**GNU Calc Reference Card**
(for GNU Emacs version 28)

**Starting and Stopping**
- start/stop standard Calc: C-x * *
- start/stop X keypad Calc: C-x * *
- start/stop either: C-x * *
- stop standard Calc: C-x * k
- Calc tutorial: C-x * t
- run Calc in other window: C-x * o
- quick calculation in minibuffer: C-x * q

**Getting Help**
The h prefix key is Calc's analogue of C-h in Emacs.
- quick summary of keys: ?
- describe key briefly: C-h k
- describe key fully: C-h t
- describe function or command: C-h f
- read Info manual: C-x * i
- read full Calc summary: C-x * s

**Error Recovery**
- abort command in progress: C-g
- display recent error messages: C-w
- undo last operation: C-u
- redo last operation: C-d
- recall last arguments: M-RET
- edit top of stack: 
- reset Calc to initial state: C-x * 0 (zero)

**Transferring Data**
- grab region from a buffer: C-x * g
- grab rectangle from a buffer: C-x * r
- grab rectangle, summing columns: C-x * i
- grab rectangle, summing rows: C-x * i
- yank data to a buffer: C-x * y

Also, try C-k/C-y or X cut and paste.

**Examples**
In RPN, enter numbers first, separated by RET if necessary, then type the operator. To enter a calculation in algebraic form, press the apostrophe first.

<table>
<thead>
<tr>
<th>RPN style</th>
<th>algebraic style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: 2 RET 3 +</td>
<td>'+3' RET</td>
</tr>
<tr>
<td>Example: 2 RET 3 + 4 *</td>
<td>'(2*3)+4' RET</td>
</tr>
<tr>
<td>Example: 2 RET 3 RET 4 + *</td>
<td>'2*(3+4)' RET</td>
</tr>
<tr>
<td>Example: 3 RET 6 + Q 3 -</td>
<td>'sqrt((36)/3)' RET</td>
</tr>
<tr>
<td>Example: P 3 / n S</td>
<td>'sin(pi/3)' RET</td>
</tr>
</tbody>
</table>

© 2022 Free Software Foundation, Inc. Permissions on back.

**Arithmetic**
- add, subtract, multiply, divide: +, -, *, /
- raise to a power, nth root: ^, I ^
- change sign: n
- reciprocal 1/x: k
- square root \( \sqrt{x} \): Q
- set precision: p
- round off last two digits: c 2
- convert to fraction, float: c F, c f
- enter using algebraic notation: >2+3*4
- refer to previous result: >3*8^2
- refer to higher stack entries: >$1*$2^2
- finish alg entry without evaluating: LFD
- set mode where alg entry used by default: m a

**Stack Commands**
Here \( S_n \) is the \( n \)th stack entry, and \( N \) is the size of the stack.

<table>
<thead>
<tr>
<th>key</th>
<th>no prefix</th>
<th>prefix ( n )</th>
<th>prefix ( -n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>RET</td>
<td>copy ( S_1 )</td>
<td>copy ( S_1 ) . . . copy ( S_n )</td>
<td></td>
</tr>
<tr>
<td>LFD</td>
<td>copy ( S_2 )</td>
<td>copy ( S_2 ) . . . copy ( S_1 ) . . . copy ( S_n )</td>
<td></td>
</tr>
<tr>
<td>DEL</td>
<td>delete ( S_1 )</td>
<td>delete ( S_1 ) . . . delete ( S_n )</td>
<td></td>
</tr>
<tr>
<td>M-DEL</td>
<td>delete ( S_2 )</td>
<td>delete ( S_2 ) . . . delete ( S_1 ) . . . delete ( S_n )</td>
<td></td>
</tr>
<tr>
<td>TAB</td>
<td>swap ( S_1 ) . . . ( S_2 )</td>
<td>roll ( S_1 ) to ( S_n )</td>
<td>roll ( S_n ) to ( S_1 )</td>
</tr>
<tr>
<td>M-TAB</td>
<td>roll ( S_1 ) to ( S_2 )</td>
<td>roll ( S_1 ) to ( S_n )</td>
<td>roll ( S_n ) to ( S_1 )</td>
</tr>
</tbody>
</table>

With a 0 prefix, these copy, delete, or reverse the entire stack.

