}(\begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}, \ll\!\!\rightarrow \texttt{M} \ll \texttt{EXP}(\texttt{M}) \gg\gg)$$

or :

$$\texttt{DIAGMAP}(\begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}, \texttt{RCL}(\texttt{'PH'}))$$

We obtain:

$$\begin{bmatrix} \texttt{EXP}(1) & -\texttt{EXP}(1)+\texttt{EXP}(2) \\ 0 & \texttt{EXP}(2) \end{bmatrix}$$

4.21.6 ISOM

ISOM has as argument a matrix representing a two or three-dimensional linear isometry.
ISOM returns the list of the isometry's characteristics elements and either +1 (for direct isometries) or -1 (for indirect isometries).
For example, type:

$$\texttt{ISOM}(\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix})$$

You obtain:

$$\{[1\ 0\ -1]\ -1\}$$

This result means that the isometry is a simmetry with respect to the plane $x - z = 0$.
If you type:

$$\texttt{ISOM}(\frac{\sqrt{2}}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix})$$

You obtain:

$$\{\frac{\pi}{4}, 1\}$$

therefore, this isometry is a rotation of $\frac{\pi}{4}$ radians.

4.21.7 MKISOM

In a three-dimensional space, MKISOM has the list of the characteristics elements of an isometry, and either +1 (denoting a direct isometry) or -1 (denoting an indirect isometry) as arguments,
In a two-dimensional space, MKISOM has the characteristic element of an isometry (either an angle or a vector) and either +1 (denoting a direct isometry) or -1 (denoting an indirect isometry) as arguments.

MKISOM returns the matrix representing the given isometry.
For example, type:

$$\texttt{MKISOM}(\{[-1,2,-1],\pi\},1)$$

You obtain the matrix of a rotation with axis $[-1,2,-1]$ and angle π:

$$\frac{1}{3}\begin{bmatrix} -2 & -2 & 1 \\ -2 & 1 & -2 \\ 1 & -2 & -2 \end{bmatrix}$$

For example, type:

$$\texttt{MKISOM}(\{\pi\},-1)$$

You obtain the matrix of a symmetry with respect to the origin:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Type:

$$\texttt{MKISOM}(\{[1,1,1],\frac{\pi}{3}\},-1)$$

You obtain the matrix of a rotation with axis $[1,1,1]$ and angle $\frac{\pi}{3}$ combined with a symmetry with respect to the plane $x+y+z=0$:

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

Type:

$$\texttt{MKISOM}(\frac{\pi}{2},1)$$

You obtain the matrix of a rotation of $\frac{\pi}{2}$ radians in two dimensions:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Type:

$$\texttt{MKISOM}([1,1],-1)$$

You obtain the matrix of a symmetry with respect to $y=x$ in two dimensions:

$$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

4.21.8 KER

KER has as argument a matrix representing a linear application f in terms of the standard basis.
KER returns a list of vectors; they are a basis of the kernel of f.
For example, type:

$$\texttt{KER}(\begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & 4 \end{bmatrix})$$

You obtain:

$$\{[1, 1, -1]\}$$

4.21.9 IMAGE

IMAGE has as argument a matrix representing a linear application f in terms of the standard basis.
IMAGE returns a list of vectors; they are a basis of the image of f.
For example, type:

$$\texttt{IMAGE}(\begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & 4 \end{bmatrix})$$

You obtain:

$$\{[1, 0, -1], [1, 1, 2]\}$$

4.21.10 IBASIS

IBASIS has as arguments two lists of vectors, defining two vectorial spaces.
IBASIS returns a list containing the vectors of a basis of the intersection between these two vectorial spaces.
For example, type:

$$\texttt{IBASIS}(\{[1, 2]\}, \{[2, 4]\})$$

You obtain:

$$\{[1, 2]\}$$

4.21.11 BASIS

BASIS has as argument a list of vectors, defining a vectorial subspace of R^n.
BASIS returns a list containing the vectors of a basis of the vectorial subspace.
For example, type:

$$\texttt{BASIS}(\{[1, 2, 3], [1, 1, 1], [2, 3, 4]\})$$

You obtain:

$$\{[1, 0, -1],\ [0, 1, 2]\}$$

4.21.12 AUGMENT

AUGMENT has as arguments two vectors, or two lists, or a list and an element. AUGMENT concatenates its arguments.
For example, type:

$$\texttt{AUGMENT}(\{1, 2\}, 3)$$

You obtain:

$$\{1\ 2\ 3\}$$

4.21.13 CYCLOTOMIC

CYCLOTOMIC has an integer n as argument.
CYCLOTOMIC returns the cyclotomic polynomial of order n. This is a polynomial having the n-th pritmitive roots of the unity as zeros.
For example, when $n = 4$ the fourth roots of the unity are: $\{1, i, -1, -i\}$; among them, the primitive roots are: $\{i, -i\}$.
Therefore, the cyclotomic polynomial of order 4 is $(X - i).(X + i) = X^2 + 1$.
Another example; if you type:

$$\texttt{CYCLOTOMIC}(20)$$

You obtain:

$$\texttt{X}^8 - \texttt{X}^6 + \texttt{X}^4 - \texttt{X}^2 + 1$$

4.21.14 STURM

STURM has a polynomial P as argument.
STURM returns a list containing the Sturm's sequences of P and their multiplicities.
The Sturm sequence $R_1, R_2, ...$ can be obtained from a square-free factor F of P as follows:
R_1 is the opposite of the remainder of the euclidean division of F by F';
then, R_2 is the opposite of the remainder of the euclidean division of F' by R_1,
and so on, until $R_k = 0$.
For example, type:

$$\texttt{STURM}(\texttt{X}^3 + 1)$$

You obtain:

$$\{[1],\ -1.,\ [X^3+1,\ -3X^2-1],\ 1.\}$$

The first element of the list denotes that the denominator of P (that is, the element with -1 power), is 1.

4.21.15 STURMAB

STURMAB has three arguments: a polynomial P and two numbers: a and b.
STURMAB returns a list containing an element with the same sign as $P(a)$ and the number of zeros of P in $[a, b[$.
For example, type:

$$\mathtt{STURMAB}(X^2 \cdot (X^3+2), -2, 0)$$

You obtain:

$$\{-6,\ 1\}$$

4.21.16 P2C

Permutations are defined with the image list $\{P(1), P(2)...P(n)\}$.
For example, defininig the permutation P as $P = \{3, 2, 1\}$ means that:
$P(1) = 3,\ P(2) = 2,\ P(3) = 1$.
A cycle is denoted with a list containing the images of an element through the cycle; for example, defining a cycle C as $C = \{3, 2, 1\}$ means that:
$C(3) = 2,\ C(2) = 1,\ C(1) = 3$.
Accordingly, a decomposition into cycles is denoted with a list of lists.
P2C has a permutation as argument.
P2C returns its decomposition into cycles and its signature.
For example, type:

$$\mathtt{P2C}(\{3,4,5,2,1\})$$

You obtain:

$$\{\{\{1,3,5\},\{2,4\}\},-1\}$$

4.21.17 C2P

C2P has a list of cycles as argument.
C2P returns the permutation having the input list as decomposition into cycles (see also P2C).
For example, type:

$$\mathtt{C2P}(\{\{1,3,5\},\{2,4\}\})$$

You obtain:

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

4.21.18 CIRC

CIRC has two permutations as arguments.
CIRC returns the permutation obtained by composition: (1^{st}argument$\circ 2^{\text{nd}}$argument).
For example, type:

$$\texttt{CIRC}(\{3, 4, 5, 2, 1\}, \{2, 1, 4, 3, 5\})$$

You obtain:

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

4.21.19 FDISTRIB

FDISTRIB performs a full distribution of multiplication with respect to addition in a single step.
For example, type:

$$\texttt{FDISTRIB}((\texttt{X} + 1) \cdot (\texttt{X} + 2) \cdot (\texttt{X} + 3))$$

You obtain:

$$\texttt{X}^3 + 6 \cdot \texttt{X}^2 + 11 \cdot \texttt{X} + 6$$

4.21.20 DISTRIB

DISTRIB applies the distributive property of multiplication with respect to addition once.
DISTRIB, when invoked multiple times, allows you to perform a full distribution of multiplication with respect to addition step-by-step.
For example, type:

$$\texttt{DISTRIB}((\texttt{X} + 1) \cdot (\texttt{X} + 2) \cdot (\texttt{X} + 3))$$

You obtain:

$$\texttt{X} \cdot (\texttt{X} + 2) \cdot (\texttt{X} + 3) + 1 \cdot (\texttt{X} + 2) \cdot (\texttt{X} + 3)$$

4.21.21 POWEXPAND

POWEXPAND rewrites a power as a product.
For example, type:

$$\texttt{POWEXPAND}((\texttt{X}+1)^3)$$

You obtain:

$$(\texttt{X}+1)*(\texttt{X}+1)*(\texttt{X}+1)$$

You can also expand $(x+1)^3$ step-by-step, by invoking DISTRIB many times on the result obtained above.

4.21.22 SIMPLIFY

SIMPLIFY attempts to simplify the expression given as argument automatically.
Like all other automatic simplification tools, you should not demand miracles to this command. For example, type:

$$\texttt{SIMPLIFY}(\frac{\texttt{SIN}(3\cdot\texttt{X})+\texttt{SIN}(7\cdot\texttt{X})}{\texttt{SIN}(5\cdot\texttt{X})})$$

After simplification, you obtain:

$$4\cdot\texttt{COS}(\texttt{X})^2-2$$

4.21.23 EXP2POW

EXP2POW rewrites an expression like

$$\exp(n\times\ln(x))$$

as a power of x.
For example, type:

$$\texttt{EXP2POW}(\texttt{EXP}(\texttt{N}*\texttt{LN}(\texttt{X})))$$

You obtain:

$$\texttt{X}^{\texttt{N}}$$

Notice the difference with respect to LNCOLLECT :
We have:
$\texttt{LNCOLLECT}(\texttt{EXP}(\texttt{N}*\texttt{LN}(\texttt{X})))=\texttt{EXP}(\texttt{N}*\texttt{LN}(\texttt{X}))$
$\texttt{LNCOLLECT}(\texttt{EXP}(\texttt{LN}(\texttt{X})/3)) = \texttt{EXP}(\texttt{LN}(\texttt{X})/3)$
But:
$\texttt{EXP2POW}(\texttt{EXP}(\texttt{LN}(\texttt{X})/3)) = \texttt{X}^{\frac{1}{3}}$

4.21.24 MSLV

MSLV solves numerically a system of non-polynomial equations.
MSLV has three vectors as arguments: a vector containing the equations, a vector containing the system's variables, and a vector containing an initial guess for the solution.
MSLV returns a vector containing an approximate solution of the given system of equations.
While the command is running, the first display line shows the last estimate $\overrightarrow{V}$, and the second line shows the modulo of $\Delta\overrightarrow{V}$
For example, type:

$$\texttt{MSLV}([\texttt{'SIN(X)+Y'},\texttt{'X+SIN(Y)=1'}],[\texttt{X},\texttt{Y}],[0,0])$$

You obtain:

$$[1.82384112611,\ -.968154636174]$$

4.21.25 PMINI

PMINI has a matrix A as argument.
PMINI returns another matrix, whose first "non-zero row" is the minimal polynomial of A.
For example, type:

$$\texttt{PMINI}([[1,0],[0,1]])$$

In step-by-step mode, you obtain:
L2=L2-L1

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & X \\ 1 & 0 & 0 & 1 & X^2 \end{bmatrix}$$

L3=L3-L1

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & X-1 \\ 1 & 0 & 0 & 1 & X^2 \end{bmatrix}$$

Reduction result

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & X-1 \\ 0 & 0 & 0 & 0 & X^2-1 \end{bmatrix}$$

So, the minimal polynomial of A: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ is:

$$\texttt{X}-1$$

4.21.26 IBERNOULLI

IBERNOULLI has an integer n as argument.
IBERNOULLI returns the n^{ty} Bernoulli's number $B(n)$.
The following relation holds:

$$\frac{t}{e^t - 1} = \sum_{n=0}^{+\infty} \frac{B(n)}{n!} t^n$$

Remember that the Bernoulli's polynomials B_k are defined as:

$$B_0 = 1$$

$$B_k{}'(x) = kB_{k-1}(x)$$

$$\int_0^1 B_k(x)dx = 0$$

Bernoulli's numbers are defined as:
$B(n) = B_n(0)$
For example, type:

```
IBERNOULLI(6)
```

You obtain:

$$\frac{1}{42}$$

4.21.27 GAMMA

Returns the value of the Γ function at the given point.
The Γ function is defined as:

$$\Gamma(x) = \int_0^{+\infty} e^{-t} t^{x-1} dt$$

We have:

$$\Gamma(1) = 1$$

$$\Gamma(x+1) = x \cdot \Gamma(x)$$

For example, type:

```
GAMMA(5)
```

You obtain:

```
24
```

Type:

$$\texttt{GAMMA}(\frac{1}{2})$$

You obtain:

$$\sqrt{\pi}$$

4.21.28 PSI

PSI has two numbers a and n as arguments.
PSI returns the value of the n-th derivative of the Digamma function at a.
The Digamma function is defined as the derivative of $\ln(\Gamma(x))$.
For example, type:

$$\texttt{PSI}(3,1)$$

You obtain:

$$-\frac{5}{4}+\frac{1}{6}.\pi^2$$

4.21.29 Psi

Psi has as argument a number, a.
Psi returns the value of the Digamma function at a.
The Digamma function is defined as the derivative of $\ln(\Gamma(x))$, so we have: PSI(a,0)=Psi(a).
For example, type:

$$\texttt{Psi}(3)$$

You obtain:

$$.922784335098$$

4.21.30 RESULTANT

RESULTANT has two polynomials as arguments.
RESULTANT returns the resultant of the two polynomials. The resultant of two polynomials is the last non-null remainder of the Euclide algorithm, and is also the determinant of their Sylvester matrix S.
The Sylvester matrix S of two polynomials $A[X]=\sum_{i=0}^{i=n} a_i X^i$ and $B[X]=\sum_{i=0}^{i=m} a_i X^i$ is a square matrix with $m+n$ rows and columns; its first m rows

are made from the coefficients of $A[X]$:

$$\begin{pmatrix} s_{11}=a_n & s_{12}=a_{n-1} & \cdots & s_{1(n+1)}=a_0 & 0 & \cdots & 0 \\ s_{21}=0 & s_{22}=a_n & \cdots & s_{2(n+1)}=a_1 & s_{2(n+2)}=a_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ s_{m1}=0 & s_{m2}=0 & \cdots & s_{m(n+1)}=a_{m-1} & s_{m(n+2)}=a_{m-2} & \cdots & a_0 \end{pmatrix}$$

and the following n rows are made in the same way from the coefficients of $B[X]$:

$$\begin{pmatrix} s_{(m+1)1}=b_m & s_{(m+1)2}=b_{m-1} & \cdots & s_{(m+1)(m+1)}=b_0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \end{pmatrix}$$

