

Pari-GP reference card

(PARI-GP version 2.15.3)

Note: optional arguments are surrounded by braces {}.

To start the calculator, type its name in the terminal: **gp**

To exit **gp**, type **quit**, **\q**, or **<C-D>** at prompt.

Help

describe function	?function
extended description	??keyword
list of relevant help topics	???pattern
name of GP-1.39 function f in GP-2.*	whatnow(f)

Input/Output

previous result, the result before	%, %`, %`` , etc.
n -th result since startup	% n
separate multiple statements on line	;
extend statement on additional lines	\
extend statements on several lines	{ seq_1 ; seq_2 ;}
comment	/* ... */
one-line comment, rest of line ignored	\\ ...

Metacommands & Defaults

set default d to val	default({ d },{ val })
toggle timer on/off	#
print time for last result	##
print defaults	\d
set debug level to n	\g n
set memory debug level to n	\gm n
set n significant digits / bits	\p n , \pb n
set n terms in series	\ps n
quit GP	\q
print the list of PARI types	\t
print the list of user-defined functions	\u
read file into GP	\r $filename$
set debuglevel for domain D to n	setdebug(D,n)

Debugger / break loop

get out of break loop	break or <C-D>
go up/down n frames	dbg_up({ n }), dbg_down
set break point	breakpoint()
examine object o	dbg_x(o)
current error data	dbg_err()
number of objects on heap and their size	getheap()
total size of objects on PARI stack	getstack()

PARI Types & Input Formats

t_INT . Integers; hex, binary	± 31 ; $\pm 0x1F$, $\pm 0b101$
t_REAL . Reals	± 3.14 , 6.022 E23
t_INTMOD . Integers modulo m	Mod(n,m)
t_FRAC . Rational Numbers	n/m
t_FFELT . Elt in finite field \mathbf{F}_q	ffgen(q , 't)
t_COMPLEX . Complex Numbers	$x + y * I$
t_PADIC . p -adic Numbers	$x + O(p^k)$
t_QUAD . Quadratic Numbers	$x + y * \text{quadgen}(D, \{ 'w \})$
t_POLMOD . Polynomials modulo g	Mod(f,g)
t_POL . Polynomials	$a * x^n + \dots + b$
t_SER . Power Series	$f + O(x^k)$
t_RFRAC . Rational Functions	f/g
t_QFB . Binary quadratic form	Qfb(a,b,c)
t_VEC/t_COL . Row/Column Vectors	[x,y,z], [x,y,z]~
t_VEC integer range	[1..10]

t_VECSMALL . Vector of small ints	Vecsmall([x,y,z])
t_MAT . Matrices	[$a,b;c,d$]
t_LIST . Lists	List([x,y,z])
t_STR . Strings	"abc"
t_INFINITY . $\pm\infty$	+oo, -oo

Reserved Variable Names

$\pi \approx 3.14$, $\gamma \approx 0.57$, $C \approx 0.91$, $I = \sqrt{-1}$	Pi, Euler, Catalan, I
Landau's big-oh notation	O

Information about an Object, Precision

PARI type of object x	type(x)
length of x / size of x in memory	# x , sizebyte(x)
real precision / bit precision of x	precision(x), bitprecision(x)
p -adic, series prec. of x	padicprec(x,p), serprec(x,v)
current dynamic precision	getlocalprec, getlocalbitprec

Operators

basic operations	+, -, *, /, ^, sqr
$i \leftarrow i+1$, $i \leftarrow i-1$, $i \leftarrow i*j$, ...	i++, i--, i*=j,...
Euclidean quotient, remainder	$x \backslash y$, $x \backslash y$, $x \% y$, divrem(x,y)
shift x left or right n bits	$x << n$, $x >> n$ or shift($x, \pm n$)
multiply by 2^n	shiftmul(x,n)
comparison operators	<=, <, >=, >, ==, !=, ==, lex, cmp
boolean operators (or, and, not)	, &&, !
bit operations	bitand, bitneg, bitor, bitxor, bitnegimply
maximum/minimum of x and y	max(x,y), min(x,y)
sign of x (gives $-1, 0, 1$)	sign(x)
binary exponent of x	exponent(x)
derivative of f , 2nd derivative, etc.	f' , f'' , ...
differential operator	diffop($f,v,d,\{n=1\}$)
quote operator (formal variable)	'x
assignment	x = value
simultaneous assignment $x \leftarrow v[1]$, $y \leftarrow v[2]$	[x,y] = v

Select Components

Caveat: components start at index $n = 1$.

n -th component of x	component(x,n)
n -th component of vector/list x	$x[n]$
components $a, a+1, \dots, b$ of vector x	$x[a..b]$
(m,n) -th component of matrix x	$x[m,n]$
row m or column n of matrix x	$x[m,]$, $x[,n]$
numerator/denominator of x	numerator(x), denominator(x)

Random Numbers

random integer/prime in $[0,N[$	random(N), randomprime(N)
get/set random seed	getrand, setrand(s)

Conversions

to vector, matrix, vec. of small ints	Col/Vec, Mat, Vecsmall
to list, set, map, string	List, Set, Map, Str
create $(x \bmod y)$	Mod(x,y)
make x a polynomial of v	Pol($x,\{v\}$)
variants of Pol <i>et al.</i> , in reverse order	Polrev, Vecrev, Colrev
make x a power series of v	Ser($x,\{v\}$)
convert x to simplest possible type	simplify(x)
object x with real precision n	precision(x,n)
object x with bit precision n	bitprecision(x,n)
set precision to p digits in dynamic scope	localprec(p)
set precision to p bits in dynamic scope	localbitprec(p)