**Display**
- scroll horizontally, vertically: <, >, { }
- home cursor: o
- line numbers on/off: d l
- trail display on/off: t d
- scientific notation: d s
- fixed-point notation: d f
- floating-point (normal) notation: d n
- group digits with commas: d g

For display mode commands, h prefix prevents screen redraw and I prefix temporarily redraws top of stack.

**Notations**
- scientific notation: 6.02e23
- minus sign in numeric entry: _23 or 23 n
- fractions: 3/4
- complex numbers: (x, y)
- polar complex numbers: (r; \( \theta \))
- vectors (commas optional): [1, 2, 3]
- matrices (or nested vectors): [1, 2; 3, 4]
- error forms (p key): 100 \( \pm^0.5 \)
- interval forms: [2 . . 5]
- modulo forms (M key): 6 mod 24
- HMS forms: 50 30' 0"
- date forms: <Jul 4, 1992>
- infinity, indeterminate

**Scientific Functions**
- ln, log_{10}, log_b: \( L, H, L, B \)
- exponential \( e^x, 10^x \): \( E, H, E \)
- sin, cos, tan: \( S, C, T \)
- arcsin, arccos, arctan: \( I S, I C, I T \)
- inverse, hyperbolic prefix keys: \( I, H \)
- two-argument arctan: f T
- degrees, radians modes: m d, m r
- pi (\( \pi \)): \( P \)
- factorial, double factorial: !, \( k \d \)
- combinations, permutations: \( k c, H k c \)
- prime factorization: k f
- next prime, previous prime: \( k n, I k n \)
- GCD, LCM: \( k g, k l \)
- random number, shuffle: \( k r, k h \)
- minimum, maximum: \( f n, f x \)
- error functions erf, erfc: \( f e, f l e \)
- gamma, beta functions: \( f g, f b \)
- incomplete gamma, beta functions: \( f g, f B \)
- Bessel \( J_\nu, Y_\nu \) functions: \( f j, f y \)
- complex magnitude, arg, conjugate: \( A, G, J \)
- real, imaginary parts: \( f r, f i \)
- convert polar/rectangular: \( c p \)

**Financial Functions**
- enter percentage: M-%
- convert to percentage: c %
- percentage change: b %
- present value: b P
- future value: b F
- rate of return: b T
- number of payments: b M
- size of payments: b H
- net present value, int. rate of return: \( b N, b I \)

Above computations assume payments at end of period. Use I prefix for beginning of period, or H for a lump sum investment.

**Units**
- straight-line depreciation: \( b S \)
- sum-of-years'-digits: \( b V \)
- double declining balance: \( b D \)
- enter with units: \( ' 65 \text{ mi/hr} \)
- convert to new units, base units: \( u c, u b \)
- convert temperature units: \( u t \)
- simplify units expression: \( u s \)
- view units table: \( u v \)

Common units:
- distance: \( m, \text{cm, mm, km}; \text{in, ft, mi, mfi}; \text{point, lyr} \)
- volume: \( \text{1 or L, ml, gal, pt, cup, floz, tsp, tasp} \)
- mass: \( m, \text{mg, kg, t}; \text{lb, oz, ton} \)
- time: \( \text{s or sec, ms, us, ns, min, hr, day, wk} \)
- temperature: \( \text{degC, degF, K} \)
GNU Calc Reference Card

Programmer's Functions

binary, octal, hex display
d 2, d 8, d 6
decimal, other radix display
d 0, d r
display leading zeros
d z
entering non-decimal numbers
16#7FFF
binary word size
b w
binary AND, OR, XOR
b a, b o, b x
binary DIFF, NOT
b d, b n
left shift
b l
logical right shift
b r
arithmetic right shift
b R
integer quotient, remainder
I Q
floor, ceiling, round to integer

Variables

Variable names are single digits or whole words.
store to variable
s t
store and keep on stack
s s
recall from variable
s r
shorthands for digit variables
n t, n s, r n
unstore, exchange variable
s u, s x
edit variable
s e

