Libfact - Quick Documentation

A C library to compute generalised factorial functions and co.

2020
October

Quick documentation for libfact. For a more talkative documentation with explanations, references and many gp-code examples, see "libfact-doc.pdf".

Contents

1 Orderings

Prefixes:

- `o` = order
- `r` = removed
- `zk` = ring of integers of `nf`
- *nothing* = finite set

1.1 Regular p-orderings

- `GEN pord(GEN nf, GEN pr, GEN S, long first = 1, long trunc = -1, GEN* ex = NULL, GEN* inv = NULL)`

  Return in a vector beginning by `S[first]` the first `trunc` elements of a `pr`-ordering of the set `S`. If `trunc = -1`, it is set to `#S`. Set to `*ex` the extraction small vector and to `*inv` the invariant vector as it would be returned by `pord_e(nf, pr, S, trunc)`

- `GEN pord_e(GEN nf, GEN pr, GEN S, long trunc = -1)`

  Return in a vector the first `trunc-1` invariant exponents of any `pr`-ordering of the set `S`. If `trunc = -1`, it is set to `#S`.

- `int ispord(GEN nf, GEN pr, GEN S, long trunc = -1, GEN *i = NULL)`

  Return 1 if the sequence of the first `trunc` elements of `S` is the beginning of a `pr`-ordering of the set `S`, 0 otherwise. If `trunc = -1`, it is set to `#S`. If 0 is returned, set to `*i` the index of the first element of `S` responsible for failure.

- `GEN pord_get_e(GEN nf, GEN pr, GEN po, long trunc = -1)`

  Return in a vector the first `trunc - 1` invariant exponents of the `pr`-ordering `po`. If `trunc = -1`, it is set to `#po`.

- `GEN zkpord(GEN nf, GEN pr, long n)`

  Return a `pr`-ordering of length `n` of `nf`.

- `GEN iszkpord(GEN nf GEN pr, GEN S)`

  Return 1 if the sequence `S` is a `pr`-ordering of length `#S` of `nf`, 0 otherwise.

- `GEN zkpord_e(GEN nf, GEN pr, long n)`

  Return the `n` first invariant exponents of any `pr`-ordering of `nf`.

1.2 r-removed p-orderings

- `GEN rpord(GEN nf, GEN pr, GEN S, long r, long trunc = -1, GEN* ex = NULL, GEN* inv = NULL)` Return in a vector the first `trunc` elements of a `r`-removed `pr`-ordering of the set `S`. If `trunc = -1`, it is set to `#S*(r+1)`. Set to `*ex` the extraction small vector and to `*inv` the invariant vector as it would be returned by `rpord_e(nf, pr, S, r, trunc)`.

- `GEN rpord_e(GEN nf, GEN pr, GEN S, long r, long trunc = -1)`

  Return in a vector the first `trunc-1` invariant exponents of any `r`-removed `pr`-ordering of the set `S`. If `trunc = -1`, it is set to `#S*(r+1)`.

- `GEN isrpord(GEN nf, GEN pr, GEN S, long r, long trunc = -1)`

  Return 1 if the sequence of the first `trunc` elements of `S` is the beginning of a `r`-removed `pr`-ordering of the set `S`, 0 otherwise. If `trunc = -1`, it is set to `#S*(r+1)`. Set to `*i` the index of the first element of `S` responsible for failure.

- `GEN rpord_get_e(GEN nf, GEN pr, GEN rpo, long r, long trunc = -1)`

  Return in a vector the first `trunc - 1` invariant exponents of the `r`-removed `pr`-ordering `rpo`. If `trunc = -1`, it is set to `#rpo`.

- `GEN zkrpord(GEN nf, GEN pr, long r, long n)`

  Return a `r`-removed `pr`-ordering of length `n` of `nf`.

- `GEN iszkrpord(GEN nf, GEN pr, long rpo, long r)`

  Return 1 if the sequence `S` is a `r`-removed `pr`-ordering of length `#S` of `nf`, 0 otherwise.

- `GEN zkrpord_e(GEN nf, GEN pr, long r, long n)`

  Return the `n` first invariant exponents of any `r`-removed `pr`-ordering of `nf`.