Character strings

convert to TeX representation	strtex(x)
string from bytes / from format+args	strchr, sprintf
split string / join strings	strsplit, strjoin
convert time t ms. to h, m, s, ms format	strtime(t)
Conjugates and Lifts	
conjugate of a number x	conj(x)
norm of x , product with conjugate	norm(x)
L^p norm of x (L^∞ if no p)	normlp($x,\{p\}$)
square of L^2 norm of x	norml2(x)
lift of x from Mods and p -adics	lift, centerlift(x)
recursive lift	liftall
lift all t_INT and t_PADIC (\rightarrow t_INT)	liftint
lift all t_POLMOD (\rightarrow t_POL)	liftpol

Lists, Sets & Maps

Sets (= row vector with strictly increasing entries w.r.t. cmp)	
intersection of sets x and y	setintersect(x,y)
set of elements in x not belonging to y	setminus(x,y)
symmetric difference $x \Delta y$	setdelta(x,y)
union of sets x and y	setunion(x,y)
does y belong to the set x	setsearch($x,y,\{flag\}$)
set of all $f(x,y)$, $x \in X$, $y \in Y$	setbinop(f,X,Y)
is x a set ?	setisset(x)

Lists. create empty list: $L = \text{List}()$

append x to list L	listput($L,x,\{i\}$)
remove i -th component from list L	listpop($L,\{i\}$)
insert x in list L at position i	listinsert(L,x,i)
sort the list L in place	listsort($L,\{flag\}$)

Maps. create empty dictionary: $M = \text{Map}()$

attach value v to key k	mapput(M,k,v)
recover value attach to key k or error	mapget(M,k)
is key k in the dict? (set v to $M(k)$)	mapisdefined($M,k,\{\&v\}$)
remove k from map domain	mapdelete(M,k)

GP Programming

User functions and closures

x,y are formal parameters; y defaults to Pi if parameter omitted; z,t are local variables (lexical scope), z initialized to 1.

fun(x, y=Pi) = my(z=1, t); seq

fun = (x, y=Pi) -> my(z=1, t); seq

attach help message h to s	addhelp(s,h)
undefine symbol s (also kills help)	kill(s)
Control Statements (X : formal parameter in expression seq)	
if $a \neq 0$, evaluate seq_1 , else seq_2	if($a,\{seq_1\},\{seq_2\}$)
eval. seq for $a \leq X \leq b$	for($X = a,b,seq$)
...for $X \in v$	foreach(v,X,seq)
...for primes $a \leq X \leq b$	forprime($X = a,b,seq$)
...for primes $\equiv a \pmod q$	forprimestep($X = a,b,q,seq$)
...for composites $a \leq X \leq b$	forcomposite($X = a,b,seq$)
...for $a \leq X \leq b$ stepping s	forstep($X = a,b,s,seq$)
...for X dividing n	fordiv(n,X,seq)
... $X = [n, factor(n)]$, $a \leq n \leq b$	forfactored($X = a,b,seq$)
...as above, n squarefree	forsquarefree($X = a,b,seq$)
... $X = [d, factor(d)]$, $d \mid n$	fordivfactored(n,X,seq)
multivariable for, lex ordering	forvec($X = v,seq$)

```

loop over partitions of  $n$ 
... permutations of  $S$ 
... subsets of  $\{1, \dots, n\}$ 
...  $k$ -subsets of  $\{1, \dots, n\}$ 
... vectors  $v, q(v) \leq B$ ;  $q > 0$ 
...  $H < G$  finite abelian group
evaluate  $seq$  until  $a \neq 0$ 
while  $a \neq 0$ , evaluate  $seq$ 
exit  $n$  innermost enclosing loops
start new iteration of  $n$ -th enclosing loop
return  $x$  from current subroutine
Exceptions, warnings
raise an exception / warning
type of error message  $E$ 
try  $seq_1$ , evaluate  $seq_2$  on error
Functions with closure arguments / results
number of arguments of  $f$ 
select from  $v$  according to  $f$ 
apply  $f$  to all entries in  $v$ 
evaluate  $f(a_1, \dots, a_n)$ 
evaluate  $f(\dots f(f(a_1, a_2), a_3) \dots, a_n)$ 
calling function as closure
Sums & Products
sum  $X = a$  to  $X = b$ , initialized at  $x$ 
sum entries of vector  $v$ 
product of all vector entries
sum  $expr$  over divisors of  $n$ 
... assuming  $expr$  multiplicative
product  $a \leq X \leq b$ , initialized at  $x$ 
product over primes  $a \leq X \leq b$ 
Sorting
sort  $x$  by  $k$ -th component
min.  $m$  of  $x$  ( $m = x[i]$ ), max.
does  $y$  belong to  $x$ , sorted wrt.  $f$ 
 $\prod g^x \rightarrow$  factorization ( $\Rightarrow$  sorted, unique  $g$ )
Input/Output
print with/without  $\backslash n$ ,  $\text{\TeX}$  format
pretty print matrix
print fields with separator
formatted printing
write  $args$  to file
write  $x$  in binary format
read file into GP
... return as vector of lines
... return as vector of strings
read a string from keyboard
Files and file descriptors
File descriptors allow efficient small consecutive reads or writes
from or to a given file. The argument  $n$  below is always a descriptor,
attached to a file in r(ead), w(rite) or a(ppend) mode.
get descriptor  $n$  for file  $path$  in given  $mode$ 
... from shell  $cmd$  output (pipe)
close descriptor
commit pending write operations
read logical line from file
... raw line from file
write  $s \backslash n$  to file
... write  $s$  to file