For example, type:

$$\texttt{RESULTANT}(\texttt{X}^3 - \texttt{p} * \texttt{X} + \texttt{q}, 3\texttt{X}^2 - \texttt{p})$$

You obtain:

$$-4\texttt{p}^3 + 27\texttt{q}^2$$

4.21.31 SEVAL

SEVAL has an expression as argument.
SEVAL simplifies the expression, operating on all but the top-level operator of the expression.
For example, type:

$$\texttt{SEVAL}(\texttt{SIN}(3 \cdot \texttt{X} - \texttt{X}) + \texttt{SIN}(\texttt{X} + \texttt{X}))$$

You obtain:

$$\texttt{SIN}(2 \cdot \texttt{X}) + \texttt{SIN}(2 \cdot \texttt{X})$$

4.21.32 SIGMA

SIGMA has two arguments: the first argument is a function $f(x)$ of a variable x given as the second argument. SIGMA returns the discrete antiderivative of the input function, that is, the function G that satisfies the relation: $G(x+1) - G(x) = f(x)$.
For example, type:

$$\texttt{SIGMA}(\texttt{X} \cdot \texttt{X}!, \texttt{X})$$

You obtain:

$$\texttt{X}!$$

4.21.33 SIGMAVX

SIGMAVX has as argument a function f of the current variable VX.
SIGMAVX returns the discrete antiderivative of the input function, that is a function G that satisfies the relation: $G(x+1) - G(x) = f(x)$.
For example, type:

$$\texttt{SIGMAVX}(\texttt{X}^2)$$

You obtain:

$$\frac{2.\texttt{X}^3 - 3.\texttt{X}^2 + \texttt{X}}{6}$$

4.21.34 VER

VER returns the version number of your CAS.
Type:
VER
You obtain:

4.20000124

This result means that you have a version 4 CAS, dated 24 January 2000.
Instead, VERSION returns the version number of the calculator's ROM as a whole.
Type:
VERSION
You obtain:

{"Version HP49 − B ↩ Revision #1.17 − 4″ "Copyright HP 1999″}

4.21.35 TABVAR

TABVAR has as argument an expression whose derivative is rational.
TABVAR returns the variation table of the expression with respect to the current variable.
Type:

$$\texttt{TABVAR}(\texttt{LN}(\texttt{X}) + \texttt{X})$$

In step-by-step mode you obtain:
$\texttt{F} =: (\texttt{LN}(\texttt{X}) + \texttt{X})$
$\texttt{F}' =: (\frac{1}{\texttt{X}} + 1)$
$\rightarrow: \frac{\texttt{X}+1}{\texttt{X}}$

Variation table :

$$\begin{bmatrix} -\infty & ? & 0 & + & +\infty & X \\ ? & ? & -\infty & \uparrow & +\infty & F \end{bmatrix}$$

4.21.36 SIGNTAB

SIGNTAB has a rational expression as argument.
SIGNTAB returns the sign table of the expression with respect to the current variable.
Type:

$$\texttt{SIGNTAB}(\texttt{X}^2+\texttt{X})$$

You obtain:

$$\{-\infty\ +\ -1\ -\ 0\ +\ +\infty\}$$

4.21.37 TABVAL

TABVAL has an expression and a list of numbers as arguments.
TABVAL stores the input expression into the EQ variable and returns a list. This list holds the expression itself and a list of values; the latter list contains the value the expression assumes when the current CAS variable is replaced by the list of numbers given as input.
For example, type:

$$\texttt{TABVAL}(\texttt{X}^2+\texttt{X},\{1,2,3\})$$

You obtain:

$$\{\texttt{X}^2+\texttt{X},\{\{1,2,3\},\{2,6,12\}\}\}$$

4.21.38 PLOT

PLOT has an expression as argument.
PLOT stores the input expression into the EQ variable and opens the PLOT SETUP window.
For example, type:

$$\texttt{PLOT}(\texttt{X}^2+\texttt{X})$$

Now, by pressing ERASE DRAW, you obtain the plot of the expression stored in EQ, and

$$\texttt{X}^2+\texttt{X}$$

is stored into the history.

4.21.39 PLOTADD

PLOTADD has an expression as argument.
PLOTADD appends this expression to the list of equations currently stored in EQ and opens the PLOT SETUP window.
Type:

$$\texttt{PLOTADD}(\texttt{X}^2 - \texttt{X})$$

Now, by pressing ERASE DRAW, you draw the plots of all expressions stored in EQ one over another, and

$$\texttt{X}^2 - \texttt{X}$$

is stored into the history.

4.21.40 SCROLL

SCROLL has a graphics object as argument.
SCROLL displays the graphics object, without touching PICT.
For example, suppose you have just plotted the function F(X). The plot is automatically stored into the reserved variable PICT.
Therefore, PICT contains the current graphics variable. You can save its contents, typing:

$$\texttt{RCL(PICT) STO} \triangleright \texttt{ GRF}$$

Next, if you type:

$$\texttt{SCROLL(GRF)}$$

you display the plot of F(X) again.

4.21.41 GROBADD

GROBADD has two graphics objects as arguments.
GROBADD concatenates its arguments and returns the result.
For example, if you previously saved the plot of F(X) into GRF, and the plot of H(X) into GRH, you can type:

$$\texttt{GROBADD(GRF, GRH) STO} \triangleright \texttt{ GRFH}$$

This command stores into GRFH a graphics object containing the plots of both F(X) and H(X), one below the other.

Chapter 5

Bac 99 and HP49G

5.1 Introduction

First of all, enter `CASCFG` (Computer Algebra System ConFiG) to put the calculator in algebraic mode and to initialize it.
The commands you will use can be found in the menu displayed by the `SYMB` key, and in the following sub-menus:
`ALGEBRA (FACTOR LIN SUBST)`
`ARITHMETIC (IEGCD ISPRIME? PROPFRAC)`
`CALCULUS (DERIVX DERIV INTVX INT LIMIT)`
`GRAPH (SIGNTAB TABVAR)`
`TRIGONOMETR (TEXPAND)`
and in the `red-shift 1 (CMPLX)` menu:
`RE IM`
Remember that after entering each command, it is necessary to press `ENTER` to execute it; this notice will often be omitted in the following.

Here, you find the mathematics test of the "Bac"[1] 1999 test (S series) solved.
Effort has been made to leverage the HP49G capabilities as much as possible, but you will notice that it is still the student's duty to explain and justify his/her calculations, and to make a bit of reasoning.

[1] Translator's note: In France, the "Bac" (short form of "Baccalauréat") certificate is awarded to students that successfully complete the upper secondary school course, at about age 18.

5.2 Exercise 1

The goal of this exercise is to plot the curve Γ represented by $M : \frac{1}{2} \cdot z^2 - z$ when m, represented by z, is a circle C of center O and radius 1. Let t be a real in $[-\pi, \pi]$ and m the point of C corresponding to $z = e^{i \cdot t}$.

1. Calculation of the coordinates of M :
   With the help of EQW, we enter the expression $\frac{1}{2} \cdot z^2 - z$.
   We type:

   EQW alpha Z y^x 2 ▷ ÷ 2 ▷ − alpha Z ENTER

   The expression is now in the command line, and we store it into the variable M:

   STO▷ M

   Since $z = e^{i \cdot t}$ we type:

   SUBST(M, Z = EXP(i × t))

   the answer is:

   $$\frac{\texttt{EXP(i} \cdot \texttt{t)}^2 - 2 \cdot \texttt{EXP(i} \cdot \texttt{t)}}{2}$$

   Now, we linearize the expression, using the history to copy the previous expression down:

   LIN(HIST ENTER) ENTER

   the answer is:

   $$\frac{1}{2} \cdot \texttt{EXP(2} \cdot \texttt{i} \cdot \texttt{t)} + -1 \cdot \texttt{EXP(i} \cdot \texttt{t)}$$

   Notice that by copying the expression down, the calculator simplifies it, too:

   △ ENTER ENTER

   gives:

   $$\frac{\texttt{EXP(2} \cdot \texttt{i} \cdot \texttt{t)} - 2 \cdot \texttt{EXP(i} \cdot \texttt{t)}}{2}$$

   - Now we want to look at the real part of this expression:

     RE(HIST ENTER) ENTER

     the answer is:

     $$\frac{\texttt{COS(t} \cdot 2) - 2 \cdot \texttt{COS(t)}}{2}$$

At this point, we can define the function $x(t)$, by typing:

$$\texttt{DEFINE (X(t) = HIST ENTER) ENTER}$$

- Then, we calculate the imaginary part (we must climb up in the history to retrieve the original expression $\frac{\texttt{EXP(2·i·t)−2·EXP(i·t)}}{\texttt{2}}$), typing:

$$\texttt{IM(HIST } \triangle\ \triangle\ \triangle\ \triangle \texttt{ ENTER) ENTER}$$

the answer is:

$$\frac{\texttt{SIN(t · 2) − 2 · SIN(t)}}{\texttt{2}}$$

To define the function $y(t)$, we must now type:

$$\texttt{DEFINE(Y(t) = HIST ENTER)ENTER}$$

2. To determine an axis of symmetry of Γ, we want to calculate $x(-t)$ and $y(-t)$; for this, we type:

$$\texttt{X(−t) ENTER}$$

the answer is:

$$\frac{\texttt{COS(t · 2) − 2 · COS(t)}}{\texttt{2}}$$

Therefore: $x(-t) = x(t)$
after this, we type:

$$\texttt{Y(−t) ENTER}$$

the answer is:

$$\frac{\texttt{−SIN(t · 2) + 2 · SIN(t)}}{\texttt{2}}$$

Therefore: $y(-t) = -y(t)$
If $M_1(x(t), y(t))$ is on Γ, $M_2(x(-t), y(-t)$ is on Γ, too.
We have just showed that M_1 and M_2 are symmetric with respect to Ox; from this we conclude that the Ox axis is an axis of symmetry of Γ.

3. Calculation of $x'(t)$:
We type:

$$\texttt{DERIV(X(t), t)}$$

the answer is:

$$\frac{\texttt{2 · (−2 · SIN(t · 2)) − 2 · (−SIN(t))}}{\texttt{4}}$$

that is, after simplification (△ ENTER ENTER):

$$-(\mathtt{SIN}(\mathtt{t}\cdot 2)-\mathtt{SIN}(\mathtt{t}))$$

We now expand the expression (transformation of $\mathtt{SIN}(2\cdot\mathtt{t})$); we type:

TEXPAND(HIST ENTER) ENTER

the answer is:

$$-(\mathtt{SIN}(\mathtt{t})\cdot 2\cdot\mathtt{COS}(\mathtt{t})-\mathtt{SIN}(\mathtt{t}))$$

Now we factorize:

FACTOR(HIST ENTER) ENTER

the answer is:

$$-\mathtt{SIN}(\mathtt{t})\cdot(2\cdot\mathtt{COS}(\mathtt{t})-1)$$

At last, we can define $x'(t)$ by typing:

DEFINE(X1(t) = HIST ENTER) ENTER

4. Calculation of $y'(t)$:
We type:

$$\mathtt{DERIV}(\mathtt{Y}(\mathtt{t}),\mathtt{t})$$

the answer is:

$$\frac{2\cdot(2\cdot\mathtt{COS}(\mathtt{t}\cdot 2))-2\cdot\mathtt{COS}(\mathtt{t})}{4}$$

that is, after simplification (△ ENTER ENTER):

$$\mathtt{COS}(\mathtt{t}\cdot 2)-\mathtt{COS}(\mathtt{t})$$

We now expand the expression (transformation of $\mathtt{COS}(2\cdot\mathtt{t})$); we type:

TEXPAND(HIST ENTER) ENTER

the answer is:

$$2\cdot\mathtt{COS}(\mathtt{t})^2-1-\mathtt{COS}(\mathtt{t}))$$

Now we factorize:

FACTOR(HIST ENTER) ENTER

the answer is:

$$(\mathtt{COS}(\mathtt{t})-1)\cdot(2\cdot\mathtt{COS}(\mathtt{t})+1)$$

At last, we can define $y'(t)$ by typing:

DEFINE(Y1(t) = HIST ENTER) ENTER

5. Variations of $x(t)$ and $y(t)$
   To do this we draw on the same plot both $x(t)$ and $y(t)$; we type:
   `blue-shift F4 (2D/3D)`: the `PLOT SETUP` window opens.
   We choose `function` as plot type, with the help of the `choos` menu key.
   Then, we enter

   $$\{\texttt{X(t)}, \texttt{Y(t)}\}$$

   into the equation (`EQ`) field, and `t` as independent variable, followed by `ENTER`.
   Afterwards, we type `blue-shift F2 (WIN)`, to set the plot window parameters.