1.3 p-orderings of order h

- `GEN opord(GEN nf, GEN pr, GEN S, long h, long first, long trunc = -1, GEN* ex = NULL, GEN* inv = NULL)`

  Return in a vector beginning by `S[first]` the first `trunc` elements of a `pr`-ordering of order `h` of the set `S`. If `trunc = -1`, it is set to `#S`. Set to `*ex` the extraction small vector and to `*inv` the vector of invariants as it would be returned by `opord_e(nf,npr, S, r, trunc)`.

- `GEN opord_e(GEN nf, GEN pr, GEN S, long h, long trunc = -1)`

  Return in a vector the first `trunc-1` invariant exponents of any `pr`-ordering of order `h` of the set `S`. If `trunc = -1`, it is set to `#S`.

- `GEN isopord(GEN nf, GEN pr, GEN S, long h, long trunc = -1)`

  Return 1 if the sequence of the first `trunc` elements of `S` is the beginning of a `pr`-ordering of order `h` of the set `S`, 0 otherwise. If `trunc = -1`, it is set to `#S`. Set to `*i` the index of the first element of `S` responsible for failure.

- `GEN opord_get_e(GEN nf, GEN pr, GEN opo, long h, long trunc = -1)`

  Return in a vector the first `trunc - 1` invariant exponents of the `pr`-ordering of order `h` `opo`. If `trunc = -1`, it is set to `#opo`.

- `GEN zkopord(GEN nf, GEN pr, long h, long n)`

  Return a `pr`-ordering of order `h` of length `n` of `nf`.

- `GEN iszkopord(GEN nf, GEN pr, long opo, long h)`

  Return 1 if the sequence `S` is a `pr`-ordering of order `h` of length `#S` of `nf`, 0 otherwise.

- `GEN zkopord_e(GEN nf, GEN pr, long h, long n)`

  Return the `n` first invariant exponents of any `pr`-ordering of order `h` of `nf`.

2 Factorial ideals

Prefixes:

- `s` = finite set
- `zk` = the whole ring of integers of `nf`
- `q` = the whole ring of integers of the quadratic field `nf`
- `rem` = removed
- `mod` = modulus

2.1 Regular factorial ideals

- `GEN sfact(GEN nf, GEN S, long k)`

  Return the `k`-th factorial ideal of the set `S`.

- `GEN sfact_vec(GEN nf, GEN S, long n = -1)`

  Return in a vector the first `n` factorial ideals of the set of algebraic integers `S`. If `n=-1`, it is set to `#S-1`. Faster than building the vector by incremental calls to `sfact`.

- `GEN sfactnorm(GEN nf, GEN S, long k)`

  Return the norm of the `k`-th factorial ideal of the set `S`.

- `GEN sfactnorm_vec(GEN nf, GEN S, long n = -1)`

  Return in a vector the norms of the first `n` factorial ideals of the set `S`. If `n=-1`, it is set to `#S-1`. Faster than building the vector by incremental calls to `sfactnorm`.

- `GEN zkfact(GEN nf, long k)`

  Return the `k`-th factorial ideal of `nf`.

- `GEN zkfact_vec(GEN nf, long n)`

  Return in a vector the `n` first factorial ideals of `nf`. Faster than building the vector by incremental calls to `zkfact`.

- `GEN zkfactnorm(GEN nf, long k)`

  Return the norm of the `k`-th factorial ideal of `nf`. Faster than calling `idealnorm(nf, zkfact(nf, k))`.

- `GEN zkfactnorm_vec(GEN nf, long n)`

  Return in a vector the norm of the `n` first factorial ideals of `nf`. Faster than building the vector by incremental calls to `zkfactnorm`.

- `GEN qfact(GEN nf, long k)`

  Return the `k`-th factorial ideal of the quadratic number field `nf`. Faster than calling `zkfact`.

- `GEN qfact_vec(GEN nf, long n)`

  Return in a vector the first `n` factorial ideals of the quadratic number field `nf`.

- `GEN qfactnorm(GEN nf, long k)`

  Return the norm of the `k`-th factorial ideal of the quadratic number field `nf`. Faster than calling `idealnorm(nf,qfact(nf,k))` or `zkfactnorm`.

- `GEN qfactnorm_vec(GEN nf, long n)`

  Return in a vector the norms of the first `n` factorial ideals of the quadratic number field `nf`.

2.2 r-removed factorial ideals

- `GEN sremfact(GEN nf, GEN S, long r, long k)`

  Return the `k`-th `r`-removed factorial ideal of the set `S`.