```

```

forpart( $p = n, seq$ )
forperm( $S, p, seq$ )
forsubset( $n, p, seq$ )
forsubset( $[n, k], p, seq$ )
forqfvec( $v, q, b, seq$ )
forsubgroup( $H = G$ )
until( $a, seq$ )
while( $a, seq$ )
break( $\{n\}$ )
next( $\{n\}$ )
return( $\{x\}$ )

error(), warning()
errname( $E$ )
iferr( $seq_1, E, seq_2$ )

Results
arity( $f$ )
select( $f, v$ )
apply( $f, v$ )
call( $f, a$ )
fold( $f, a$ )
self()

sum( $X = a, b, expr, \{x\}$ )
vecsum( $v$ )
vecprod( $v$ )
sumdiv( $n, X, expr$ )
sumdivmult( $n, X, expr$ )
prod( $X = a, b, expr, \{x\}$ )
prodeuler( $X = a, b, expr$ )

vecsort( $x, \{k\}, \{fl = 0\}$ )
vecmin( $x, \{\&i\}$ ), vecmax
vecsearch( $x, y, \{f\}$ )
matreduce( $m$ )

print, print1, printtex
printp
printsep( $sep, \dots$ ), printsep1
printf()

write, write1, writetex( $file, args$ )
writebin( $file, x$ )
read( $\{file\}$ )
readvec( $\{file\}$ )
readstr( $\{file\}$ )
input()

fileclose( $n$ )
fileflush( $n$ )
fileread( $n$ )
filereadstr( $n$ )
filewrite( $n, s$ )
filewrite1( $n, s$ )

```

Pari-GP reference card

(PARI-GP version 2.15.3)

Timers

CPU time in ms and reset timer
CPU time in ms since gp startup
time in ms since UNIX Epoch
timeout command after s seconds

Interface with system

allocates a new stack of s bytes
alias old to new
install function from library
execute system command a
... and feed result to GP
... returning GP string
get $\$VAR$ from environment
expand env. variable in string

```

gettime()
getabstime()
getwalltime()
alarm( $s, expr$ )

allocatemem( $\{s\}$ )
alias( $new, old$ )
install( $f, code, \{gpf\}, \{lib\}$ )
system( $a$ )
extern( $a$ )
externstr( $a$ )
getenv("VAR")
strexpand( $x$ )

```

Parallel evaluation

These functions evaluate their arguments in parallel (pthreads or MPI); args. must not access global variables (use **export** for this) and must be free of side effects. Enabled if threading engine is not *single* in gp header.

```

evaluate  $f$  on  $x[1], \dots, x[n]$ 
evaluate closures  $f[1], \dots, f[n]$ 
as select
as sum
as vector
eval  $f$  for  $i = a, \dots, b$ 
... for each element  $x$  in  $v$ 
... for  $p$  prime in  $[a, b]$ 
... for  $p = a \bmod q$ 
... multivariate
parforvec( $X = v, f, \{r\}, \{f_2\}, \{flag\}$ )
export  $x$  to parallel world
... all dynamic variables
frees exported value  $x$ 
... all exported values

```

```

parapply( $f, x$ )
pareval( $f$ )
parselect( $f, A, \{flag\}$ )
parsum( $i = a, b, expr$ )
parvector( $n, i, \{expr\}$ )
parfor( $i = a, \{b\}, f, \{r\}, \{f_2\}$ )
parforeach( $v, x, f, \{r\}, \{f_2\}$ )
parforprime( $p = a, \{b\}, f, \{r\}, \{f_2\}$ )
parforprimestep( $p = a, \{b\}, q, f, \{r\}, \{f_2\}$ )

```

Linear Algebra

dimensions of matrix x
multiply two matrices
... assuming result is diagonal
concatenation of x and y
extract components of x
transpose of vector or matrix x
adjoint of the matrix x
eigenvectors/values of matrix x
characteristic/minimal polynomial of x
trace/determinant of matrix x
permanent of matrix x
Frobenius form of x
QR decomposition
apply **matqr**'s transform to v

```

matsize( $x$ )
 $x * y$ 
matmultodiagonal( $x, y$ )
concat( $x, \{y\}$ )
vecextract( $x, y, \{z\}$ )
 $x \sim$ , mattranspose( $x$ )
matadjoint( $x$ )
mateigen( $x$ )
charpoly( $x$ ), minpoly( $x$ )
trace( $x$ ), matdet( $x$ )
matpermanent( $x$ )
matfrobenius( $x$ )
matqr( $x$ )
mathouseholder( $Q, v$ )

```

Constructors & Special Matrices

```

{ $g(x): x \in v$  s.t.  $f(x)$ }
{ $x: x \in v$  s.t.  $f(x)$ }
{ $g(x): x \in v$ }
row vec. of  $expr$  eval'ed at  $1 \leq i \leq n$ 
col. vec. of  $expr$  eval'ed at  $1 \leq i \leq n$ 
vector of small ints

```

```

[g(x) | x <- v, f(x)]
[x | x <- v, f(x)]
[g(x) | x <- v]
vector( $n, \{i\}, \{expr\}$ )
vectorv( $n, \{i\}, \{expr\}$ )
vectorsmall( $n, \{i\}, \{expr\}$ )

```

```

[ $c, c \cdot x, \dots, c \cdot x^n$ ]
[ $1, 2^x, \dots, n^x$ ]
matrix  $1 \leq i \leq m, 1 \leq j \leq n$ 
define matrix by blocks
diagonal matrix with diagonal  $x$ 
is  $x$  diagonal?
 $x \cdot \text{matdiagonal}(d)$ 
 $n \times n$  identity matrix
Hessenberg form of square matrix  $x$ 
 $n \times n$  Hilbert matrix  $H_{ij} = (i + j - 1)^{-1}$ 
 $n \times n$  Pascal triangle
companion matrix to polynomial  $x$ 
Sylvester matrix of  $x$  and  $y$ 
Gaussian elimination
kernel of matrix  $x$ 
intersection of column spaces of  $x$  and  $y$ 
solve  $MX = B$  ( $M$  invertible)
one sol of  $M * X = B$ 
basis for image of matrix  $x$ 
columns of  $x$  not in matimage
supplement columns of  $x$  to get basis
rows, cols to extract invertible matrix
rank of the matrix  $x$ 
solve  $MX = B \bmod D$ 
image mod  $D$ 
kernel mod  $D$ 
inverse mod  $D$ 
determinant mod  $D$ 