6. Trace of Γ:

   - Values of $x(t)$ and $y(t)$
     We calculate the values of $x(t)$ and $y(t)$ for $t = 0, \frac{\pi}{3}, \frac{2\cdot\pi}{3}, \pi,$ by typing:

     $$\texttt{X(0) ENTER}$$

     answer: $\frac{-1}{2}$

     $$\texttt{X}(\pi \div 3)\ \texttt{ENTER}$$

     answer: $\frac{-3}{4}$

     $$\texttt{X}(2 \times \pi \div 3)\ \texttt{ENTER}$$

     answer: $\frac{1}{4}$

     $$\texttt{X}(\pi)\ \texttt{ENTER}$$

     answer: $\frac{3}{2}$

     $$\texttt{Y(0) ENTER}$$

     answer: 0

     $$\texttt{Y}(\pi \div 3)\ \texttt{ENTER}$$

     answer: $\frac{-\sqrt{3}}{4}$

     $$\texttt{Y}(2 \times \pi \div 3)\ \texttt{ENTER}$$

     answer: $\frac{-3\cdot\sqrt{3}}{4}$

     $$\texttt{Y}(\pi)\ \texttt{ENTER}$$

     answer: 0

- Slope of tangents ($m = \frac{y'(t)}{x'(t)}$)
  We calculate the values of $\frac{y'(t)}{x'(t)}$ for $t = 0, \frac{\pi}{3}, \frac{2\cdot\pi}{3}, \pi$, by typing:

  $$\texttt{LIMIT(Y1(t)/X1(t), t = 0) ENTER}$$

  answer: 0

  $$\texttt{LIMIT(Y1(t)/X1(t), t = } \pi \div \texttt{3) ENTER}$$

  answer: ∞

  $$\texttt{LIMIT(Y1(t)/X1(t), t = 2} \times \pi \div \texttt{3) ENTER}$$

  answer: 0

  $$\texttt{LIMIT(Y1(t)/X1(t), t = } \pi\texttt{) ENTER}$$

  answer: ∞
  These are the variations of $x(t)$ and $y(t)$

t	0		$\frac{\pi}{3}$		$\frac{2\pi}{3}$		π
$x'(t)$	0	$-$	0	$+$	?	$+$	0
$x(t)$	$\frac{-1}{2}$	$\downarrow$	$\frac{-3}{4}$	$\uparrow$	$\frac{1}{4}$	$\uparrow$	$\frac{3}{2}$
$y(t)$	0	$\downarrow$	$\frac{\sqrt{3}}{4}$	$\downarrow$	$\frac{-3\sqrt{3}}{4}$	$\uparrow$	0
$y'(t)$	0	$-$	?	$-$	0	$+$	?
m	0		∞		0		∞

- Plotting Γ:
  We now plot the parametric curve.
  We type:
  `blue-shift F4 (2D/3D)` and the `PLOT SETUP` opens up.
  We choose the `parametric` plot type, with the help of the `choos` menu key. Then, we enter

  $$\texttt{X(t)} + \texttt{i} \times \texttt{Y(t)}$$

  into the equation (`EQ`) field and `t` as independent variable, followed by `ENTER`.
  We then press `blue-shift F2 (WIN)` to set the plot window parameters.

5.3 Exercise 2 (specialized)

Let be, for a natural n:

$$a_n = 4 \times 10^n - 1,\ b_n = 2 \times 10^n - 1 \text{ and } c_n = 2 \times 10^n + 1$$

We type:

$$\texttt{DEFINE(A(N)} = \texttt{4} \cdot \texttt{10}^{\texttt{N}} - \texttt{1)}$$

$$\texttt{DEFINE(B(N)} = \texttt{2} \cdot \texttt{10}^{\texttt{N}} - \texttt{1)}$$

$$\texttt{DEFINE(C(N)} = \texttt{2} \cdot \texttt{10}^{\texttt{N}} + \texttt{1)}$$

1. • a) Calculate $a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, c_3$:
   It suffices to type:

   A(1)

   answer 39

   B(1)

   answer 19

   C(1)

   answer 21

   A(2)

   answer 399

   B(2)

   answer 199

   C(2)

   answer 201

   A(3)

   answer 3999

   B(3)

   answer 1999

   C(3)

   answer 2001

- b) number of digits and divisibility
  Here, the calculator is useful only to do some evaluation trials for different values of n...
  We know that integers n satisfying:

$$10^n \leq n < 10^{n+1}$$

  have $(n+1)$ digits when they are written using the decimal representation.
  We have:

$$3 \cdot 10^n \ < \ a_n \ < \ 4 \cdot 10^n$$

$$10^n \ < \ b_n \ < \ 2 \cdot 10^n$$

$$2 \cdot 10^n \ < \ c_n \ < \ 3 \cdot 10^n$$

  so the decimal representation of $a_n,\ b_n,\ c_n$ has $(n+1)$ digits.

  Moreover, $d_n = 10^n - 1$ is divisible by 9, since all digits of its decimal representation are 9.
  We have

$$a_n = 3 \cdot 10^n \ + \ d_n$$

  and

$$c_n = 3 \cdot 10^n \ - \ d_n$$

  so a_n and c_n are divisible by 3.

- c) b_3 is prime
  We type:

$$\texttt{ISPRIME?(B(3))}$$

  the answer is:

$$\texttt{1.}$$

  that means `true`
  To show that $b_3 = 1999$ actually is prime, we only have to test if 1999 is divisible by all prime numbers less than or equal to $\sqrt{1999}$.
  Since $1999 < 2025 = 45^2$, we check the divisibility of 1999 with $n = 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41$.
  Since 1999 is not divisible by any of these numbers, we conclude that 1999 actually is prime.

- d) $a_n = b_n \times c_n$
  We type:
  $$\texttt{B(N)} \cdot \texttt{C(N)}$$
  The answer is:
  $$\texttt{4} \cdot \left(\texttt{10}^{\texttt{N}}\right)^{\texttt{2}} - \texttt{1}$$
  that is the same value as a_n
  Factorization of a_6 into prime factors
  We type:
  $$\texttt{FACTOR(A(6))}$$
  The answer is:
  $$\texttt{3} \cdot \texttt{23} \cdot \texttt{29} \cdot \texttt{1999}$$

- e) b_n et c_n are mutually prime.
  Here, the calculator is useful only to do some evaluation trials for different values of n...
  To show that c_n and b_n are mutually prime, we observe that:
  $$c_n = b_n + 2$$
  So, the common divisors of c_n and b_n are the common divisors of b_n and 2, and are the common divisors of c_n and 2, too. b_n and 2 are mutually prime, because b_n is a prime number not equal to 2. Therefore,
  $$GCD(c_n,\ b_n) = GCD(c_n,\ 2) = GCD(b_n,\ 2) = 1$$

2. Consider the equation:
   $$b_3 \cdot x + c_3 \cdot y = 1$$

- a) There is at least a solution, because we can apply the Bézout identity. In fact, the Bézout theorem states that:
  If a and b are mutually prime, two values x and y do exist so that:
  $$a \cdot x + b \cdot y = 1$$
  Therefore, the equation:
  $$b_3 \cdot x + c_3 \cdot y = 1$$
  has at least a solution.

- b)We type:

  $$\texttt{IEGCD(B(3),C(3))}$$

  the answer is:

  $$\{\texttt{1},\texttt{1000},\texttt{-999}\}$$

  this means that:

  $$1 = b_3 \times 1000 + c_3 \times (-999)$$

  so, we have just found a (particular) solution:
  $x = 1000,\ y = -999$.
  By hand, we write:
  $c_3 = b_3 + 2$ and $b_3 = 999 \times 2 + 1$
  so, $b_3 = 999 \times (c_3 - b_3) + 1$ and:

  $$b_3 \times 1000 + c_3 \times (-999) = 1$$

- c) Here, the calculator cannot find the general solution.
  We have:

  $$b_3 \cdot x + c_3 \cdot y = 1$$

  and

  $$b_3 \times 1000 + c_3 \times (-999) = 1$$

  subtracting the second equation from the first, we have:

  $$b_3 \cdot (x - 1000) + c_3 \cdot (y + 999) = 0$$

  and then:

  $$b_3 \cdot (x - 1000) = -c_3 \cdot (y + 999)$$

  Applying the Gauss theorem: c_3 is prime with b_3, so c_3 divides $(x - 1000)$.
  Therefore, it exists $k \in Z$ so that:

  $$(x - 1000) = k \times c_3$$

  and

  $$-(y + 999) = k \times b_3$$

  Reciprocally, let be:

  $$x = 1000 + k \times c_3$$

and

$$y \ = \ -999 \ - \ k \times \ b_3 \ \text{ for } \ k \ \in Z$$

We have:

$$b_3 \cdot x + c_3 \cdot y = b_3 \times 1000 + c_3 \times (-999) \ = \ 1$$

The general solution therefore is: for any $k \ \in Z$:

$$x \ = \ 1000 \ + \ k \times \ c_3$$

$$y \ = \ -999 \ - \ k \times \ b_3$$

5.4 Exercise 2 (not specialized)

Let us consider the succession

$$u_n = \int_0^2 \frac{2x+3}{x+2} \, e^{\frac{x}{n}} \, dx$$

1. • a) Variations of $g(x) = \frac{2x+3}{x+2}$ for $x \in [0, 2]$
We type:

$$\texttt{DEFINE(G(X)} = \frac{\texttt{2X+3}}{\texttt{X+2}}\texttt{)}$$

then, we type:

$$\texttt{TABVAR(G(X))}$$

We obtain:

$-\infty$	$+$	-2	$+$	$+\infty$	X
2	$\uparrow$	∞	$\uparrow$	2	F

The first line gives the sign of `f'(x)` depending on x, and the second one gives the variations of `f(x)`.

From this result, we conclude that $g(x)$ increases in the interval $[0, 2]$.

Notice that if the calculator is in step-by-step mode (to enable this mode, you must press on `MODE`, then on the `cas` menu key, and set `Step/Step` using the `chk` menu key followed by `ok ok`), we obtain (the input function is denoted by `F`):

$$\texttt{F} := \frac{\texttt{2} \cdot \texttt{X} + \texttt{3}}{\texttt{X} + \texttt{2}}$$

After pressing the ok menu key, we obtain:

$$\mathtt{F}' := \frac{2\cdot(\mathtt{X}+2)-(2\cdot\mathtt{X}+3)}{\mathtt{SQ}(\mathtt{X}+2)}$$

and, using the $\bigtriangledown$ arrow key to scroll the screen:

$$\rightarrow \frac{1}{(\mathtt{X}+2)^2}$$

At last, pressing the ok menu key again displays the variation table.

If the step-by-step mode is disabled, it is possible to calculate the derivative with:

$$\mathtt{DERVX}(\mathtt{G}(\mathtt{X}))$$

We want to calculate $g(0)$ and $g(2)$; to do this we type:

$$\mathtt{G}(0)$$

answer $\frac{3}{2}$

$$\mathtt{G}(2)$$

answer $\frac{7}{4}$

$$\frac{3}{2} \leq g(x) \leq \frac{7}{4} \text{ for } x \in [0,2]$$

- b) Here the calculator is not useful at all... it is enough to notice that:

$$e^{\frac{x}{n}} \geq 0 \text{ for } x \in [0,2]$$

to show that, for $x \in [0,2]$, we have:

$$\frac{3}{2}e^{\frac{x}{n}} \leq g(x)e^{\frac{x}{n}} \leq \frac{7}{4}e^{\frac{x}{n}}$$

- c) We integrate the inequality written above, typing:

$$\int_0^2 \mathtt{e}^{\frac{\mathtt{X}}{\mathtt{N}}}\,\mathtt{dX}$$

We obtain:

$$\mathtt{N}\cdot\mathtt{e}^{\frac{2}{\mathtt{N}}}-\mathtt{N}$$

From this, we conclude that:

$$\frac{3}{2}(ne^{\frac{2}{n}} - n) \leq u_n \leq \frac{7}{4}(ne^{\frac{2}{n}} - n)$$

To justify the previous calculation, we must also say that a primitive of $e^{\frac{x}{n}}$ is $n \cdot e^{\frac{x}{n}}$.
If we don't know this, we can always type:

$$\texttt{INTVX(EXP(X} \div \texttt{N))}$$

and the answer is: $\texttt{N} \cdot \texttt{e}^{\frac{\texttt{X}}{\texttt{N}}}$

- d) We look for the limit of $(ne^{\frac{2}{n}} - n)$ when $n \to +\infty$:

$$\texttt{LIMIT(N} \cdot \texttt{EXP(2} \div \texttt{N)} - \texttt{N , N} = +\infty\texttt{)}$$

We obtain:

$$2$$

To explain this result, we must say that:

$$\lim_{x \to 0} \frac{e^x - 1}{x} = 1$$

and, as a consequence:

$$\lim_{n \to +\infty} \frac{e^{\frac{2}{n}} - 1}{\frac{2}{n}} = 1$$

or, also:

$$\lim_{n \to +\infty} (e^{\frac{2}{n}} - 1) \cdot n = 2$$

If L does exist, letting n go towards $+\infty$ in the inequalities of point 1b), we obtain:

$$\frac{3}{2} \cdot 2 \leq L \leq \frac{7}{4} \cdot 2$$

2. - a) $g(x) = 2 - \frac{1}{x+2}$ and calculation of $I = \int_0^2 g(x)dx$
We type:

$$\texttt{PROPFRAC(G(X))}$$

We obtain:

$$2 - \frac{1}{\texttt{X} + 2}$$

To calculate I, we type in the equation editor (EQW key):

$$\int_0^2 \texttt{G(X)dX}$$

We obtain:

$$-(\texttt{LN(2)} - 4)$$

Doing it by hand, we have $2x + 3 = 2(x + 2) - 1$, so

$$g(x) = 2 - \frac{1}{x+2}$$

We now integrate term by term between 0 et 2, obtaining:

$$\int_0^2 g(x)dx = [2x - \ln(x+2)]_{x=0}^{x=2}$$

that is, since $\ln 4 = 2\ln 2$:

$$\int_0^2 g(x)dx = 4 - \ln 2$$

- b) Here, the calculator is not useful... it is enough to notice that $e^{\frac{x}{n}}$ increases for $x \in [0, 2]$, to obtain the inequality:

$$1 \ \leq \ e^{\frac{x}{n}} \ \leq e^{\frac{2}{n}}$$

then, since $g(x)$ is positive on $[0, 2]$, we have:

$$g(x) \ \leq \ g(x)e^{\frac{x}{n}} \ \leq g(x)e^{\frac{2}{n}}$$

integrating we have:

$$I \ \leq u_n \ \leq \ e^{\frac{2}{n}} I$$

- c) Convergence of u_n
We look for the limit of $e^{\frac{2}{n}}$ when $n \to +\infty$:

$$\texttt{LIMIT(EXP(2} \div \texttt{N)} \ , \ \texttt{N} = +\infty)$$

We obtain:

$$\texttt{1}$$

Actually, $\frac{2}{n}$ goes towards 0 when n goes towards $+\infty$, so $e^{\frac{2}{n}}$ goes towards $e^0 = 1$ when n goes towards $+\infty$.
When n goes towards $+\infty$, u_n stays between I and a quantity that goes towards I (see the inequalities in point 2b)).
Therefore, u_n converges and its limit is I.
We have shown that:

$$L = I = 4 - \ln 2$$

5.5 Problem

Part A
Let a function f be defined on $]0,\ +\infty[$ as

$$f(x) = \left(1 - \frac{1}{x}\right)(\ln x - 2)$$

We type (using the Equation Writer):

$$\texttt{DEFINE(F(X)} = (1 - 1 \div \texttt{X}) \times (\texttt{LN(X)} - 2))$$

This is, in detail, the sequence of keys we must press ($\triangleleft$ represents the cursor):

$$\texttt{DEFINE(F(X)} = \triangleleft)$$

To enter the expression that follows, using the equation editor, we press the `EQW` key.
We now are in the equation editor; we type:

$$1\ -\ 1\ \div\ \texttt{X} \triangleright\ \triangleright\ \triangleright\ \times\ \texttt{LN(X)}\ \triangleright\ -\ 2\ \texttt{ENTER}$$