Vector Operations

vector of 1, 2, . . . , n
v v n
vector of n counts from a by b
v C-u v x
vector of copies of a value
v b
concatenate into vector
v l
pack many stack items into vector
v p
unpack vector or object
v u
length of vector (list)
v l
reverse vector
v v
sort, grade vector
v V S, V G
histogram of vector data
v V H
extract vector element
v x
matrix determinant, inverse
v D, k
matrix transpose, trace
v t, V T
cross, dot products
v C, *
identity matrix
v i
extract matrix row, column
v r, v c
intersection, union, diff of sets
v v - , V V, V -
cardinality of set
v #
add vectors elementwise (i.e., map +)
add elements in vector (i.e., reduce +)
sum rows in matrix
sum columns in matrix
sum elements, accumulate results

Algebra

enter an algebraic formula
' 2x+3y^2
enter an equation
' 2x^2+18
symbolic (vs. numeric) mode
m s
fractions (vs. float) mode
m f
suppress evaluation of formulas
m O
return to default evaluation rules
m D
"Big" display mode
m B
C, Pascal, FORTRAN modes
m C, m D, m P, m F
TyX, LaTeX, eqn modes
m T, m D, m L, m E
Maxima
m X
Unformatted mode
m U
Normal language mode
m N
simplify formula
a s
put formula into rational form
a n
evaluate variables in formula
= evaluate numerically
s l x=eval
declare properties of variable
s d
Common decks: pos, int, real, scalar, [a..b].
expand, collect terms
a x, a c
factor, partial fractions
a z, a a
polynomial quotient, remainder, GCD
a \, a ., a g
derivative, integral
a d, a i
taylor series
a t
principal solution to equation(s)
a S
list of solutions
a P
generic solution
H a S
apply function to both sides of eqn
a M
rewrite formula
a r
Example: a r a*b + a*c := a*(b+c)
Example: a r sin(x)^2 := 1-cos(x)^2
Example: a r cos(n*pi) := 1 :: integer(n) :: n%2 = 0
Example: a r f(a) := 1, f(a) := n if(a<1) :: a > 0
Put rules in EvalRules to have them apply automatically.
Put rules in AlgSimpRules to apply during a a command.
Common markers: opt, plain, quote, eval, let, remember.

Numerical Computations

sum formula over a range
a +
product of formula over a range
a *
tabulate formula over a range
a T
integrate numerically over a range
a I
find zero of formula or equation
a R
find local min, max of formula
a N, a X
fit data to line or curve
a F
mean of data in vector or variable
u M
median of data
u M
geometric mean of data
u G
sum, product of data
u +, u *
minimum, maximum of data
u N, u X
sample, pop. standard deviation
u S, I u S

Selections

select subformula under cursor
j s
select nth subformula
j n
select more
j m
unselect this, all formulas
j u, j c
copy indicated subformula
j RET
delete indicated subformula
j DEL
commute selected terms
j C
commute term leftward, rightward
j L, j R
distribute, merge selection
j D, j M
isolate selected term in equation
j I
negate, invert term in context
j N, j &
rewrite selected term
j r

Graphics

graph function or data
g f
graph 3D function or data
g F
replot current graph
g p
add curve to graph
g a
set number of data points
s n
g set line, point styles
s g
set log vs. linear
s g
set range for x, y axis
s g
close graphics window
s q

Programming

begin, end recording a macro
C-x (, C-x )
replay keyboard macro
X
read region as written-out macro
C-x * m
if, else, endif
Z [ , Z ; , Z ]
equal to, less than, member of
a <, a < a, a {
repeat n times, break from loop
Z <, Z >, Z /
"for" loop: start, end; body, step
Z =, Z ,
save, restore mode settings
Z <, Z >,
query user during macro
Z K
put finished macro on a key
Z K
define function with formula
Z F
edit definition
Z E
record user-defined command permanently
Z P
record variable value permanently
s p
record mode settings permanently
m m

Copyright © 2022 Free Software Foundation, Inc.
designed by Dave Gilmore and Stephen Gildea,
for GNU Emacs Calc.

Released under the terms of the GNU General Public License version 3
or later.
For more Emacs documentation, and the TeX source for this card, see
the Emacs distribution, or https://www.gnu.org/software/emacs