- `GEN sremfact_vec(GEN nf, GEN S, long r, long n)`

  Return in a vector the `n` first `r`-removed factorial ideals of the set `S`. If `n = -1`, it is set to `((r+1)*#S)-1`. Faster than building the vector by incremental calls to `sremfact`.

- `GEN sremfactnorm(GEN nf, GEN S, long r, long k)`

  Return the norm of the `k`-th `r`-removed factorial ideal of the set `S`.

- `GEN sremfactnorm_vec(GEN nf, GEN S, long r, long n)`

  Return in a vector the norms of the n `first` `r`-removed factorial ideals of the set `S`. If `n = -1`, it is set to `((r+1)*#S)-1`. Faster than building the vector by incremental calls to `sremfactnorm`.

- `GEN zkremfact(GEN nf, long r, long k)`

  Return the `k`-th `r-removed` factorial ideal of `nf`.

- `GEN zkremfact_vec(GEN nf, long r, long n)`

  Return in a vector the `n` first `r`-removed factorial ideals of `nf`. Faster than building the vector by incremental calls to `zkremfact`.

- `GEN zkremfactnorm(GEN nf, long r, long k)`

  Return the norm of the `k`-th `r`-removed factorial ideal of `nf`.

- `GEN zkremfactnorm_vec(GEN nf, long r, long n)`

  Return in a vector the norms of the first `n` `r`-removed factorial ideals of `nf`. Faster than building the vector by incremental calls to `zkremfactnorm`.

2.3 Factorial ideals of modulus M

- `GEN sfactmod(GEN nf, GEN S, GEN M, long k)`

  Return the `k`-th factorial ideal of modulus `M` of the set `S`.

- `GEN sfactmod_vec(GEN nf, GEN S, GEN M, long n)`

  Return in a vector the `n` first factorial ideals of modulus `M` of the set `S`. If `n = -1`, it is set to `#S - 1`. Faster than building the vector by incremental calls to `sfactmod`.

- `GEN sfactmodnorm(GEN nf, GEN S, GEN M, long k)`

  Return the norm of the `k`-th factorial ideal of modulus `M` of the set `S`.

- `GEN sfactmodnorm_vec(GEN nf, GEN S, GEN M, long n)`

  Return in a vector the norms of the first `n` factorial ideals of modulus `M` of the set `S`. If `n = -1`, it is set to `#S - 1`. Faster than building the vector by incremental calls to `sfactmodnorm`.

- `GEN zkfactmod(GEN nf, GEN M, long k)`

  Return the `k`-th factorial ideal of modulus `M` of `nf`.

- `GEN zkfactmod_vec(GEN nf, GEN M, long n)`

  Return in a vector the `n` first factorial ideals of modulus `M` of `nf`. Faster the building the vector by incremental calls to `zkfactmod`.

- `GEN zkfactmodnorm(GEN nf, GEN M, long k)`

  Return the norm of the `k`-th factorial ideal of modulus `M` of `nf`.

- `GEN zkfactmodnorm_vec(GEN nf, GEN M, long n)`

  Return in a vector the norms of the first `n` factorial ideals of modulus `M` of `nf`. Faster than building the vector by incremental calls to `zkfactmodnorm`.

3 Regular basis

3.1 Regular basis for integer-valued polynomials

- `GEN zkfactpol(GEN nf, long k, const char *s, long cmode = 1)`

  Return a polynomial *pol* of degree `k` in $zk[X]$ such that $pol(zk)$ is included in the `k`-th factorial ideal of `nf`. The variable name is set to `s`. The flag `cmode` tunes the returned polynomial coefficients : 0 for `t_POLMOD`, 1 for `t_POL`, 2 for `t_COL`.

- `GEN zkfactpol_vec(GEN nf, long n, const char *s, long cmode = 1)`

  Return a vector v of length `n` + 1 such that $v[i]$ = `zkfactpol(nf,i-1,s,cmode)`.

- `int ispolyaupto(GEN bnf, long n)`

  Return 1 if the `n` first factorial ideals of `bnf` are principal, 0 otherwise. Useful to test if `zkregbasis` is callable.

- `GEN zkregbasis(GEN bnf, long n, const char *s, long cmode = 1)`

  Return in a vector v of length `n`+1 a regular basis for the zk-module Int(n,X). For such a basis to exist, it is **mandatory** that all factorial ideals up to `n` are principal and this can be checked with the function `ispolyaupto`.