```

```

powers( $x, n, \{c = 1\}$ )
dirpowers( $n, x$ )
matrix( $m, n, \{i\}, \{j\}, \{expr\}$ )
matconcat( $B$ )
matdiagonal( $x$ )
matisdiagonal( $x$ )
matmuldiagonal( $x, d$ )
matid( $n$ )
mathess( $x$ )
mathilbert( $n$ )
matpascal( $n - 1$ )
matcompanion( $x$ )
polsylvestermatrix( $x, y$ )

matker( $x, \{flag\}$ )
matintersect( $x, y$ )
matsolve( $M, B$ )
matinverseimage( $M, B$ )
matimage( $x$ )
matimagecompl( $x$ )
matsupplement( $x$ )
matindexrank( $x$ )
matrank( $x$ )
matsolvemod( $M, D, B$ )
matimagemod( $M, D$ )
matkermod( $M, D$ )
matinvmod( $M, D$ )
matdetmod( $M, D$ )

```

Lattices & Quadratic Forms

Quadratic forms

```

evaluate  ${}^t x Q y$ 
evaluate  ${}^t x Q x$ 
signature of quad form  ${}^t y * x * y$ 
decomp into squares of  ${}^t y * x * y$ 
eigenvalues/vectors for real symmetric  $x$ 

```

```

qfeval( $\{Q = id\}, x, y$ )
qfeval( $\{Q = id\}, x$ )
qfsign( $x$ )
qfgaussred( $x$ )
qfjacobi( $x$ )

```

HNF and SNF

```

upper triangular Hermite Normal Form
HNF of  $x$  where  $d$  is a multiple of  $\det(x)$ 
multiple of  $\det(x)$ 
HNF of  $(x \mid \text{diagonal}(D))$ 
elementary divisors of  $x$ 
 $q$ -rank from elementary divisors
elementary divisors of  $\mathbf{Z}[a]/(f'(a))$ 
integer kernel of  $x$ 

```

Z-module \leftrightarrow **Q**-vector space

Lattices

```

LLL-algorithm applied to columns of  $x$ 
... for Gram matrix of lattice
find up to  $m$  sols of qfnorm( $x, y$ )  $\leq b$ 
 $v, v[i] :=$  number of  $y$  s.t. qfnorm( $x, y$ ) =  $i$ 
perfection rank of  $x$ 
find isomorphism between  $q$  and  $Q$ 
precompute for isomorphism test with  $q$ 
automorphism group of  $q$ 

```

```

mathnf( $x$ )
mathnfmod( $x, d$ )
matdetint( $x$ )
mathnfmodid( $x, D$ )
matsnf( $x$ )
snfrank( $v, q$ )
poldiscreduced( $f$ )
materint( $x$ )
matrixqz( $x, p$ )

qflll( $x, \{flag\}$ )
qflllgram( $x, \{flag\}$ )
qfminim( $x, b, m$ )
qfrep( $x, B, \{flag\}$ )
qfperfection( $x$ )
qfisom( $q, Q$ )
qfisominit( $q$ )
qfauto( $q$ )

```

convert `qfauto` for GAP/Magma `qfautoexport($G, \{flag\}$)`
orbits of V under $G \subset \mathrm{GL}(V)$ `qforbits(G, V)`

Polynomials & Rational Functions

all defined polynomial variables `variables()`
get var. of highest priority (higher than v) `varhigher($name, \{v\}$)`
... of lowest priority (lower than v) `varlower($name, \{v\}$)`

Coefficients, variables and basic operators

degree of f `poldegree(f)`
coef. of degree n of f , leading coef. `polcoef(f, n), pollead`
main variable / all variables in f `variable(f), variables(f)`
replace x by y in f `subst(f, x, y)`
evaluate f replacing vars by their value `eval(f)`
replace polynomial expr. $T(x)$ by y in f `substpol(f, T, y)`
replace x_1, \dots, x_n by y_1, \dots, y_n in f `substvec(f, x, y)`

$f \in A[x]$; reciprocal polynomial $x^{\deg f} f\left(\frac{1}{x}\right)$ `polrecip(f)`
gcd of coefficients of f `content(f)`
derivative of f w.r.t. x `deriv($f, \{x\}$)`
... n -th derivative of f `derivn($f, n, \{x\}$)`
formal integral of f w.r.t. x `intformal($f, \{x\}$)`
formal sum of f w.r.t. x `sumformal($f, \{x\}$)`

Constructors & Special Polynomials

interpolation polynomial at $(x[1], y[1]), \dots, (x[n], y[n])$, evaluated at t , with error estimate e `polinterpolate($x, \{y\}, \{t\}, \{&e\}$)`
 $T_n/U_n, H_n$ `polchebyshev(n), polhermite(n)`
 $P_n, L_n^{(\alpha)}$ `pollegendre(n), pollaguerre(n, a)`
 n -th cyclotomic polynomial Φ_n `polcyclo(n)`
return n if $f = \Phi_n$, else 0 `poliscyclo(f)`
is f a product of cyclotomic polynomials? `poliscycloprod(f)`
Zagier's polynomial of index (n, m) `polzagier(n, m)`

Resultant, elimination

discriminant of polynomial f `poldisc(f)`
find factors of `poldisc(f)` `poldiscfactors(f)`
resultant $R = \mathrm{Res}_v(f, g)$ `polresultant($f, g, \{v\}$)`
 $[u, v, R], xu + yv = \mathrm{Res}_v(f, g)$ `polresultantext($x, y, \{v\}$)`
solve Thue equation $f(x, y) = a$ `thue($t, a, \{sol\}$)`
initialize t for Thue equation solver `thueinit(f)`