In the command line, we have:

$$\texttt{DEFINE(F(X)} = (1 - \frac{1}{\texttt{X}}) \cdot (\texttt{LN(X)} - 2))$$

At this point, we press `ENTER` to execute it.
`F` is added to the variable's menu and `NOVAL` is displayed in the screen.
To check our work, we type `F(X)`; we should obtain:

$$\frac{(\texttt{X} - 1) \cdot \texttt{LN(X)} - (2 \cdot \texttt{X} - 2)}{\texttt{X}}$$

1. Limit of f in $+\infty$ and 0.
   We type:

   $$\texttt{LIMIT(F(X)}, +\infty)$$

   answer $+\infty$
   then,

   $$\texttt{LIMIT(F(X)}, 0)$$

   answer ∞

2. Calculation of $f'(x)$.
   We type:
   $$\texttt{DERVX(F(X))}$$
   We obtain:
   $$\frac{\texttt{LN(X)+X-3}}{\texttt{X}^2}$$

3. $u(x) = \ln x + x - 3$
   Now, we type:
   $$\texttt{DEFINE(U(X)=LN(X)+X-3)}$$

   - a) Variations of u.
     We type:
     $$\texttt{TABVAR(U(X))}$$
     The calculator asks to enable complex mode: answer `YES`.
     We obtain:

$-\infty$	$+$	-1	$-$	0	$+$	$+\infty$	X
$-\infty$	$\uparrow$	$i\pi - 4$	$\downarrow$	$-\infty$	$\uparrow$	$+\infty$	F

     Warning!!!!
     Only the portion of table in which $x > 0$ must be taken into account (when $x < 0$ the calculator assumes $\ln(x)$ exists, and has a complex value).

   - b) $u(x) = 0$ has an unique solution α in $[2,\ 3]$.
     From a) u increases in $]0,\ +\infty[$.
     We type:
     $$\texttt{U(2) red} - \texttt{shift ENTER}$$
     answer $-0.306...$
     then,
     $$\texttt{U(3) red} - \texttt{shift ENTER}$$
     answer $1.098...$
     Thereafter, we calculate:
     $$\texttt{U(2.20) red} - \texttt{shift ENTER}$$
     answer $-0.306...$
     and
     $$\texttt{U(2.21) red} - \texttt{shift ENTER}$$

answer $-0.306...$
From the theorem of intermediate values (u is both increasing and continuous in $[2,3]$, therefore u zeroes itself exactly once between 2 et 3 (since $u(2) < 0$ and $u(3) > 0$)).
So, if we call α the unique zero of u in $[2,\ 3]$, we have:

$$2.20 < \alpha < 2.21$$

since $u(2.20) < 0$ and $u(2.21) > 0$.

- c) Sign of $u(x)$ in $]0,\ +\infty[$
The sign of $u(x)$ can be deducted from the variation table of u; we have:

$$\begin{cases} u(x) < 0 & \text{for } x < \alpha \\ u(x) = 0 & \text{for } x = \alpha \\ u(x) > 0 & \text{for } x > \alpha \end{cases}$$

4. - a) Variations of f.
We must do the variation table by hand, because the derivative of f is not a rational function... and the calculator is not yet able to handle this case!
Since the sign of $f'(x)$ is the same as of $u(x)$, we have:

$f'(x)$	0	$-$	α	$+$	$+\infty$
$f(x)$	$+\infty$	$\downarrow$	$\frac{\alpha-1}{\alpha}(ln(\alpha)-2)$	$\uparrow$	$+\infty$

- b)$f(\alpha) = -\frac{(\alpha-1)^2}{\alpha}$
We have:
$u(\alpha) = 0$, so $\ln(\alpha) = 3 - \alpha$
We type in the equation writer:

$$(1 - \frac{1}{A})(LN(A) - 2)$$

then we highlight `LN(A)`,
open the `ALG (red-shift 4)` menu,
press the (`SUBST`) key
to complete the command `SUBST(LN(A),LN(A)=3-A)`,
and press `ENTER ENTER`.
We obtain:

$$-\frac{A^2 - 2 \cdot A + 1}{A}$$

Now,

$$\texttt{FACTOR}(-\frac{\texttt{A}^2-2\cdot\texttt{A}+1}{\texttt{A}})$$

returns

$$-\frac{(\texttt{A}-1)^2}{\texttt{A}}$$

We type:

$$\texttt{DERIV}(-\frac{(\texttt{A}-1)^2}{\texttt{A}},\texttt{A})$$

We obtain:

$$-\frac{(\texttt{A}^2-1)}{\texttt{A}^2}$$

So, the function $v(x)=-\frac{(x-1)^2}{x}$ decreases for $x>=1$.
We obtain a bounded approximation of $f(\alpha)$ calculating:
$v(2.21)$ and $v(2.20)$.
We type:

$$-\frac{(1.21)^2}{2.21}\ \texttt{red}-\texttt{shift ENTER}$$

answer -0.662488

$$-\frac{(1.2)^2}{2.2}\ \texttt{red}-\texttt{shift ENTER}$$

answer $-0.65454...$
so, we have:

$$-0.663<f(\alpha)<-0.654$$

This is an approximation with tolerance $9\cdot10^{-3}$(since $0.663-0.654=9\cdot10^{-3}$)
or, again:

$$-0.67<f(\alpha)<-0.65$$

that is an approximation with tolerance $2\cdot10^{-2}$(since $0.67-0.65=2\cdot10^{-2}$)

5. • a) Sign of f
We notice that $f(1)=0$ and that $f(e^2)=0$
We type:

$$\texttt{F(1)}$$

answer : 0

$$\texttt{F(EXP(2))}$$

answer : 0
These are the variations of f and the sign of $f(x)$:

$f'(x)$	0	$-$	1	$-$	α	$+$	e^2	$+$	$+\infty$
$f(x)$	$+\infty$	$\downarrow$	0	$\downarrow$	$\frac{\alpha-1}{\alpha}(ln(\alpha)-2)$	$\uparrow$	0	$\uparrow$	$+\infty$
$f(x)$	$+\infty$	$+$	0	$-$	$\simeq -0.66$	$-$	0	$+$	$+\infty$

- b) Plot C of f
  We open the `PLOT SETUP (blue_shift F4)` menu, we choose `function` and `F(X)` for `EQ`, then we set the window parameters in `WIN (blue_shift F2)`

Part B
Let H be the antiderivative of f in $]0,\ +\infty[$ and Γ its plot.
Be careful with the notations; they are not the same as those of the textbook!

1. - a) Variations of H
     Since $H'(x) = f(x)$ we have the following variation table:

$f(x)$	0	$+$	1	$-$	e^2	$+$	$+\infty$
$H(x)$	?	$\uparrow$	0	$\downarrow$	?	$\uparrow$	?

   - b) Tangent lines for $x = 1$ and $x = e^2$
     It is $f(1) = 0$ and $f(e^2) = 0$. The tangents of Γ for the abscissas 1 and e^2 have a zero slope, and are therefore horizontal.

2. Calculation of $H(x)$

   - a) Calculation of $\int_1^x \ln t dt$
     In the equation editor, we type:

$$\int_1^{\mathtt{X}} \mathtt{LN(T)dT}$$

     We obtain:

$$\mathtt{X}\cdot\mathtt{LN(X)} - (\mathtt{X}-\mathtt{1})$$

     We can also ask for the antiderivative of $\ln x$, we type:

$$\mathtt{INTVX(LN(X))}$$

We obtain:

$$\texttt{X}\cdot\texttt{LN(X)}-\texttt{X}$$

By hand, we set $u=\ln(t)$ and $dv=dt$, so $du=\frac{dt}{t}$ and $v=t$ we have:

$$\int_1^x \ln t\ dt=[t\cdot\ln t]_{t=1}^{t=x}-\int_1^x t\cdot\frac{dt}{t}=x\cdot\ln x-\int_1^x dt=x\cdot\ln x-(x-1)$$

- b) Expanding the expression of $f(x)$ we obtain:

$$f(x)=\ln x-\frac{\ln x}{x}+\frac{2}{x}-2$$

- c) Expression of $H(x)$
We type (using the equation writer):

$$\texttt{DEFINE(H(X)}=\int_1^{\texttt{X}}\texttt{F(T)dT)}$$

then,

$$\texttt{H(X) ENTER}$$

We obtain:

$$-\frac{\texttt{LN(X)}^2-(\texttt{2}\cdot\texttt{X}+\texttt{4})\cdot\texttt{LN(X)}+\texttt{6}\cdot\texttt{X}-\texttt{6}}{\texttt{2}}$$

We do the calculation by hand, integrating term by term the expression of $f(x)$ found in 2b); we have:

$$\int_1^x \ln t\ dt=x\ln x-x+1$$

$$-\int_1^x \frac{\ln t}{t}\ dt=-\frac{(\ln x)^2}{2}$$

$$\int_1^x \frac{2}{t}\ dt=2\ln x$$

that's why:

$$H(x)=x\ln x-x+1-\frac{(\ln x)^2}{2}+2\ln x-2x+2$$

$$H(x)=-\frac{(\ln x)^2}{2}+(x+2)\ln x-3x+3$$

3.

- a) We type

$$\texttt{LIMIT(LN(X)/X, X = 0)}$$

answer 0
From the course, we know that "x dominates $\ln x$"; this explains the result!
We type:

$$\texttt{LIMIT(H(X), X = 0)}$$

answer $-\infty$
We have:

$$H(x) = x \ln x + \ln x \frac{(4 - \ln x)}{2} - 3x + 3$$

When x goes towards 0, the first term of $H(x)$ goes towards 0, its second term goes towards $-\infty$ (because $\ln x$ goes towards $-\infty$ when x goes towards 0), and the following terms go towards 3.
Threrefore,

$$\lim_{x \to 0} H(x) = -\infty$$

- b) We type:

$$\texttt{LIMIT(H(X), X = +}\infty\texttt{)}$$

answer $+\infty$
We put in evidence $x \ln x$ at the beginning of $H(x)$, obtaining:

$$H(x) = x \ln x (1 - \frac{\ln x}{2x} + \frac{2}{x} - \frac{3}{\ln x}) + 3$$

$$\lim_{x \to +\infty} x \ln x = +\infty$$

and the term within paretheses goes towards 1 when x goes towards $+\infty$, so

$$\lim_{x \to +\infty} H(x) = +\infty$$

- c) Variations of $H(x)$
We calculate $H(e^2)$
We type:

$$\texttt{H(EXP(2))}$$

The result is the expression obtained replacing, in `H(X)`, `X` with `EXP(2)`.

We copy this expression to simplify it: (△ ENTER ENTER), obtaining:

$$-(\texttt{EXP}(2)-5)$$

then,

$$\triangle \texttt{ ENTER red} - \texttt{shift ENTER}$$

answer -2.38905...
We have also an approximate value of $H(e^2)$.
We recall the table already worked out in 1a)

$f(x)$	0	$+$	1	$-$	e^2	$+$	$+\infty$
$H(x)$	$-\infty$	$\uparrow$	0	$\downarrow$	$\simeq -2.39$	$\uparrow$	$+\infty$

- d) Plot C of f and plot Γ of H
  We open the PLOT SETUP (blue_shift F4) window, choose function and put {F(X),H(X)} in the EQ field, then we set the window parameters using WIN (blue_shift F2) .

4. Area calculation
   We have calculated:

$$\int_1^{e^2} f(x)dx$$

obtaining:

$$-(\texttt{EXP}(2)-5)$$

this integral is negative because the function is below the x axis between 1 and e^2. Since the measurement unit is 2cm, the area expressed in cm^2 is therefore equal to $-4H(e^2)\text{cm}^2$, that is :

$$4\cdot(\texttt{EXP}(2)-5)\ \ \text{cm}^2$$

that is,

$$A=(4e^2-20)\ \ \text{cm}^2$$

or

$$9.55\ \ \text{cm}^2 < A < 9.56\ \ \text{cm}^2$$

5.6 Conclusion

We have showed that a good leverage of the HP49G calculator's capabilities enabled us to solve a large portion of the problems...
However, we must also notice that for arithmetic problems, more reasoning is needed: then, the calculator is useful to do checks and verifications...

Chapter 6

Programming

6.1 Programming in algebraic mode

6.1.1 Entering a program

You write a program in the command line, between delimiters ≪ and ≫

6.1.2 Saving a program

It is enough to append

```
STO▷ NOMDUPROGRAMME
```

after the last ≫ delimiter.

6.1.3 Editing a program

When entering a program, if the program syntax is bad, the calculator automatically places the command line cursor where the compiler has detected the error, so you can immediately fix it!!!

Instead, if the error is detected during program execution, you must enter: `VISIT('NOMDUPROGRAMME')` to edit the program stored in the `NOMDUPROGRAMME` variable.
You can then edit it; `ENTER` saves the edited program.