- `GEN zkregbasis_dec(GEN bnf, GEN pol, const char *s)`

  Return if possible a $(n + 1) \times 2$ matrix (where n = deg(`pol`)) with a regular basis in the second column and the coefficients of the K-decomposition of the polynomial `pol` of `bnf[X]` in the first column. The variable in the basis is set to s.

3.2 Regular basis for integer-valued polynomials with integer-valued r-divided differences

- `GEN zkremfactpol(GEN nf, long r, long k, const char *s, long cmode = 1)`

  Return a polynomial *pol* of degree `k` in $zk[X]$ such that $pol(zk)$ generates the `k`-th `r`-removed factorial ideal of `nf`. The variable name is set to `s`. The flag `cmode` tunes the returned polynomial coefficients : 0 for `t_POLMOD`, 1 for `t_POL`, 2 for `t_COL`.

- `GEN zkremfactpol_vec(GEN nf, long r, long n, const char *s, long cmode = 1)`

  Return a vector v of length `n+1` such that $v[i]$ = `zkremfactpol(nf,r,i-1,s,cmode)`.

- `int ispolyaupto_rem(GEN bnf, long r, long n)`

  Return 1 if the `n` first `r`-removed factorial ideals of `bnf` are principal, 0 otherwise. Useful to test if `zkremregbasis` is callable.

- `GEN zkremregbasis(GEN bnf, long r, long n, const char *s, long cmode = 1)`

  Return in a vector v of length `n+1` a regular basis for the zk-module Int(n, r, X).

  It is the module of all integer-valued polynomials of `bnf`[X] of degree at most `n` such that their `r` first divided differences are also integer-valued.

  Being a regular basis means that $\deg(v[i]) = i - 1$ for $1 \leq i \leq n + 1$.

  For such a basis to exist, it is **mandatory** that the `n` first `r`-removed factorial ideals of `bnf` are principal and this can be checked with the function `ispolyaupto_rem`.

  If the later condition is not met, the behavior is undefined. The flag `cmode` tunes the returned polynomial coefficients : 0 for `t_POLMOD`, 1 for `t_POL`, 2 for `t_COL`.

- `GEN zkremregbasis_dec(GEN bnf, GEN pol, long r, const char *s)`

  Return if possible a $(n + 1) \times 2$ matrix (where n = deg(`pol`)) with a regular basis of Int(zk,n,`rr`) in the second column and the coefficients of the K-decomposition of the polynomial `pol` of `bnf[X]` in the first column. The variable in the basis is set to s.

3.3 Regular basis for integer-valued polynomials of modulus M

- `GEN zkfactmodpol(GEN nf, GEN M, long k, const char *s, long cmode = 1)`

  Return a polynomial *pol* of degree `k` in $zk[X]$ such that $pol(zk)$ generates the `k`-th factorial ideal of modulus `M` of `nf`. The variable name is set to `s`. The flag `cmode` tunes the returned polynomial coefficients : 0 for `t_POLMOD`, 1 for `t_POL`, 2 for `t_COL`.

- `GEN zkfactmodpol_vec(GEN nf, GEN M, long n, const char *s, long cmode = 1)`

  Return a vector v of length `n+1` such that $v[i] =$ `zkfactmodpol(nf,M,i-1,s,cmode)`.

- `int ispolyaupto_mod(bnf, M, n)`

  Return 1 if the `n` first factorial ideals of modulus `M` of `bnf` are principal, 0 otherwise. Useful to test id `zkmodregbasis` is callable.

- `GEN nfX_divdiff(GEN nf, GEN pol, long k, GEN* vars = NULL)`

  Return the `k`-th divided difference of the polynomial `pol` $\in$ `nf`$[X]$. The returned polynomial is in `nf`$[x_0, x_1, \ldots, x_n]$. Set to `*vars` the vector of variables $[x_0, \ldots, x_k]$.

- `GEN zkmodregbasis(GEN nf, long h, long n, const char *s, long cmode = 1)`

  Return in a vector v of length `n+1` a regular basis for the zk-module Int(n, M, X).

  It is the module of all integer-valued polynomials *pol* of `bnf`[X] of degree at most `n` such that if I_M is the ideal represented by the modulus `M` and $m \in I_M$, then $pol(mX+s) \in zk[X]$ for all $s \in zk$.

Being a regular basis means that $\deg(v[i]) = i - 1$ for $1 \leq i \leq n + 1$.

For such a basis to exist, it is **mandatory** that the `n` first factorial ideals of modulus `M` of `bnf` are principal and this can be checked with the function `ispolyaupto_mod`.