Roots and Factorization (Complex/Real)

complex roots of f `polroots(f)`
bound complex roots of f `polrootsbound(f)`
number of real roots of f (in $[a, b]$) `polsturm($f, \{[a, b]\}$)`
real roots of f (in $[a, b]$) `polrootsreal($f, \{[a, b]\}$)`
complex embeddings of `t_POLMOD` z `conjvec(z)`

Roots and Factorization (Finite fields)

factor f mod p , roots `factormod(f, p), polrootsmod`
factor f over $\mathbf{F}_p[x]/(T)$, roots `factormod($f, [T, p]$), polrootsmod`
squarefree factorization of f in $\mathbf{F}_q[x]$ `factormodSQF($f, \{D\}$)`
distinct degree factorization of f in $\mathbf{F}_q[x]$ `factormodDDF($f, \{D\}$)`
factor n -th cyclotomic pol. Φ_n mod p `factormodcyclo(n, p)`

Roots and Factorization (p -adic fields)

factor f over \mathbf{Q}_p , roots `factorpadic(f, p, r), polrootspadic`
 p -adic root of f congruent to a mod p `padicappr(f, a)`
Newton polygon of f for prime p `newtonpoly(f, p)`
Hensel lift $A/\mathrm{lc}(A) = \prod_i B[i]$ mod p^e `polhensellift(A, B, p, e)`
 $T = \prod (x - z_i) \mapsto \prod [x - \omega(z_i)] \in \mathbf{Z}_p[x]$ `polteichmuller(T, p, e)`
extensions of \mathbf{Q}_p of degree N `padicfields(p, N)`

Pari-GP reference card

(PARI-GP version 2.15.3)

Roots and Factorization (Miscellaneous)

symmetric powers of roots of f up to n `polsym(f, n)`
Graeffe transform of $f, g(x^2) = f(x)f(-x)$ `polgraeffe(f)`
factor f over coefficient field `factor(f)`
cyclotomic factors of $f \in \mathbf{Q}[X]$ `polcyclofactors(f)`

Finite Fields

A finite field is encoded by any element (`t_FFELT`).
find irreducible $T \in \mathbf{F}_p[x]$, $\deg T = n$ `ffinit($p, n, \{x\}$)`
Create t in $\mathbf{F}_q \simeq \mathbf{F}_p[t]/(T)$ `t = ffgen($T, 't$)`
... indirectly, with implicit T `t = ffgen($q, 't$); T = t.mod`
map m from $\mathbf{F}_q \ni a$ to $\mathbf{F}_{q^k} \ni b$ `m = ffbend(a, b)`
build $K = \mathbf{F}_q[x]/(P)$ extending $\mathbf{F}_q \ni a$, `ffextend(a, P)`
evaluate map m on x `ffmap(m, x)`
inverse map of m `ffinvmap(m)`
compose maps $m \circ n$ `ffcompomap(m, n)`
 x as polmod over codomain of map m `ffmaprel(m, x)`
 F^n over $\mathbf{F}_q \ni a$ `fffrobenius(a, n)`
 $\#\{\text{monic irred. } T \in \mathbf{F}_q[x], \deg T = n\}$ `ffnbirred(q, n)`

Formal & p -adic Series

truncate power series or p -adic number `truncate(x)`
valuation of x at p `valuation(x, p)`
Dirichlet and Power Series
Taylor expansion around 0 of f w.r.t. x `taylor(f, x)`
Laurent series of closure F up to x^k `laurentseries(f, k)`
 $\sum a_k b_k t^k$ from $\sum a_k t^k$ and $\sum b_k t^k$ `serconvol(a, b)`
 $f = \sum a_k t^k$ from $\sum (a_k/k!) t^k$ `serlaplace(f)`
reverse power series F so $F(f(x)) = x$ `serreverse(f)`
remove terms of degree $< n$ in f `serchop(f, n)`
Dirichlet series multiplication / division `dirmul, dirdiv(x, y)`
Dirichlet Euler product (b terms) `direuler($p = a, b, expr$)`

Transcendental and p -adic Functions

real, imaginary part of x `real(x), imag(x)`
absolute value, argument of x `abs(x), arg(x)`
square/nth root of x `sqrt(x), sqrtn($x, n, \{&z\}$)`
all n -th roots of 1 `rootsof1(n)`
FFT of $[f_0, \dots, f_{n-1}]$ `w = fftinit(n), fft/fftinw(w, f)`
trig functions `sin, cos, tan, cotan, sinc`
inverse trig functions `asin, acos, atan`
hyperbolic functions `sinh, cosh, tanh, cotanh`
inverse hyperbolic functions `asinh, acosh, atanh`
 $\log(x), \log(1+x), e^x, e^x - 1$ `log, loglp, exp, expm1`
Euler Γ function, $\log \Gamma, \Gamma'/\Gamma$ `gamma, lngamma, psi`
half-integer gamma function $\Gamma(n+1/2)$ `gammah(n)`
Riemann's zeta $\zeta(s) = \sum n^{-s}$ `zeta(s)`
 $\sum_{1 \leq n \leq N} n^s$ `dirpowerssum(N, s)`
Hurwitz's $\zeta(s, x) = \sum (n+x)^{-s}$ `zetahurwitz(s, x)`
Lerch $\Phi(z, s, x) = \sum z^n (n+x)^{-s}$ `lerchphi(z, s, x)`
Lerch $L(s, x, t) = \Phi(e^{2i\pi t}, s, x)$ `lerchzeta(s, x, t)`
multiple zeta value (MZV), $\zeta(s_1, \dots, s_k)$ `zetamult($s, \{T\}$)`
all MZVs for weight $\sum s_i = n$ `zetamultall(n)`
convert MZV id to $[s_1, \dots, s_k]$ `zetamultconvert($f, \{flag\}$)`
MZV dual sequence `zetamultdual(s)`
multiple polylog $Li_{s_1, \dots, s_k}(z_1, \dots, z_k)$ `polylogmult(s, z)`