6.1.4 Executing a program

If the program has no parameters, you type its name in the command line; if it has one or more parameters, the program name must be followed by its arguments, enclosed in brackets and delimited by commas.

Example: `PGCD(45,75)`

6.1.5 Modifying a program and saving it with another name

You type:
`RCL('NOMDUPROGRAMME')` and press the `edit` menu key.

Then, you modify the program as you wish, and append

`STO ▷ NOUVEAUNOM`

after the last ≫ delimiter.

6.2 Program comments

It is a good habit to put comments in programs.

In algorithmic language a comment starts with `//` and ends at the end of the line.

On the `HP49G`, a comment starts with `@` and ends at the end of the line or when another `@` is encountered.

You obtain the `@` character by pressing `red-shift ENTER`.

Warning!!! The compiler deletes all comments so, to preserve your comments, you should store the program as a text string and then compile it with `STR` →; this makes things a little more difficult, though...

6.3 Variables

6.3.1 Variable names

Variables are places where you can store values, numbers, expressions and any other object.

6.3.2 Local variables

The HP49G has local variables. Local variables are both declared and initialized (the initialization is mandatory!) with → (red − shift 0)

In RPN mode, you can define and initialize more than one variable at once, for example:

≪ 1 2 → A B ≪ subprogrambody ≫≫

In Algebraic mode, each definition must be followed by a subprogram (with delimiters ≪≫).

The arrow must be surrounded by spaces (these spaces are automatically inserted when the calculator is not in α mode).
Example :

≪ 3.14 → PI ≪ 2 ∗ PI ∗ R ≫≫ STO▷ PER

In this example, we have written a program named PER.
PI is a local variable defined and initialized by the code fragment 3.14 → PI. The scope of this variable is limited to the subprogram that follows its definition (here, ≪ 2 ∗ PI ∗ R ≫).
On the other hand, R is a global variable (it must exist before the program PER is executed). If, during the execution of a program, you want to store a value into a variable (either local or global), you must use the STO▷ operator.

6.3.3 Parameters

When you define a function, it is possible to use parameters.
For example, if you want R to be a parameter of function PER, you should write:

≪ → R ≪ 3.14 → PI ≪ 2 ∗ PI ∗ R ≫≫≫ STO▷ PER

Parameter R behaves the same as local variables; the only difference is that it is initialized when the function is invoked.
The invocation is done when, for example, you evaluate: PER(5).
To define a function with two argument, the correct syntax, in both RPN and Algebraic modes is:

≪ → A B ≪ ...

6.4 Data input

6.4.1 Translation in algorithmic language

To denote that the user enters a value into variable `A` during the execution of a program, in algorithmic language you write: `input A`
To input values into `A` and `B` you write: `input A,B`

6.4.2 Translation for the HP49G in RPN mode

```
'A' PROMPTSTO
```
or
```
"A" "" INPUT STR-> EVAL 'A' STO
```

6.4.3 Translation for the HP49G in Algebraic mode

To input a value into a local variable `A`, you type:
....0 → A ≪ PROMPTSTO('A')... ≫ or
....0 → A ≪ PROMPTSTO('A');0 → B ≪ PROMPTSTO('B')..... ≫ ≫

If you enter only:
`...PROMPTSTO('A')`, `A` is then a global variable, or
`...PROMPTSTO('A'); PROMPTSTO('B')`, `A` and `B` are then global variables.
You can use `INPUT`, too. For example, if a local variable `A` must contain a character string, you can enter:

INPUT("A ="," ") → A

or, if `A` must contain a number:
.....INPUT("A ="," ") → A
≪ OBJ → (A) → A
≪ ..... ≫
≫

6.5 Data output

6.5.1 Translation in algorithmic language

In algorithmic language, you write:
`print "A=",A`

6.5.2 Translation for the HP49G in RPN mode

Usually it is enough to push results into the stack, so that you can reuse them; if this is the case, you simply write:
`A B`
You can also tag the result, as follows:
`A "A=" ->TAG`
`HALT` halts the program.

6.5.3 Translation for the HP49G in Algebraic mode

Only the last result is written into the history.
To display intermediate results, you can use either:

$$\texttt{DISP}(\text{"}\texttt{A} = \text{"} + \texttt{A}, 3)$$

here, `3` is the line number on which the result is displayed, or:

$$\texttt{MSGBOX}(\text{"}\texttt{A} = \text{"} + \rightarrow \texttt{STR}(\texttt{A}))$$

`CLEAR` clears the screen.
`FREEZE(7)` stops the program and freezes the screen so that you can see the results you have previously written on it.

6.6 Sequence of instructions, or "block"

A "block" is a sequence of one or more instructions.

6.6.1 Translation in algorithmic language

In algorithmic language, you can use either the space or a newline to terminate an instruction.

6.6.2 Translation for the HP49G in RPN mode

Like the algorithmic language, spaces and newlines act as instruction separators.

6.6.3 Translation for the HP49G int algebraic mode

When the `HP49G` works in algebraic mode, `;` acts as instruction separator. You can type `;` pressing `red-shift SPC` simultaneously.

6.7 Store instruction

The store instruction is used to store a value or an expression into a variable.

6.7.1 Translation in Algorithmic language

In algorithmic language, the store operation is denoted, for example, by:
`2*A->B` to store `2*A` into `B`.

6.7.2 Translation for the HP49G in RPN mode

When the `HP49G` works in RPN mode, you must use the postfix notation and the `STO` command:

```
2 A * 'B' STO
```

6.7.3 Translation for the HP49G in Algebraic mode

when the `HP49G` works in algebraic mode, you use the `STO` key, that is displayed on the calculator screen as: ▷ (and will be denoted here by `STO`▷).

6.8 Conditional branch instructions

6.8.1 Translation in algorithmic language

`if` *condition* `then` *action* `end if`
`if` *condition* `then` *action1* `else` *action2* `end if`
Example :

```
if A = 10 or A < B then B-A->B else A-B->A end if
```

6.8.2 Translation for the HP49G in RPN mode

`IF` *condition* `THEN` *action* `END`
`IF` *condition* `THEN` *action1 action2* `END`

Warning : you must use the postfix notation and `==` to denote the equality test operator.

The above mentioned example can be written as:

```
IF A 10 == A B < OR THEN B A - 'B' STO ELSE A B - 'A' STO END
```

you can also write:

```
IF 'A==10' 'A < B' OR THEN ...
```

6.8.3 Translation for the HP49G in Algebraic mode

IF *condition* THEN *action* END
IF *condition* THEN *action1* ELSE *action2* END
Example : Pay attention to always use == to denote the equality test!
IF A == 10 OR A < B THEN B-A STO▷ B ELSE A-B STO▷ A END

6.9 "For" loops

6.9.1 Translation in algorithmic language

for I from A to B do *action* end for
for I from A to B (step P) do *action* end for

6.9.2 Translation for the HP49G in RPN mode

A B FOR I *action* NEXT
   A B FOR I *action* P STEP

6.9.3 Translation for the HP49G in Algebraic mode

FOR (I , A , B) *action* NEXT
   FOR (I , A , B) *action* STEP P

It is not necessary to declare the loop variable I in advance.
I is automatically declared and initialized by: FOR (I,.,.)...

6.10 "While" loops

6.10.1 Translation in algorithmic language

while *condition* do *action* end while

6.10.2 Translation for the HP49G in RPN mode

WHILE *condition* REPEAT *action* END

6.10.3 Translation for the HP49G in Algebraic mode

WHILE *condition* REPEAT *action* END

6.11 Boolean (or conditional) expressions

A boolean (or conditional) expression is a function having a boolean value, that is, either `true` or `false`.

6.11.1 Translation in algorithmic language

To express a simple condition you use the following operators: `=` `<` `>` $\leq$ $\geq$ $\neq$

6.11.2 Translation for the HP49G in RPN mode

Warning, you must use the postfix notation and `==` to denote the equality test operator.

6.11.3 Translation for the HP49G in Algebraic mode

Warning, for the `HP49G`, the equality test operator is denoted by: `==`

6.12 Logical operators

6.12.1 Translation in algorithmic language

To express complex conditions, you can use the following logical operators: `or`, `and`, and `not`.

6.12.2 Translation for the HP49G in RPN mode

Warning, you must use the postfix notation.
`or`, `and`, and `not` are denoted, respectively, by `OR`, `AND`, and `NOT`.

6.12.3 Translation for the HP49G in Algebraic mode

`or`, `and`, and `not` are denoted on the `HP49G` by `OR`, `AND`, and `NOT`.

6.13 Functions

In a function, you do not perform data entry explicitly:
instead, you use function parameters, that are automatically initialized when the function is called.

In a function, you must be able to reuse the result at will:
in algorithmic language, you use the `return` command instead of `print`.

6.13.1 Translation in algorithmic language

You can write, for example:

```
function addition(A,B)
return A+B
end function
```

This means that:

- The result of the function will be displayed when the function is executed.
- The function can be used in an expression.

6.13.2 Translation for the HP49G in RPN mode

When the `HP49G` works in RPN mode, it is assumed that function arguments are on the stack when the function is invoked.

Inside the function, the arguments become local variables initialized with the topmost elements of the stack. The result of the function is pushed on the stack.
The example above can be written as:

```
≪→ A B
 ≪ A B + ≫
≫
'ADDITION' STO
```

To invoke the function you must push the intended values of `A` and `B` on stack levels 2 and 1; after the execution of `ADDITION` their sum will be on stack level 1.

6.13.3 Translation for the HP49G in Algebraic mode

The example above can be written as:

```
  ≪→ A B
 ≪ A + B ≫
≫
```

`STO` ▷ `ADDITION`

To invoke the function, you enter:

`ADDITION`(4, 5)

6.14 Lists

6.14.1 Translation in algorithmic language

In the algorithmic language, { and } are used as list delimiters.
For example, {} denotes the empty list, and {1, 2, 3} is a list of 3 elements.
The + operator can be used to concatenate two lists together or a list with an element:
`{1, 2, 3}->TAB`
`TAB + 4 ->TAB` (now `TAB` contains {1, 2, 3, 4})
`TAB[2]` denotes the second element of `TAB`, 2 in this example.

6.14.2 Translation for the HP49G in RPN mode

It is not necessary to declare the maximum length of a list in advance.
The list delimiters are { and }. For example {1 2 3} is a list of 3 elements, and {} is an empty list.
You can retrieve the `P`-th element of list `L` and put it on the stack using:
`L P GET` or `'L' P GET`
To update the `P`-th element of `L` (for example, set it to 0) you must enter:
`'L' P 0 PUT` or `L P 0 PUT 'L' STO`
Actually, `L P 0 PUT` leaves the updated list on the stack, and:
`'L' P 0 PUT` updates list `L` in-place instead.
The + operator can be used to concatenate two lists together or a list with an element.

6.14.3 Translation for the HP49G in algebraic mode

It is not necessary to declare the maximum length of a list in advance.
The list delimiters are { and }. For example {1 2 3} is a list of 3 elements, and {} is an empty list.
You can retrieve the `P`-th element of list `L` using:
`L[P]` or `GET (L, P )`
To update the `P`-th element of `L` (for example, set it to 0) you must enter:

`PUT(L, P, 0) STO▷ L`
or
`PUT('L', P, 0)`
Actually, `PUT(L, P, 0)` returns the updated list, and:
`PUT ('L', P, 0)` updates list `L` in-place instead.
The + operator can be used to concatenate two lists together or a list with an element. The `SEQ` command allows you to create a list; for example, if you type:

$$\texttt{SEQ}('\texttt{X}*\texttt{X}','\texttt{X}',4,10,1)$$

you obtain:

$$\{16,25,36,49,64,81,100\}$$

6.15 Example: The sieve of Erastothenes

6.15.1 Description

To find the prime numbers less than or equal to N :

1. Put the numbers from 1 to N into a list.
2. Mark 1 and store 2 into the variable P.
   If $P \cdot P \leq N$ we must process all elements from P to N.
3. Mark all multiples of P starting from $P \cdot P$.
4. Increment P by 1
   If $P \cdot P$ is less than or equal to N, we must process the non-marked elements from P to N.
5. Store into P the smallest non-marked elements of the list.
6. Repeat steps 3, 4 and 5 while $P \cdot P$ is less than or equal to N.

6.15.2 Algorithmic language

```
function crible(N)
local TAB PREM I P
// TAB and PREM are lists
{} ->TAB
{} ->PREM
for I from 2 to N do
```

```
  TAB+I -> TAB
end for
0 +TAB -> TAB
2 -> P
// Done steps 1 and 2
// marking has been implemented replacing the element to be marked with 0
// TAB is the list 0 2 3 4 ...N

while P*P ≤ N do

  for I from P to int(N/P) do
    0 -> TAB[I*P]
  end for
// We marked all multiples of P starting from P*P
  P+1 -> P
// We search the smallest number <= N, not marked, between P and N

  while (P*P ≤ N) and (TAB[P]=0) do

    P+1 -> P
  end while
end while
// We copy the result into PREM
for I from 2 to N do

  if TAB[I] ≠ 0 then

    PREM +I -> PREM
  end if
end for
return PREM
```

6.15.3 HP49G, RPN mode

This is the program `CRIBLE`:

`N` is the parameter whose actual value must be on the stack.