If the later condition is not met, the behavior is undefined. The flag `cmode` tunes the returned polynomial coefficients : 0 for `t_POLMOD`, 1 for `t_POL`, 2 for `t_COL`.

- `GEN zkmodregbasis_dec(GEN bnf, GEN pol, GEN M, const char *s)`

  Return if possible a $(n + 1) \times 2$ matrix (where n = deg(`pol`)) with a regular basis of Int(zk,n,`M`) in the second column and the coefficients of the K-decomposition of the polynomial `pol` of `bnf[X]` in the first column. The variable name in the basis is set to s.

4 Simultaneous orderings of $\mathbb{Z}_K$

4.1 Almost strong simultaneous ordering of $\mathbb{Z}_K$

- `GEN zkalmostsso(GEN nf, long n, GEN a0 = 0, GEN* ipr = NULL)`

  Return an almost strong simultaneous ordering of length `n` starting by `a0`, i.e a sequence of length `n` of algebraic integers in `nf` satisfying the two following property:

  1. for every prime ideal pr, the sequence obtained by slicing at most one element (depending on pr) is a strong pr-ordering of length `n-1` (or `n` if no slice happened)
  2. every subsequence of $k + 2$ consecutive terms of the sequence is a k-universal set of zk.

  The argument `ipr` (for **i**nitial **pr**imes) can be a single prime ideal or a vector (`t_VEC` or `t_COL`) of prime ideals (possibly empty), those one for which `a0` might have to be sliced to satisfy the first property.

  In particular, the following returns a n-universal set of zk : `zkalmostsso(nf, n + 2)`.

- `int zkissimulord(GEN nf, GEN S)`

  Return 1 if the sequence `S` is a simultaneous ordering of length #`S` $- 1$ of `nf`, 0 otherwise.

4.2 Simultaneous ordering of the ring of integers of a quadratic number field

- `GEN qallsimulord(GEN nf, long n)`

  Return in a vector of vectors all basal (i.e starting by `[0,1]`) simultaneous ordering of length `n` of the quadratic number field `nf`.

- `GEN qrso_testfirstnonsplit(GEN d)`

  The argument `d` is a positive squarefree integer and represents the real quadratic number field $nf = \mathbb{Q}(\sqrt{d})$. Let m_d be the least prime who does not split in nf as returned by `qfirstnonsplit`. This function will test efficiently if there exist a basal simultaneous ordering of length m_d in $\mathbb{Q}(\sqrt{d})$ such that the first $m_d - 1$ terms are contained in $\mathbb{Z}$ and return in a vector the candidates if any, the empty vector otherwise.

- `int qrso_search(GEN first, GEN upto, int verbose = 0, GEN *found = NULL)`

  This function looks for a real quadratic number field $\mathbb{Q}(\sqrt{d})$, for d running from `first` to `upto` and $d = 1 \pmod 8$, such that there exist a sequence of length superior or equal to $m_d =$ `qfirstnonsplit(d)` with the first $m_d - 1$ terms contained in $\mathbb{Z}$. The function returns 1 if some exception is found and set to `*found` the value of d, 0 otherwise. Setting `verbose` to 1 will print informations about the search on standart output.

5 Miscellaneous

- `GEN allpord(GEN nf, GEN pr, GEN S, GEN SS, long trunc = -1, GEN *ex = NULL)`

  Return in a vector of vectors all sequences of `trunc` elements of `S` beginning by the subsequence `SS` which are the beginning of a `pr`-ordering of the set `S`. If `trunc = -1`, it is set to `#S`. Set to `*ex` all the extraction small vectors. The argument `SS` can also be a `t_INT` i which is interpreted as if `SS` $=$ `[S[i]]`.

- `GEN simulord(GEN nf, GEN S, long trunc = -1, GEN *ex = NULL)`

  Return in a vector the first `trunc` elements of a possible simultaneous ordering (or Newton sequence) of the set `S`. If no such sequence exists, return an empty vector. If `trunc = -1`, it is set to `#S`. Set to `*ex` the extraction small vector.

- `GEN issimulord(GEN nf, GEN S, long trunc = -1, GEN *i = NULL)`

  Return 1 if the sequence of the first `trunc` elements of `S` could be the beginning of a simultaneous ordering of the set `S`, 0 otherwise. If `trunc = -1`, it is set to `#S`. If 0 is returned, set to `*i` the index of the first element responsible for failure.