incomplete Γ function ($y = \Gamma(s)$) `incgam($s, x, \{y\}$)`
complementary incomplete Γ `incgamc(s, x)`
 $\int_x^\infty e^{-t} dt/t, (2/\sqrt{\pi}) \int_x^\infty e^{-t^2} dt$ `eint1, erfc`
elliptic integral of 1st and 2nd kind `ellK(k), ellE(k)`
dilogarithm of x `dilog(x)`
 m -th polylogarithm of x `polylog($m, x, \{flag\}$)`
 U -confluent hypergeometric function `hyperu(a, b, u)`
Hypergeometric ${}_pF_q(A, B; z)$ `hypergeom(A, B, z)`
Bessel $J_n(x), J_{n+1/2}(x)$ `besselj(n, x), besseljh(n, x)`
Bessel $I_\nu, K_\nu, H_\nu^1, H_\nu^2, Y_\nu$ `(bessel)i, k, h1, h2, y`
 k -th zero of $J_\nu(x)$ `besseljzero($nu, \{k=1\}$)`
 k -th zero of $Y_\nu(x)$ `besselyzero($nu, \{k=1\}$)`
Airy functions $A_i(x), B_i(x)$ `airy(x)`
Lambert $W: x$ s.t. $xe^x = y$ `lambertw(y)`
Teichmuller character of p -adic x `teichmuller(x)`

Iterations, Sums & Products

Numerical integration for meromorphic functions

Behaviour at endpoint for Double Exponential (DE) methods: either a scalar ($a \in \mathbf{C}$, regular) or $\pm\infty$ (decreasing at least as x^{-2}) or
 $(x-a)^{-\alpha}$ singularity `[a, α]`
exponential decrease $e^{-\alpha|x|}$ `[$\pm\infty, \alpha$], $\alpha > 0$`
slow decrease $|x|^\alpha$ `... $\alpha < -1$`
oscillating as $\cos(kx)$ `$\alpha = k\mathbf{I}, k > 0$`
oscillating as $\sin(kx)$ `$\alpha = -k\mathbf{I}, k > 0$`

numerical integration `intnum($x = a, b, f, \{T\}$)`
weights T for intnum `intnuminit($a, b, \{m\}$)`
weights T incl. kernel K `intfuncinit($t = a, b, K, \{m\}$)`
integrate $(2i\pi)^{-1} f$ on circle $|z-a| = R$ `intcirc($x = a, R, f, \{T\}$)`
Other integration methods
 n -point Gauss-Legendre `intnumgauss($x = a, b, f, \{n\}$)`
weights for n -point Gauss-Legendre `intnumgaussinit($\{n\}$)`
quasi-periodic function, period $2H$ `intnumosc($x = a, f, H$)`
Romberg (low accuracy) `intnumromb($x = a, b, f, \{flag\}$)`

Numerical summation

sum of series $f(n), n \geq a$ (low accuracy) `suminf($n = a, expr$)`
sum of alternating/positive series `sumalt, sumpos`
sum of series using Euler-Maclaurin `sumnum($n = a, f, \{T\}$)`
... Sidi summation `sumnumsidi($n = a, f$)`
 $\sum_{n \geq a} F(n)$, F rational function `sumnumrat(F, a)`
... $\sum_{p \geq a} F(p^s)$ `sumeulerrat($F, \{s=1\}, \{a=2\}$)`
weights for sumnum, a as in DE `sumnuminit($\{\infty, a\}$)`
sum of series by Monien summation `sumnummonien($n = a, f, \{T\}$)`
weights for sumnummonien `sumnummonieninit($\{\infty, a\}$)`
sum of series using Abel-Plana `sumnumap($n = a, f, \{T\}$)`
weights for sumnumap, a as in DE `sumnumapinit($\{\infty, a\}$)`
sum of series using Lagrange `sumnumlagrange($n = a, f, \{T\}$)`
weights for sumnumlagrange `sumnumlagrangeinit`

Products

product $a \leq X \leq b$, initialized at x `prod($X = a, b, expr, \{x\}$)`
product over primes $a \leq X \leq b$ `prodeuler($X = a, b, expr$)`
infinite product $a \leq X \leq \infty$ `prodinf($X = a, expr$)`
 $\prod_{n \geq a} F(n)$, F rational function `prodnumrat(F, a)`
 $\prod_{p \geq a} F(p^s)$ `prodeulerrat($F, \{s=1\}, \{a=2\}$)`

Other numerical methods

real root of f in $[a, b]$; bracketed root	<code>solve($X = a, b, f$)</code>
...interval splitting, step s	<code>solvestep($X = a, b, s, f, \{flag = 0\}$)</code>
limit of $f(t)$, $t \rightarrow \infty$	<code>limitnum($f, \{\alpha\}$)</code>
asymptotic expansion of f (rational)	<code>asypnum($f, \{\alpha\}$)</code>
... $N + 1$ terms as floats	<code>asypnumraw($f, N, \{\alpha\}$)</code>
numerical derivation w.r.t x : $f'(a)$	<code>derivnum($x = a, f$)</code>
evaluate continued fraction F at t	<code>contfraceval($F, t, \{L\}$)</code>
power series to cont. fraction (L terms)	<code>contfracinit($S, \{L\}$)</code>
Padé approximant (deg. denom. $\leq B$)	<code>bestapprPade($S, \{B\}$)</code>