The local variables are:

`P` and `I` (integers),

`TA` and `PREM` (lists).

```
≪ {} {} 2 1 → N TA PREM P I
  ≪ 0 'X' 'X' 2 N 1 SEQ + 'TA' STO
    WHILE P P * N ≤ REPEAT
```

```
      P N P / FLOOR FOR I
         TA I P * 0 PUT 'TA' STO
      NEXT
      1 'P' STO+
      WHILE P P * N ≤ TA P GET 0 == AND REPEAT
         1 'P' STO+
      END
   END
   2 N FOR I
      IF TA I GET 0 ≠ THEN
         I 'PREM' STO+
      END
   NEXT
   PREM
 ≫
≫
```

6.15.4 HP49G, Algebraic mode

This is the program CRIBLE; the user must type, for example: CRIBLE(100), to execute it.

```
≪ → N
  ≪ 0+SEQ('I','I',1,N,1) → TA
    ≪ 2 → P
      ≪ WHILE P ∗ P ≤ N REPEAT
        FOR (I , P , FLOOR(N/P))
          PUT('TA',I*P,0)
        NEXT;
         P + 1 STO ▷ P;
        WHILE P*P ≤ N AND GET(TA,P) == 0 REPEAT
          P + 1 STO ▷ P
        END
      END;
      {2} → PREM
        ≪ FOR (I, 3, N)
            IF TA(I) ≠ 0 THEN
              PREM + I STO ▷ PREM;
            END
        NEXT;
```

```
            PREM
              ≫
           ≫
        ≫
     ≫
  ≫ STO▷ CRIBLE
```

Chapter 7

Arithmetic programs

7.1 Calculating the GCD using the Euclide's algorithm

This algorithm is rooted on the recursive definition of GCD:

$$\begin{aligned} GCD(A,0) &= A \\ GCD(A,B) &= GCD(B, A \ mod \ B) \ if \ B \neq 0 \end{aligned}$$

The algorithm can be described as follows:
we carry out this sequence of euclidean divisions:

$$\begin{aligned} A = &\quad B \times Q_1 + R_1 \quad 0 \leq R_1 < B \\ B = &\quad R_1 \times Q_2 + R_2 \quad 0 \leq R_2 < R_1 \\ R_1 = &\quad R_2 \times Q_3 + R_3 \quad 0 \leq R_3 < R_2 \\ &..... \end{aligned}$$

After a finite number of steps, it exists an integer n so that: $R_n = 0$.
We obtain then:
$GCD(A,B) = GCD(B,R_1) = ...$
$GCD(R_{n-1}, R_n) = GCD(R_{n-1}, 0) = R_{n-1}$

7.1.1 Algorithmic language

- Iterative implementation
  If B $\neq$ 0 we calculate R=A mod B then, using B instead of A (storing

B into A) and R instead of B (storing R into B) we repeat the process until B=0; when this happens, the GCD is A.

```
function GCD(A,B)
local R

while B ≠ 0 do

  A mod B->R
  B->A
  R->B
end while
return A
end function
```

- Recursive implementation
  We simply write out the recursive definition given above.

```
function GCD(A,B)

if B ≠ 0 then

  return GCD(B,A mod B)
  else
  return A
end if
end function
```

7.1.2 HP49G, RPN mode

- Iterative implementation

```
≪ 0 → A B R
   ≪ WHILE B 0 ≠ REPEAT
        A B MOD 'R' STO
        B 'A' STO
        R 'B' STO
      END
      A
   ≫
≫
```

- Recursive implementation

```
≪ 0 → A B
   ≪ IF B 0 ≠ THEN
         B A B MOD PGCDR
      ELSE
         A
      END
   ≫
≫
```

The program must be stored into variable PGCDR.

7.1.3 HP49G, Algebraic mode

- Iterative implementation

```
≪ → A,B
   ≪ 0 → R
      ≪ WHILE B ≠ 0 REPEAT
            A MOD B STO▷ R;
            B STO▷ A;
            R STO▷ B
         END;
         A
      ≫
   ≫
≫ STO▷ PGCD
```

To execute the program you enter, for example, PGCD(45,75).

- Recursive implementation

```
≪ → A,B
   ≪ IF B ≠ 0 THEN
         PGCDR(B,A MOD B)
      ELSE
         A
      END
   ≫
≫ STO▷ PGCDR
```

To execute the program you enter, for example, PGCDR(45,75).
Notice:
If you use the symbolic function IREMAINDER instead of MOD in the pro-

grams above, `PGCD` (or `PGCDR`) can then have Gauss integers as arguments, too.

7.2 Bézout identity using the Euclide's algorithm

In this section, the function `Bezout(A,B)` returns the list {U, V, GCD(A,B)}, where `U` and `V` satisfy: $A \times U + B \times V = GCD(A, B)$.

7.2.1 Iterative implementation without lists

The Euclide's algorithm allows us to find a pair U and V satisfying:
$A \times U + B \times V = GCD(A, B)$
Actually, if we denote A_0 and B_0 the starting values of A and B, we have:

$$\begin{aligned} A &= A_0 \times U + B_0 \times V \quad \text{with} \quad U = 1 \quad \text{and} \quad V = 0 \\ B &= A_0 \times W + B_0 \times X \quad \text{with} \quad W = 0 \quad \text{and} \quad X = 1 \end{aligned}$$

Next, we make A, B, U, V, W, X evolve so that the above relations continue to be satisfied.
If:
$A = B \times Q + R \quad 0 \leq R < B \quad (R = A \ mod \ B \ and \ Q = E(A/B))$
We can write:

$$R = A - B \times Q = A_0 \times (U - W \times Q) + B_0 \times (V - X \times Q) =$$
$$A_0 \times S + B_0 \times T \quad \text{with} \quad S = U - W \times Q \quad \text{and} \quad T = V - X \times Q$$

We must now repeat the process with B in place of A (B->A W->U X->V) and R in place of B (R->B S->W T->X).

We can write the whole algorithm as follows:

```
function Bezout (A,B)
local U,V,W,X,S,T,Q,R
1->U 0->V 0->W 1->X

while B ≠ 0 do

A mod B->R
E(A/B)->Q
//R=A-B*Q
U-W*Q->S
V-X*Q->T
```

```
B->A W->U X->V
R->B S->W T->X
end while
return {U, V, A}
end function
```

7.2.2 Iterative implementation with lists

The algorithm above can be simplified using less variables: in order to do this, we introduce the lists LA, LB, and LR to store the terns {U, V, A}, {W, X, B} and {S, T, R}.

```
function Bezout (A,B)
local LA LB LR
{1, 0, A}->LA
{0, 1, B}->LB

while LB[3] ≠ 0 do

LA-LB*E(LA[3]/LB[3])->LR
LB->LA
LR->LB
end while
return LA
end function
```

7.2.3 Recursive version with lists

The Bezout function can be defined recursively by: $Bezout(A, 0) = \{1,\ 0,\ A\}$ If $B \neq 0$ we must define $Bezout(A, B)$ in function of $Bezout(B, R)$, where $R = A - B \times Q$ and $Q = E(A/B)$.

We have:

$$Bezout(B, R) = LT = \{W, X, gcd(B, R)\}$$
$$\text{with } W \times B + X \times R = gcd(B, R)$$

Therefore:

$$\begin{aligned} W \times B + X \times (A - B \times Q) &= gcd(B, R) \text{ or, again,} \\ X \times A + (W - X \times Q) \times B &= gcd(A, B). \end{aligned}$$

So, if $B \neq 0$ and $Bezout(B, R) = LT$ we have:

$Bezout(A,B) = \{LT[2],\ LT[1] - LT[2] \times Q,\ LT[3]\}$.

```
function Bezout (A,B)
local LT Q R

if B ≠ 0 do

E(A/B)->Q
A-B*Q->R
Bezout(B,R)->LT
return {LT[2], LT[1]-LT[2]*Q, LT[3]}
else return {1, 0, A}
end if
end function
```

7.2.4 HP49G, RPN mode

- Iterative implementation with lists.
  At the very beginning, A et B contain the two numbers for which we are seeking the Bézout identity, later they denote the lists LA and LB mentioned in the algorithm.

```
≪ {} → A B R
   ≪ {1 0} 'A' STO+
      {0 1} 'B' STO+
      WHILE B 3 GET 0 ≠ REPEAT
         A B A 3 GET B 3 GET / FLOOR * - 'R' STO
         B 'A' STO
         R 'B' STO
      END
      A
         ≫
≫
```

- Recursive implementation with lists

```
≪ {} → A B T
   ≪ IF B 0 ≠ THEN
         B A B MOD BEZOUR 'T' STO
         T 2 GET DUP A B / FLOOR *
         T 1 GET SWAP -
         T 3 GET  + + +
```

```
      ELSE
         {1 0} A +
      END
   ≫
≫
```

This program must be stored into the variable BEZOUR.

7.2.5 HP49G, Algebraic mode

- Iterative implementation with lists.
  At the very beginning, A et B contain the two numbers for which we are seeking the Bézout identity, later they denote the lists LA and LB mentioned in the algorithm.

```
≪ → A,B
   ≪ {} → R
      ≪ {1,0,A} STO▷ A;
         {0,1,B} STO▷ B;
         WHILE B[3] ≠ 0 REPEAT
            A − B ∗ FLOOR(A[3]/B[3]) STO▷ R;
            B STO▷ A;
            R STO▷ B
         END
         A
      ≫
   ≫
≫ STO▷ BEZOUT
```

  To execute the program you type, for example, BEZOUT(45,75).

- Recursive implementation with lists

```
≪ → A,B
   ≪ {} → T
      ≪ IF B ≠ 0 THEN
            BEZOUR(B, A MOD B) STO▷ T;
            {T[2], T[1] − T[2] ∗ FLOOR(A/B), T[3]}
         ELSE
            {1,0,A}
         END
      ≫
   ≫
≫ STO▷ BEZOUR
```

To execute the program you type, for example, `BEZOUR(45,75)`.
Notice :
If you use the symbolic function `IREMAINDER` instead of `MOD` in the programs above, `PGCD` (or `PGCDR`) can then have Gauss integers as arguments, too.

7.3 Factorization

7.3.1 Algorithms and their translations

- First algorithm
  We check, for all integers D from 2 to N, whether N is divided by D.
  If D divides N, we search the factors of N/D ... and so on.
  All factors are stored into the list FACT.

```
function facprem(N)
local D FACT
2 -> D
{} -> FACT

while N ≠ 1 do

  if N mod D = 0 then
    FACT + D -> FACT
    N/D -> N
   else
    D+1 -> D
 end if
end while
return FACT
end function
```

- First improvement
  We only check potential factors between 2 and $E(\sqrt{N})$.

```
function facprem(N)
local D FACT
2 -> D
{} -> FACT

while D*D ≤ N do
```

```
  if N mod D = 0 then
    FACT + D -> FACT
    N/D-> N
   else
    D+1 -> D
 end if
end while
FACT + N -> FACT
return FACT
end function
```

- Second improvement
  We check if 2 divides N, then we check only odd potential factors D between 3 and $E(\sqrt{N})$.
  In the FACT list, each factor is now followed by its exponent:
  facprem(12)={2,2,3,1}.

```
function  facprem(N)
local K D FACT
{}->FACT
0 -> K
while N mod 2 = 0 do
    K+1 -> K
    N/2 -> N
end while

if K ≠0 then

    FACT + {2 K} -> FACT
end if
3 ->D

while D*D ≤ N do

  0 -> K
  while N mod D = 0 do
    K+1 -> K
    N/D -> N
  end while

  if K ≠0 then
```

```
        FACT + {D K} -> FACT
      end if
      D+2 -> D
    end while

    if N ≠ 1 then

    FACT + {N 1} -> FACT
    end if
    return FACT
    end function
```

7.3.2 HP49G, RPN mode

This is the translation of the second improvement:

```
≪ 0 3 {} → N K D FACT
  ≪ WHILE N 2 MOD 0 == REPEAT
       1 'K' STO+
       'N' 2 STO/
     END
     IF K 0 ≠ THEN
       {2 K} 'FACT' STO
     END
     WHILE N D D * ≥ REPEAT
       0 'K' STO
       WHILE N D MOD 0 == REPEAT
         1 'K' STO+
         'N' D STO/
       END
       IF K 0 ≠ THEN
         {D K} 'FACT' STO+
       END
       2 'D' STO+
     END
     IF N 1 ≠ THEN
       {N 1} 'FACT' STO+
     END
  ≫
≫
```

7.3.3 HP49G, algebraic mode

```
≪ → N
  ≪ 0 → K
    ≪ WHILE N MOD 2 == 0 REPEAT
        K + 1 STO▷ K;
        N/2 STO▷ N
      END;
      {} → FACTO
        ≪ IF K ≠ 0 THEN
            FACTO + {2, K} STO▷ FACTO
          END;
          3 → D
            ≪ WHILE D * D ≤ N REPEAT
                0 STO▷ K;
                WHILE N MOD D == 0 REPEAT
                  K + 1 STO▷ K;
                  N/D STO▷ N;
                END;
                IF K ≠ 0 THEN
                  FACTO + {D, K} STO▷ FACTO
                END;
                D + 2 STO▷ D
              END;
              IF N ≠ 1 THEN
                FACTO + {N, 1} STO▷ FACTO
              END;
              FACTO;
            ≫
        ≫
    ≫
  ≫
≫ STO▷ FACTEUR
```

To execute the program you type, for example, `FACTEUR(45)`.

7.4 Calculation of $A^P \ mod \ N$

7.4.1 Algorithmic language

- First algorithm
  We use local variables PUIS et I.
  We make an iterative program so that at each step, PUIS represents $A^I \pmod N$.

```
function puismod (A, P, N)
local PUIS, I
1->PUIS
for I from 1 to P do
  A*PUIS mod N ->PUIS
end for
return PUIS
end function
```

- Second algorithm
  We use only one local variable, PUI, but we update P so that at each iteration step we always have:
  $result = PUI * A^P \pmod N$

```
function puismod (A, P, N)
local PUI
1->PUI
while  P>0  do
  A*PUI mod N ->PUI
  P-1->P
end while
return PUI
end function
```

- Third algorithm
  We can improve the previous program by noticing that:
  $A^{2*P} = (A * A)^P$.
  So, when P is even, the following is true:
  $PUI * A^P = PUI * (A * A)^{P/2} \pmod N$
  and when P is odd, the following is true:
  $PUI * A^P = PUI * A * A^{P-1} \pmod N$.
  The result is a fast algorithm to compute $A^P \pmod N$.

```
function puismod (A, P, N)
local PUI
1->PUI
while  P>0  do
  if P mod 2 =0 then
    P/2->P
    A*A mod N->A
  else
    A*PUI mod N ->PUI
    P-1->P
  end if
end while
return PUI
end function
```

We can also notice that if P is odd, then P-1 is even.
We can write:

```
function puismod (A, P, N)
local PUI
1->PUI
while  P>0  do
  if P mod 2 =1 then
    A*PUI mod N ->PUI
    P-1->P
  end if
P/2->P
A*A mod N->A
end while
return PUI
end functions
```

- Recursive program
  We can leverage the following recurrence relation:
  $A^0 = 1 \;\; A^{P+1} \pmod N = (A^P \pmod N) * A \pmod N$

```
function puimod(A, P, N)
if P>0 then
return puimod(A, P-1, N)*A mod N
else
return 1
```

```
end if
end function
```

- Fast, recursive program

```
function puimod(A, P, N)
if P>0 then
  if P mod 2 =0 then
    return puimod(A*A, P/2, N)
  else
    return puimod(A, P-1, N)*A mod N
  end if
else
return 1
end if
end function
```

7.4.2 HP49G, RPN mode

The user must push on the stack :
A, P, N to obtain $A^P mod\ N$.
This is the translation of the fast, iterative algorithm:

```
≪ 1 → A P N PUI
  ≪ WHILE P 0 > REPEAT
       IF P 2 MOD 1 == THEN
          A PUI * N MOD 'PUI' STO
          'P' STO-
       END
       P 2 / 'P' STO
       A A * N MOD 'A' STO
     END
     PUI
  ≫
≫
```

We can store the program into PUIMOD (using 'PUIMOD' STO).