- `GEN allsimulord(GEN nf, GEN S, GEN SS, long trunc = -1, GEN *ex = NULL)`

  Return in a vector of vectors all sequences of `trunc` elements of `S` beginning by the subsequence `SS` which could be the beginning of a simultaneous ordering of the set `S`. If `trunc = -1`, it is set to `#S`. Set to `*ex` all the extraction small vectors. The argument `SS` can also be a `t_INT` i which is interpreted as if `SS` $=$ `[S[i]]`.

- `GEN strongpord(GEN nf, GEN pr, long n)`

  Return a strong `pr`-ordering of length `n` of `nf`. The argument `pr` can be a single prime ideal or a vector of prime ideals in which case the returned sequence will be a strong p-ordering for every prime ideal p in the vector `pr`.

- `int isstrongpord(GEN nf, GEN pr, GEN S)` Return 1 if the sequence `S` is a strong `pr`-ordering of length #`S` of `nf`, 0 otherwise. The argument `pr` can be a single prime ideal or a vector of prime ideals in which case the function return 1 if `S` is a strong p-ordering for every prime ideal p in the vector `pr`.

6 Useful functions

- `GEN vdiffprod(GEN nf, GEN v, GEN x)`

  Return the product of differences of `x` with components of the vector `v`.

- `GEN vdiffprod_i(GEN nf, GEN v, long i)`

  Equivalent to `vdiffprod(nf, v[1..i-1], v[i])`.

- `GEN vdiffs(GEN nf, GEN v)`

  Return the vector `[vdiffprod_i(nf, v, i)]`, $2 \leq i \leq \#$`v`.

- `GEN volume(GEN nf, GEN v)`

  Return the volume of the vector `v`, i.e the product of all distinct pairs of elements of `v`. Volume is defined up to ± 1.

- `GEN volume_i(GEN nf, GEN v, long i)`

  Equivalent to `volume(nf, vec_shorten(S, i))`.

- `GEN volume2(GEN nf, GEN v)`

  This is just the square of `volume`, sometimes preferred to `volume`.

- `GEN qfirstnonsplit(GEN nf)`

  Return the first prime number who does not split in the quadratic number field `nf`. The argument `nf` can also be a fundamental discriminant or a squarefree integer.

- `GEN idealmaxlist(GEN nf, long n)`

  Same as `ideallist` but for maximal ideals.

- `GEN idealmaxprod(GEN nf, GEN p, long k)`

  Return the product of all maximal ideals of norm equal to p^k.

- `GEN qfunorm(GEN nf)`

  Return the norm of the fundamental unit of the quadratic number field `nf`. The argument `nf` can also be a fundamental discriminant or a squarefree integer.

- `GEN legf(GEN q, GEN n)`

  Generalised Legendre formula. If $v_q(n)$ is the exponent of the highest power of $q \geq 2$ dividing n, the function computes $w_q(n) = \sum_{i=1}^{n} v_q(i)$. If `q` is prime, this is just the `q` valuation of `n!`.

- `GEN legf_vec(GEN q, GEN n)`

  Return the vector `[legf(q, k)]`, $1 \leq k \leq n$.

- `GEN rlegf(GEN q, GEN n, long r)`

  Analogous of Legendre formula for `r`-removed $\mathfrak{p}$-ordering.

  The function returns $\texttt{legf}(q,n) - \texttt{legf}(q, \left\lfloor \frac{n}{q^k} \right\rfloor) - kr$ where $k = \left\lfloor \frac{\log \frac{n}{r}}{\log q} \right\rfloor$.

- `GEN rlegf_vec(GEN q, long r, GEN n)`

  Return the vector `[rlegf(q,k,r)]`, $1 \leq k \leq n$.

- `GEN olegf(GEN q, GEN n, GEN h)`

  Analogous of Legendre formula for $\mathfrak{p}$-orderings of order `h`.

  The function returns $\texttt{legf}(q,n) - \texttt{legf}(q, \left\lfloor \frac{n}{q^h} \right\rfloor)$.

- `GEN olegf_vec(GEN q, GEN h, long n)`

  Return the vector `[olegf(q, k, h)]`, $1 \leq k \leq n$.

- `int qispolya(GEN nf)`

  Return 1 if the quadratic number field `nf` is a Polya number field, 0 otherwise. If it is the case, `zkregbasis` is callable without restriction.