Elementary Arithmetic Functions

vector of binary digits of $ x $	<code>binary(x)</code>
bit number n of integer x	<code>bittest(x, n)</code>
Hamming weight of integer x	<code>hammingweight(x)</code>
digits of integer x in base B	<code>digits($x, \{B = 10\}$)</code>
sum of digits of integer x in base B	<code>sumdigits($x, \{B = 10\}$)</code>
integer from digits	<code>fromdigits($v, \{B = 10\}$)</code>
ceiling/floor/fractional part	<code>ceil, floor, frac</code>
round x to nearest integer	<code>round($x, \{\&e\}$)</code>
truncate x	<code>truncate($x, \{\&e\}$)</code>
gcd/LCM of x and y	<code>gcd(x, y), lcm(x, y)</code>
gcd of entries of a vector/matrix	<code>content(x)</code>

Primes and Factorization

extra prime table	<code>addprimes()</code>
add primes in v to prime table	<code>addprimes(v)</code>
remove primes from prime table	<code>removeprimes(v)</code>
Chebyshev $\pi(x)$, n -th prime p_n	<code>primepi(x), prime(n)</code>
vector of first n primes	<code>primes(n)</code>
smallest prime $\geq x$	<code>nextprime(x)</code>
largest prime $\leq x$	<code>precprime(x)</code>
factorization of x	<code>factor($x, \{lim\}$)</code>
...selecting specific algorithms	<code>factorint($x, \{flag = 0\}$)</code>
$n = df^2$, d squarefree/fundamental	<code>core($n, \{fl\}$), coredisc</code>
certificate for (prime) N	<code>primecert(N)</code>
verifies a certificate c	<code>primecertisvalid(c)</code>
convert certificate to Magma/PRIMO	<code>primecertexport</code>
recover x from its factorization	<code>factorback($f, \{e\}$)</code>
$x \in \mathbf{Z}$, $ x \leq X$, $\gcd(N, P(x)) \geq N$	<code>zncoppersmith($P, N, X, \{B\}$)</code>
divisors of N in residue class r mod s	<code>divisorslensstra(N, r, s)</code>

Divisors and multiplicative functions

number of prime divisors $\omega(n)$ / $\Omega(n)$	<code>omega(n), bigomega</code>
divisors of n / number of divisors $\tau(n)$	<code>divisors(n), numdiv</code>
sum of (k -th powers of) divisors of n	<code>sigma($n, \{k\}$)</code>
Möbius μ -function	<code>moebius(x)</code>
Ramanujan's τ -function	<code>ramanujantau(x)</code>

Combinatorics

factorial of x	<code>x!</code> or <code>factorial(x)</code>
binomial coefficient $\binom{x}{k}$	<code>binomial($x, \{k\}$)</code>
Bernoulli number B_n as real/rational	<code>bernreal(n), bernfrac</code>
$[B_0, B_2, \dots B_{2k}]$	<code>bernvec(k)</code>
Bernoulli polynomial $B_n(x)$	<code>bernpol($n, \{x\}$)</code>
Euler numbers	<code>eulerfrac, eulerreal, eulervec</code>
Euler polynomial $E_n(x)$	<code>eulerpol($n, \{x\}$)</code>
Eulerian polynomial $A_n(x)$	<code>eulerianpol</code>
Fibonacci number F_n	<code>fibonacci(n)</code>
Harmonic number $H_{n,r} = 1^{-r} + \dots + n^{-r}$	<code>harmonic(n, r)</code>
Stirling numbers $s(n, k)$ and $S(n, k)$	<code>stirling($n, k, \{flag\}$)</code>

Pari-GP reference card

(PARI-GP version 2.15.3)

number of partitions of n	<code>numbpart(n)</code>
k -th permutation on n letters	<code>numtoperm(n, k)</code>
...index k of permutation v	<code>permtomv(n)</code>
order of permutation p	<code>permorder(p)</code>
signature of permutation p	<code>permsign(p)</code>
cyclic decomposition of permutation p	<code>permcycles(p)</code>

Multiplicative groups $(\mathbf{Z}/N\mathbf{Z})^*$, \mathbf{F}_q^*

Euler ϕ -function	<code>eulerphi(x)</code>
multiplicative order of x (divides ϕ)	<code>znorder($x, \{o\}$), fforder</code>
primitive root mod q / x .mod	<code>znprimroot(q), fprimroot(x)</code>
structure of $(\mathbf{Z}/n\mathbf{Z})^*$	<code>znstar(n)</code>
discrete logarithm of x in base g	<code>znlog($x, g, \{o\}$), fflag</code>
Kronecker-Legendre symbol $(\frac{x}{y})$	<code>kronecker(x, y)</code>
quadratic Hilbert symbol (at p)	<code>hilbert($x, y, \{p\}$)</code>

Euclidean algorithm, continued fractions

CRT: solve $z \equiv x$ and $z \equiv y$	<code>chinese(x, y)</code>
minimal u, v so $xu + yv = \gcd(x, y)$	<code>gcdext(x, y)</code>
half-gcd algorithm	<code>halfgcd(x, y)</code>
continued fraction of x	<code>confrac($x, \{b\}, \{lmax\}$)</code>
last convergent of continued fraction x	<code>confracpnqn(x)</code>
rational approximation to x (den. $\leq B$)	<code>bestappr($x, \{B\}$)</code>
recognize $x \in \mathbf{C}$ as polmod mod $T \in \mathbf{Z}[X]$	<code>bestapprnf(x, T)</code>

Miscellaneous

integer square / n -th root of x	<code>sqrtint(x), sqtrtnint(x, n)</code>
largest integer e s.t. $b^e \leq b$, $e = \lfloor \log_b(x) \rfloor$	<code>logint($x, b, \{\&z\}$)</code>