7.4.3 HP49G, Algebraic mode

This is the translation of the fast, iterative algorithm:

```
≪ → A P N
```

```
≪ 1 → PUI
   ≪ WHILE P > 0 REPEAT
        IF P MOD 2 == 1 THEN
           A * PUI MOD N STO▷ PUI
           P − 1 STO▷ P;
        END;
        P/2 STO▷ P;
        A * A MOD N STO▷ A;
     END;
     PUI
   ≫
≫
≫ STO▷ PUIMOD
```

We can type, for example, `PUIMOD(45,32,13)` to execute it.

7.5 The function "isprime"

7.5.1 Algorithmic language

- First algorithm

  We are about to write a boolean function of N, returning TRUE if N is prime, and FALSE if it is not.

  In order to do this, we check whether N has a factor $\neq 1$ and $\leq E(\sqrt{N})$ (integer part of the square root of N).

  The special case $N = 1$ is handled separately!

  We use a boolean variable PREM, initially set to TRUE and changed to be FALSE when we find a factor of N.

```
function isprime(N)
local PREM, I, J

E(√N) ->J

if N = 1 then
  FALSE->PREM
  else
  TRUE->PREM
```

```
end if
2->I

while PREM and I ≤J do

  if N mod I = 0 then
     FALSE->PREM
     else
     I+1->I
  end if
end while
return PREM
end function
```

- First improvement

  We notice that first of all, we can check if N is even and, if it is not, only check whether it has odd factors.

```
function iswprime(N)
local PREM, I, J

E(√N) ->J
if (N = 1) or (N mod 2 = 0) and N≠2 then

  FALSE->PREM
  else
  TRUE->PREM
end if
3->I

while PREM and I ≤J do

  if N mod I = 0 then
     FALSE->PREM
     else
     I+2->I
  end if
end while
return PREM
end function
```

- Second improvement

  We check if N cen be divided by 2 or by 3, else we check whether N has a factor that can be expressed as either $6 \times k - 1$ or $6 \times k + 1$.

```
function isprime(N)
local PREM, I, J

E(√N) ->J

if (N = 1) or (N mod 2 = 0) or ( N mod 3 = 0) then
  FALSE->PREM
  else
  TRUE->PREM
end if
if N=2 or N=3 then
TRUE->PREM
end if
5->I

while PREM and I ≤J do

  if (N mod I = 0) or (N mod I+2 =0) then
     FALSE->PREM
     else
     I+6->I
  end if
end while
return PREM
end function
```

7.5.2 HP49G, RPN mode

We translate the last algorithm listed above: the result is either 0 (false) or 1 (true).

```
≪ DUP √ FLOOR 0 5 → N J PREM I
 ≪ IF N 1 == N 2 MOD 0 = 3D = 3D OR N 3 MOD 0 == OR THEN
     0 'PREM' STO
   ELSE
     1 'PREM' STO
   END
```

```
    IF N 2 == N 3 == OR THEN
       1 'PREM' STO
    END
    WHILE PREM I J ≤ AND REPEAT
       IF N I MOD 0 == N I 2 + MOD 0 == OR THEN
          0 'PREM' STO
       ELSE
          I 6 + 'I' STO
       END
   END
   PREM
 ≫
≫
```

7.5.3 HP49G. Algebraic mode

```
≪ → N
   ≪ 0 → P
      ≪ IF N MOD 2 == 0 OR N MOD 3 == 0 OR N == 1THEN
            0 STO▷ P
         ELSE;
            1 STO▷ P;
         END;
         IF N == 2 OR N == 3 THEN
            1 STO▷ P;
         END;
         FLOOR(√N) → J
            ≪ 5 → I
               ≪ WHILE I ≤ J AND P REPEAT
                     IF N MOD I == 0 OR N MOD (I + 2) == 0 THEN
                        0 STO▷ P;
                     ELSE;
                        I + 6 STO▷ I;
                     END;
                  END;
                  P
               ≫
            ≫
      ≫
   ≫
```

≫ `STO▷ PREM?`
to execute this program you type, for example, `PREM?(45789)`.

7.6 Rabin's probabilistic method

If N is prime, all integers K less that N are prime with N; therefore, applying the Fermat theorem we can state that:

$K^{N-1} = 1 \pmod{N}$

for all integers K less than N. Instead, if N is not prime, integer values K satisfying:

$K^{N-1} = 1 \pmod{N}$

are rare.

To be more precise, it can be shown that if $N > 4$, the probabilty to randomly find such an integer K is less than 0.25.

An integer N satisfying $K^{N-1} = 1 \pmod{N}$ for 20 random trials of K is a preudo-prime integer.

The Rabin's probabilistic method consists of randomly generate an integer K ($1 < K < N$) and calculate :

$K^{N-1} \pmod{N}$

If $K^{N-1} = 1 \pmod{N}$ we repeat the process with a different value of K; if, instead, $K^{N-1} \neq 1 \pmod{N}$ we are sure that N is not prime.

If $K^{N-1} = 1 \pmod{N}$ for 20 random trials of K we can state that N is prime with a small probability of error, less than 0.25^{20}, that is about 10^{-12}.

Of course, this method is very fast, and is widely used to check whether very big integers are pseudo-prime.

7.6.1 Algorithmic language

We assume that: Hasard(N) returns a random integer between 0 and N-1.

We calculate:

$K^{N-1}\ mod\ N$

using the fast power algorithm described above (see page 158).

We denote as:

puismod(K, P, N) the function calculating and returning $K^P\ mod\ N$.

```
function isprime(N)
local K, I, P
```

```
1->I
1->P
while P = 1 and I < 20 do
hasard(N-2)+2->K
puismod(K, N-1, N)->P
I+1->I
end while
if P =1 then
return TRUE
else
return FALSE
end if
end function
```

7.6.2 HP49G, RPN mode

We assume that the function PUIMOD, taking from the stack three arguments A, K, N, and returning $A^K mod\ N$ is already available.

```
≪ 1 0 1 → N I K P
 ≪ 0 RDZ
   WHILE P 1 == 20 I > AND REPEAT
       1 'I' STO+
      RAND N 2 - * FLOOR 2 + 'K' STO
      K N 1 - N PUIMOD 'P' STO
   END
   IF P 1 == THEN
       1
    ELSE
       0
   END
 ≫
≫
```

7.6.3 HP49G, Algebraic mode

```
≪ → N
   ≪ 1 → I
      ≪ 0 → K
         ≪ 1 → P
            ≪ RDZ(0);
```

```
            WHILE P == 0 AND I < 20 REPEAT
               1 + I STO▷ I;
               FLOOR((N − 2) ∗ RAND) + 2 STO▷ K;
               PUIMOD(K, N − 1, N) STO▷ P;
            END;
            IF P == 1 THEN
               1
            ELSE
               0
            END;
            P
         ≫
      ≫
   ≫
≫
≫ STO▷ RABIN
```

To execute this program you can type, for example, `RABIN(45313)`.

Notice:
We can also use the built-in command `POWMOD` and write:

- In `RPN` mode:
  `N MODSTO`
  `K N 1 - POWMOD 'P' STO`
  instead of:
  `K N 1 - N PUIMOD 'P' STO`

- In `Algebraic mode` :
  `MODSTO(N);`
  `POWMOD(K,N-1) STO▷ P`
  instead of:
  `PUIMOD(K,N-1,N) STO▷ P`

Index

△ ◁ ▷ ▽, 8
⇐, 8
⟻, 8
→, 8
▷ STO▷ , 8
= ∼, 7

ABCUV, 61
ABS, 42, 80
ACOS2S, 50
ADDTMOD, 68
ARG, 42
ASIN2C, 50
ASIN2T, 51
ATAN2S, 51
AUGMENT, 99
AXL, 77
AXM, 77
AXQ, 80

BASIS, 98

C2P, 100
CASCFG, 7
CF, 9
CHINREM, 63
CHOLESKY, 95
CIRC, 101
CLEAR, 137
CONJ, 42
COPY, 24
CROSS, 80
CURL, 83
CUT, 24
CYCLOTOMIC, 99

DEFINE, 33
DERIV, 45, 82
DERVX, 44
DESOLVE, 90
DIAGMAP, 95
DISP, 137
DISTRIB, 101
DIV, 83
DIV2, 60
DIV2MOD , 69
DIVIS, 37, 59
DIVMOD, 69
DIVPC, 71
DOT, 80

EGCD, 60
EGV, 78
EGVL, 77
EPSX0, 92
EULER, 40
EVAL, 43
EXLR, 83
EXP2POW, 102
EXPAND, 43
EXPANDMOD, 70
EXPLN, 55

FACTOR, 36, 43, 58

FACTORMOD, 71
FACTORS, 36, 59
FC?, 9
FCOEF, 63
FDISTRIB, 101
FOURIER, 54
FREEZE, 137
FROOTS, 62
FS?, 9
FXND, 41, 66

GAMMA, 104
GAUSS, 81
GCD, 35, 57
GCDMOD, 70
GET, 142
GRAMSCHMIDT, 94
GROBADD, 109

HADAMARD, 77
HALFTAN, 52
HERMITE, 65
HESS, 82
HILBERT, 79
HORNER, 61

IABCUV, 39
IBASIS, 98
IBERNOULLI, 104
IBP, 48
ICHINREM, 39
IDIV2, 37
IEGCD, 39
ILAP, 91
IM, 42
IMAGE, 98
INPUT, 136
INTVX, 45
INVMOD, 70
IQUOT, 37
IREMAINDER, 37
ISOL, 85
ISOM, 96
ISPRIME?, 34, 38

JORDAN, 78

KER, 98

LAGRANGE, 64
LAP, 91
LAPL, 82
LCM, 36, 58
LCXM, 80
LDEC, 89
LEGENDRE, 64
LGCD, 35, 57
LIMIT, 46, 47, 75
LIN, 55
LINSOLVE, 88
LNAME, 93
LNCOLLECT, 56
LVAR, 93

MAD, 76
MKISOM, 96
MOD, 37
MODSTO, 68
MSGBOX, 137
MSLV, 103
MULTMOD, 69

NEG, 42
NEXTPRIME, 38

P2C, 100
PA2B2, 40
PARTFRAC, 67
PASTE, 24
PCAR, 78
PCOEF, 63
PLOT, 31, 108

PLOTADD, 31, 109
PMINI, 103
POWEXPAND, 102
POWMOD, 70
PREVAL, 44
PREVPRIME, 38
PROMPTSTO, 136
PROOT, 62
PROPFRAC, 41, 67
PSI, 105
Psi, 105
PTAYL, 61
PURGE, 25
PUT, 143

qr, 94
QUOT, 60
QXA, 80

RCL, 25
RE, 42
REF, 86
REMAINDER, 60
REORDER, 66
RESULTANT, 105
RISCH, 48
RREF, 87
rref, 86
RREFMOD, 71

SCROLL, 109
SEQ, 143
SERIES, 73
SEVAL, 106
SF, 9
SIGMA, 106
SIGMAVX, 107
SIGN, 42
SIGNTAB, 108
SIMP2, 35, 41, 66
SIMPLIFY, 102
SINCOS, 51
SOLVE, 84
SOLVEVX, 84
STO, 24
STURM, 99
STURMAB, 100
SUBST, 43, 90
SUBTMOD, 68
SYLVESTER, 81
SYST2MAT, 95

TABVAL, 108
TABVAR, 107
TAN2SC, 52
TAN2SC2, 52
TAYLOR0, 72
TAYLR , 72
TCHEBYCHEFF, 65
TCOLLECT, 50
TEXPAND, 48
TLIN, 49
TRAN, 75
TRIG, 53
TRIGCOS, 54
TRIGSIN, 53
TRIGTAN, 54
TRN, 76
TRUNC, 64
TSIMP, 57

VANDERMONDE, 79
VER, 107
VISIT, 133

XNUM, 93
XQ, 93

ZEROS, 62

Contents

0.1 Introduction 5
0.1.1 Turning on the calculator 5
0.1.2 What am I looking at? 5
0.2 Calculator modes 7
0.3 Notation 8
0.4 Flags 9

1 Important keys **11**
1.1 The APPS key 11
1.1.1 Plot functions 11
1.1.2 I/O functions 11
1.1.3 Constants library 12
1.1.4 Numeric solver 12
1.1.5 Time & date 12
1.1.6 Equation writer 13
1.1.7 File manager 13
1.1.8 Matrix writer 13
1.1.9 Text editor 13
1.1.10 Math menu 13
1.1.11 CAS menu 13
1.2 The MODE key 14
1.3 The TOOL key 14
1.4 The UNDO key (red-shift HIST) 14
1.5 The VAR key 14
1.6 The EQW key 15
1.7 The MTRW key (blue-shift EQW) 15
1.8 The SYMB key 15
1.9 The MTH (blue-shift SYMB) key 15
1.10 The UNITS (red-shift 6) key 16