Characters

Let $cyc = [d_1, \dots, d_k]$ represent an abelian group $G = \oplus (\mathbf{Z}/d_j\mathbf{Z}) \cdot g_j$ or any structure G affording a .cyc method; e.g. `znstar($q, 1$)` for Dirichlet characters. A character χ is coded by $[c_1, \dots, c_k]$ such that $\chi(g_j) = e(n_j/d_j)$.
 $\chi \cdot \psi$; χ^{-1} ; $\chi \cdot \psi^{-1}$; χ^k `charm, charconj, chardiv, charpow`
order of χ `charorder(cyc, χ)`
kernel of χ `charker(cyc, χ)`
 $\chi(x)$, G a GP group structure `chareval($G, \chi, x, \{z\}$)`
Galois orbits of characters `chargalois(G)`

Dirichlet Characters

initialize $G = (\mathbf{Z}/q\mathbf{Z})^*$	<code>G = znstar($q, 1$)</code>
convert datum D to $[G, \chi]$	<code>znchar(D)</code>
is χ odd?	<code>zncharisodd(G, χ)</code>
real $\chi \rightarrow$ Kronecker symbol $(D/.)$	<code>znchartokronecker(G, χ)</code>
conductor of χ	<code>zncharconductor(G, χ)</code>
$[G_0, \chi_0]$ primitive attached to χ	<code>znchartoprimitive(G, χ)</code>
induce $\chi \in \hat{G}$ to $\mathbf{Z}/N\mathbf{Z}$	<code>zncharinduce(G, χ, N)</code>
χp	<code>znchardecompose(G, χ, p)</code>
$\prod_p (\chi, N) \chi p$	<code>znchardecompose(G, χ, Q)</code>
complex Gauss sum $G_a(\chi)$	<code>znchargauss(G, χ)</code>

Conrey labelling

Conrey label $m \in (\mathbf{Z}/q\mathbf{Z})^* \rightarrow$ character	<code>znconreychar(G, m)</code>
character \rightarrow Conrey label	<code>znconreyexp(G, χ)</code>
log on Conrey generators	<code>znconreylog(G, m)</code>
conductor of χ (χ_0 primitive)	<code>znconreyconductor($G, \chi, \{\chi_0\}$)</code>

True-False Tests

is x the disc. of a quadratic field?	<code>isfundamental(x)</code>
is x a prime?	<code>isprime(x)</code>
is x a strong pseudo-prime?	<code>ispseudoprime(x)</code>
is x square-free?	<code>issquarefree(x)</code>
is x a square?	<code>issquare($x, \{\&n\}$)</code>
is x a perfect power?	<code>ispower($x, \{k\}, \{\&n\}$)</code>
is x a perfect power of a prime? ($x = p^n$)	<code>isprimepower($x, \&n$)</code>
... of a pseudoprime?	<code>ispseudoprimepower($x, \&n$)</code>
is x powerful?	<code>ispowerful(x)</code>
is x a totient? ($x = \varphi(n)$)	<code>istotient($x, \{\&n\}$)</code>
is x a polygonal number? ($x = P(s, n)$)	<code>ispolygonal($x, s, \{\&n\}$)</code>
is pol irreducible?	<code>polisirreducible(pol)</code>

Graphic Functions

crude graph of $expr$ between a and b	<code>plot($X = a, b, expr$)</code>
High-resolution plot (immediate plot)	<code>plot(X = a, b, expr, {flag}, {n})</code>
plot $expr$ between a and b	<code>plotoh($X = a, b, expr, \{flag\}, \{n\}$)</code>
plot points given by lists lx, ly	<code>plotthraw($lx, ly, \{flag\}$)</code>
terminal dimensions	<code>plotsizes()</code>

Rectwindow functions

init window w , with size x, y	<code>plotinit(w, x, y)</code>
erase window w	<code>plotkill(w)</code>
copy w to w_2 with offset (dx, dy)	<code>plotcopy(w, w_2, dx, dy)</code>
slice contents of w	<code>plotclip(w)</code>
scale coordinates in w	<code>plotscale(w, x_1, x_2, y_1, y_2)</code>
plotoh in w	<code>plotrecth($w, X = a, b, expr, \{flag\}, \{n\}$)</code>
plotthraw in w	<code>plotrectthraw($w, data, \{flag\}$)</code>
draw window w_1 at $(x_1, y_1), \dots$	<code>plotdraw([[w_1, x_1, y_1], ...])</code>

Low-level Rectwindow Functions

set current drawing color in w to c	<code>plotcolor(w, c)</code>
current position of cursor in w	<code>plotcursor(w)</code>
write s at cursor's position	<code>plotstring(w, s)</code>
move cursor to (x, y)	<code>plotmove(w, x, y)</code>
move cursor to $(x + dx, y + dy)$	<code>plotrmove(w, dx, dy)</code>
draw a box to (x_2, y_2)	<code>plotbox(w, x_2, y_2)</code>
draw a box to $(x + dx, y + dy)$	<code>plotrbox(w, dx, dy)</code>
draw polygon	<code>plotlines($w, lx, ly, \{flag\}$)</code>
draw points	<code>plotpoints(w, lx, ly)</code>
draw line to $(x + dx, y + dy)$	<code>plotrline(w, dx, dy)</code>
draw point $(x + dx, y + dy)$	<code>plotrpoint(w, dx, dy)</code>

Convert to Postscript or Scalable Vector Graphics

The format f is either "ps" or "svg".	
as plotoh	<code>plotexport($f, X = a, b, expr, \{flag\}, \{n\}$)</code>
as plotthraw	<code>plotthrawexport($f, lx, ly, \{flag\}$)</code>
as plotdraw	<code>plotexport($f, [[w_1, x_1, y_1], \dots])$</code>