1.11 The HIST key 16

2 Data entry **17**
2.1 The equation editor 17
2.1.1 Entering the equation writer 17
2.1.2 How to select? 17
2.1.3 How to modify an expression 21
2.1.4 How to enter AND, $\int$ and $\sum$ 22
2.1.5 Cursor mode 22
2.1.6 To view all 22
2.2 The matrix writer 23
2.3 The text editor 23
2.3.1 BEGIN END 23
2.3.2 COPY 24
2.3.3 CUT 24
2.3.4 PASTE 24
2.4 Variables 24
2.4.1 STO 24
2.4.2 RCL 25
2.4.3 PURGE 25
2.4.4 Predefined variables 26
2.5 Directories 26
2.5.1 Creating a directory 26
2.5.2 Working in a directory 27
2.5.3 Deleting, renaming, moving a directory 27

3 Plotting graphs **29**
3.1 Plot windows 29
3.1.1 Equation entry 29
3.1.2 Plot window 29
3.1.3 Graph display 29
3.1.4 Plot setup 30
3.1.5 Table setup 30
3.1.6 Table display 30
3.2 Plot setup 30
3.2.1 Plot type 30
3.2.2 The equation 31
3.2.3 Independent variable and equation types 31
3.3 Drawing the plot 32

4 Symbolic calculations **33**
4.1 Integers (and Gauss integers) . . . 33
4.1.1 Infinite-precision integers . . . 33
4.1.2 `DEFINE` . . . 33
4.1.3 `GCD` . . . 35
4.1.4 `LGCD` . . . 35
4.1.5 `SIMP2` . . . 35
4.1.6 `LCM` . . . 36
4.1.7 `FACTOR` . . . 36
4.1.8 `FACTORS` . . . 36
4.1.9 `DIVIS` . . . 37
4.1.10 `IQUOT` . . . 37
4.1.11 `IREMAINDER MOD` . . . 37
4.1.12 `IDIV2` . . . 37
4.1.13 `ISPRIME?` . . . 38
4.1.14 `NEXTPRIME` . . . 38
4.1.15 `PREVPRIME` . . . 38
4.1.16 `IEGCD` . . . 39
4.1.17 `IABCUV` . . . 39
4.1.18 `ICHINREM` . . . 39
4.1.19 `PA2B2` . . . 40
4.1.20 `EULER` . . . 40
4.2 Rationals . . . 40
4.2.1 `PROPFRAC` . . . 41
4.2.2 `FXND` . . . 41
4.2.3 `SIMP2` . . . 41
4.3 Reals . . . 42
4.4 Complex numbers . . . 42
4.5 Algebraic expressions . . . 43
4.5.1 `FACTOR` . . . 43
4.5.2 `EXPAND EVAL` . . . 43
4.5.3 `SUBST` . . . 43
4.5.4 `PREVAL` . . . 44
4.6 Functions . . . 44
4.6.1 `DERVX` . . . 44
4.6.2 `DERIV` . . . 45
4.6.3 `INTVX` . . . 45
4.6.4 `LIMIT` . . . 46
4.6.5 `LIMIT` and $\int$ . . . 47
4.6.6 `IBP` . . . 48

4.6.7 RISCH . . . 48
4.7 Trigonometric expressions . . . 48
4.7.1 TEXPAND . . . 48
4.7.2 TLIN . . . 49
4.7.3 TCOLLECT . . . 50
4.7.4 ACOS2S . . . 50
4.7.5 ASIN2C . . . 50
4.7.6 ASIN2T . . . 51
4.7.7 ATAN2S . . . 51
4.7.8 SINCOS . . . 51
4.7.9 TAN2SC . . . 52
4.7.10 TAN2SC2 . . . 52
4.7.11 HALFTAN . . . 52
4.7.12 TRIG . . . 53
4.7.13 TRIGSIN . . . 53
4.7.14 TRIGCOS . . . 54
4.7.15 TRIGTAN . . . 54
4.7.16 FOURIER . . . 54
4.8 Exponentials and Logarithms . . . 55
4.8.1 EXPLN . . . 55
4.8.2 LIN . . . 55
4.8.3 LNCOLLECT . . . 56
4.8.4 TSIMP . . . 57
4.9 Polynomials . . . 57
4.9.1 GCD . . . 57
4.9.2 LGCD . . . 57
4.9.3 SIMP2 . . . 58
4.9.4 LCM . . . 58
4.9.5 FACTOR . . . 58
4.9.6 FACTORS . . . 59
4.9.7 DIVIS . . . 59
4.9.8 QUOT . . . 60
4.9.9 REMAINDER . . . 60
4.9.10 DIV2 . . . 60
4.9.11 EGCD . . . 60
4.9.12 ABCUV . . . 61
4.9.13 HORNER . . . 61
4.9.14 PTAYL . . . 61
4.9.15 ZEROS . . . 62
4.9.16 PROOT . . . 62

4.9.17 FROOTS . . . 62
4.9.18 PCOEF . . . 63
4.9.19 FCOEF . . . 63
4.9.20 CHINREM . . . 63
4.9.21 TRUNC . . . 64
4.9.22 LAGRANGE . . . 64
4.9.23 LEGENDRE . . . 64
4.9.24 HERMITE . . . 65
4.9.25 TCHEBYCHEFF . . . 65
4.9.26 REORDER . . . 66
4.10 Rational fractions . . . 66
4.10.1 FXND . . . 66
4.10.2 SIMP2 . . . 66
4.10.3 PROPFRAC . . . 67
4.10.4 PARTFRAC . . . 67
4.11 Modular calculations . . . 68
4.11.1 MODSTO . . . 68
4.11.2 ADDTMOD . . . 68
4.11.3 SUBTMOD . . . 68
4.11.4 MULTMOD . . . 69
4.11.5 DIV2MOD . . . 69
4.11.6 DIVMOD . . . 69
4.11.7 POWMOD . . . 70
4.11.8 INVMOD . . . 70
4.11.9 GCDMOD . . . 70
4.11.10 EXPANDMOD . . . 70
4.11.11 FACTORMOD . . . 71
4.11.12 RREFMOD . . . 71
4.12 Limited and asymptotic expansions . . . 71
4.12.1 DIVPC . . . 71
4.12.2 TAYLORO . . . 72
4.12.3 TAYLR . . . 72
4.12.4 SERIES . . . 73
4.12.5 LIMIT . . . 75
4.13 Matrices . . . 75
4.13.1 TRAN . . . 75
4.13.2 TRN . . . 76
4.13.3 MAD . . . 76
4.13.4 HADAMARD . . . 77
4.13.5 AXM . . . 77

4.13.6 AXL . . . 77
4.13.7 EGVL . . . 77
4.13.8 EGV . . . 78
4.13.9 PCAR . . . 78
4.13.10 JORDAN . . . 78
4.13.11 HILBERT . . . 79
4.13.12 VANDERMONDE . . . 79
4.13.13 LCXM . . . 80
4.14 Vectors . . . 80
4.15 Quadratic forms . . . 80
4.15.1 QXA . . . 80
4.15.2 AXQ . . . 80
4.15.3 GAUSS . . . 81
4.15.4 SYLVESTER . . . 81
4.16 Functions of multiple variables . . . 82
4.16.1 DERIV . . . 82
4.16.2 LAPL . . . 82
4.16.3 HESS . . . 82
4.16.4 DIV . . . 83
4.16.5 CURL . . . 83
4.17 Equations . . . 83
4.17.1 EXLR . . . 83
4.17.2 SOLVEVX . . . 84
4.17.3 SOLVE . . . 84
4.17.4 ISOL . . . 85
4.18 Linear systems . . . 86
4.18.1 REF . . . 86
4.18.2 rref . . . 86
4.18.3 RREF . . . 87
4.18.4 LINSOLVE . . . 88
4.19 Differential equations . . . 89
4.19.1 LDEC . . . 89
4.19.2 DESOLVE and SUBST . . . 90
4.19.3 LAP ILAP . . . 91
4.20 Other functions . . . 92
4.20.1 EPSXO . . . 92
4.20.2 LVAR . . . 93
4.20.3 LNAME . . . 93
4.20.4 XNUM . . . 93
4.20.5 XQ . . . 93

4.21 New commands 94
4.21.1 qr 94
4.21.2 GRAMSCHMIDT 94
4.21.3 SYST2MAT 95
4.21.4 CHOLESKY 95
4.21.5 DIAGMAP 95
4.21.6 ISOM 96
4.21.7 MKISOM 96
4.21.8 KER 98
4.21.9 IMAGE 98
4.21.10 IBASIS 98
4.21.11 BASIS 98
4.21.12 AUGMENT 99
4.21.13 CYCLOTOMIC 99
4.21.14 STURM 99
4.21.15 STURMAB 100
4.21.16 P2C 100
4.21.17 C2P 100
4.21.18 CIRC 101
4.21.19 FDISTRIB 101
4.21.20 DISTRIB 101
4.21.21 POWEXPAND 102
4.21.22 SIMPLIFY 102
4.21.23 EXP2POW 102
4.21.24 MSLV 103
4.21.25 PMINI 103
4.21.26 IBERNOULLI 104
4.21.27 GAMMA 104
4.21.28 PSI 105
4.21.29 Psi 105
4.21.30 RESULTANT 105
4.21.31 SEVAL 106
4.21.32 SIGMA 106
4.21.33 SIGMAVX 107
4.21.34 VER 107
4.21.35 TABVAR 107
4.21.36 SIGNTAB 108
4.21.37 TABVAL 108
4.21.38 PLOT 108
4.21.39 PLOTADD 109

4.21.40 SCROLL . . . 109
4.21.41 GROBADD . . . 109

5 Bac 99 and HP49G **111**
5.1 Introduction . . . 111
5.2 Exercise 1 . . . 112
5.3 Exercise 2 (specialized) . . . 117
5.4 Exercise 2 (not specialized) . . . 121
5.5 Problem . . . 125
5.6 Conclusion . . . 132

6 Programming **133**
6.1 Programming in algebraic mode . . . 133
6.1.1 Entering a program . . . 133
6.1.2 Saving a program . . . 133
6.1.3 Editing a program . . . 133
6.1.4 Executing a program . . . 134
6.1.5 Modifying a program and saving it with another name . 134
6.2 Program comments . . . 134
6.3 Variables . . . 134
6.3.1 Variable names . . . 134
6.3.2 Local variables . . . 135
6.3.3 Parameters . . . 135
6.4 Data input . . . 136
6.4.1 Translation in algorithmic language . . . 136
6.4.2 Translation for the HP49G in RPN mode . . . 136
6.4.3 Translation for the HP49G in Algebraic mode . . . 136
6.5 Data output . . . 136
6.5.1 Translation in algorithmic language . . . 136
6.5.2 Translation for the HP49G in RPN mode . . . 137
6.5.3 Translation for the HP49G in Algebraic mode . . . 137
6.6 Sequence of instructions, or "block" . . . 137
6.6.1 Translation in algorithmic language . . . 137
6.6.2 Translation for the HP49G in RPN mode . . . 137
6.6.3 Translation for the HP49G int algebraic mode . . . 137
6.7 Store instruction . . . 138
6.7.1 Translation in Algorithmic language . . . 138
6.7.2 Translation for the HP49G in RPN mode . . . 138
6.7.3 Translation for the HP49G in Algebraic mode . . . 138
6.8 Conditional branch instructions . . . 138

6.8.1 Translation in algorithmic language . . . 138
6.8.2 Translation for the HP49G in RPN mode . . . 138
6.8.3 Translation for the HP49G in Algebraic mode . . . 139
6.9 "For" loops . . . 139
6.9.1 Translation in algorithmic language . . . 139
6.9.2 Translation for the HP49G in RPN mode . . . 139
6.9.3 Translation for the HP49G in Algebraic mode . . . 139
6.10 "While" loops . . . 139
6.10.1 Translation in algorithmic language . . . 139
6.10.2 Translation for the HP49G in RPN mode . . . 139
6.10.3 Translation for the HP49G in Algebraic mode . . . 139
6.11 Boolean (or conditional) expressions . . . 140
6.11.1 Translation in algorithmic language . . . 140
6.11.2 Translation for the HP49G in RPN mode . . . 140
6.11.3 Translation for the HP49G in Algebraic mode . . . 140
6.12 Logical operators . . . 140
6.12.1 Translation in algorithmic language . . . 140
6.12.2 Translation for the HP49G in RPN mode . . . 140
6.12.3 Translation for the HP49G in Algebraic mode . . . 140
6.13 Functions . . . 140
6.13.1 Translation in algorithmic language . . . 141
6.13.2 Translation for the HP49G in RPN mode . . . 141
6.13.3 Translation for the HP49G in Algebraic mode . . . 141
6.14 Lists . . . 142
6.14.1 Translation in algorithmic language . . . 142
6.14.2 Translation for the HP49G in RPN mode . . . 142
6.14.3 Translation for the HP49G in algebraic mode . . . 142
6.15 Example: The sieve of Erastothenes . . . 143
6.15.1 Description . . . 143
6.15.2 Algorithmic language . . . 143
6.15.3 HP49G, RPN mode . . . 144
6.15.4 HP49G, Algebraic mode . . . 145

7 Arithmetic programs **147**
7.1 Calculating the GCD using the Euclide's algorithm . . . 147
7.1.1 Algorithmic language . . . 147
7.1.2 HP49G, RPN mode . . . 148
7.1.3 HP49G, Algebraic mode . . . 149
7.2 Bézout identity using the Euclide's algorithm . . . 150
7.2.1 Iterative implementation without lists . . . 150

7.2.2 Iterative implementation with lists 151
7.2.3 Recursive version with lists 151
7.2.4 HP49G, RPN mode 152
7.2.5 HP49G, Algebraic mode 153
7.3 Factorization 154
7.3.1 Algorithms and their translations 154
7.3.2 HP49G, RPN mode 156
7.3.3 HP49G, algebraic mode 157
7.4 Calculation of $A^P \ mod \ N$ 158
7.4.1 Algorithmic language 158
7.4.2 HP49G, RPN mode 160
7.4.3 HP49G, Algebraic mode 160
7.5 The function "isprime" 161
7.5.1 Algorithmic language 161
7.5.2 HP49G, RPN mode 163
7.5.3 HP49G. Algebraic mode 164
7.6 Rabin's probabilistic method 165
7.6.1 Algorithmic language 165
7.6.2 HP49G, RPN mode 166
7.6.3 HP49G, Algebraic mode 166