

MLGMPIDL: OCaml interface for GMP and MPFR libraries  
(version 1.2.1)

July 9, 2015

---

All files distributed in the MLGMPIDL interface are distributed under LGPL license.  
Copyright (C) Bertrand Jeannet 2005-2009 for the MLGMPIDL interface.

# Contents

<b>1</b>	<b>Module Introduction</b>	<b>3</b>
<b>2</b>	<b>Module <code>Mpz</code> : GMP multi-precision integers</b>	<b>4</b>
<b>3</b>	<b>Module <code>Mpq</code> : GMP multi-precision rationals</b>	<b>9</b>
<b>4</b>	<b>Module <code>Mpf</code> : GMP multi-precision floating-point numbers</b>	<b>11</b>
<b>5</b>	<b>Module <code>Mpfr</code> : MPFR multi-precision floating-point numbers</b>	<b>14</b>
<b>6</b>	<b>Module <code>Gmp_random</code> : GMP random generation functions</b>	<b>19</b>
<b>7</b>	<b>Module <code>Mpzf</code> : GMP multi-precision integers, functional version</b>	<b>20</b>
<b>8</b>	<b>Module <code>Mpqf</code> : GMP multi-precision rationals, functional version</b>	<b>22</b>
<b>9</b>	<b>Module <code>Mpfrf</code> : MPFR multi-precision floating-point version, functional version</b>	<b>24</b>

## Chapter 1

# Module Introduction

## Chapter 2

# Module Mpz : GMP multi-precision integers

```
type 'a tt
type m
    Mutable tag

type f
    Functional (immutable) tag

type t = m tt
    Mutable multi-precision integer

val print : Format.formatter -> 'a tt -> unit
val init : unit -> 'a tt
val init2 : int -> 'a tt
val realloc2 : t -> int -> unit
val set : t -> 'a tt -> unit
val set_si : t -> int -> unit
val set_d : t -> float -> unit
val _set_str : t -> string -> int -> unit
val set_str : t -> string -> base:int -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> 'b tt
val init_set_si : int -> 'a tt
val init_set_d : float -> 'a tt
val _init_set_str : string -> int -> 'a tt
val init_set_str : string -> base:int -> t
val get_si : 'a tt -> nativeint
val get_int : 'a tt -> int
val get_d : 'a tt -> float
val get_d_2exp : 'a tt -> float * int
val _get_str : int -> 'a tt -> string
val get_str : base:int -> 'a tt -> string
```

```
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val of_string : string -> 'a tt
val of_float : float -> 'a tt
val of_int : int -> 'a tt
val add : t -> 'a tt -> 'b tt -> unit
val add_ui : t -> 'a tt -> int -> unit
val sub : t -> 'a tt -> 'b tt -> unit
val sub_ui : t -> 'a tt -> int -> unit
val ui_sub : t -> int -> 'a tt -> unit
val mul : t -> 'a tt -> 'b tt -> unit
val mul_si : t -> 'a tt -> int -> unit
val addmul : t -> 'a tt -> 'b tt -> unit
val addmul_ui : t -> 'a tt -> int -> unit
val submul : t -> 'a tt -> 'b tt -> unit
val submul_ui : t -> 'a tt -> int -> unit
val mul_2exp : t -> 'a tt -> int -> unit
val neg : t -> 'a tt -> unit
val abs : t -> 'a tt -> unit
val cdiv_q : t -> 'a tt -> 'b tt -> unit
```

The first parameter holds the quotient.

```
val cdiv_r : t -> 'a tt -> 'b tt -> unit
```

The first parameter holds the remainder.

```
val cdiv_qr : t -> t -> 'a tt -> 'b tt -> unit
```

The two first parameters hold resp. the quotient and the remainder).

```
val cdiv_q_ui : t -> 'a tt -> int -> int
```

The first parameter holds the quotient.

```
val cdiv_r_ui : t -> 'a tt -> int -> int
```

The first parameter holds the remainder.

```
val cdiv_qr_ui : t -> t -> 'a tt -> int -> int
```

The two first parameters hold resp. the quotient and the remainder).

```
val cdiv_ui : 'a tt -> int -> int
```

```
val cdiv_q_2exp : t -> 'a tt -> int -> unit
```

The first parameter holds the quotient.

```
val cdiv_r_2exp : t -> 'a tt -> int -> unit
```

The first parameter holds the remainder.

```
val fdiv_q : t -> 'a tt -> 'b tt -> unit
```

```
val fdiv_r : t -> 'a tt -> 'b tt -> unit
```

```
val fdiv_qr : t -> t -> 'a tt -> 'b tt -> unit
```

```
val fdiv_q_ui : t -> 'a tt -> int -> int
```

---

```

val fdiv_r_ui : t -> 'a tt -> int -> int
val fdiv_qr_ui : t -> t -> 'a tt -> int -> int
val fdiv_ui : 'a tt -> int -> int
val fdiv_q_2exp : t -> 'a tt -> int -> unit
val fdiv_r_2exp : t -> 'a tt -> int -> unit
val tdiv_q : t -> 'a tt -> 'b tt -> unit
val tdiv_r : t -> 'a tt -> 'b tt -> unit
val tdiv_qr : t -> t -> 'a tt -> 'b tt -> unit
val tdiv_q_ui : t -> 'a tt -> int -> int
val tdiv_r_ui : t -> 'a tt -> int -> int
val tdiv_qr_ui : t -> t -> 'a tt -> int -> int
val tdiv_ui : 'a tt -> int -> int
val tdiv_q_2exp : t -> 'a tt -> int -> unit
val tdiv_r_2exp : t -> 'a tt -> int -> unit
val gmod : t -> 'a tt -> 'b tt -> unit
val gmod_ui : t -> 'a tt -> int -> int
val divexact : t -> 'a tt -> 'b tt -> unit
val divexact_ui : t -> 'a tt -> int -> unit
val divisible_p : 'a tt -> 'b tt -> bool
val divisible_ui_p : 'a tt -> int -> bool
val divisible_2exp_p : 'a tt -> int -> bool
val congruent_p : 'a tt -> 'b tt -> 'c tt -> bool
val congruent_ui_p : 'a tt -> int -> int -> bool
val congruent_2exp_p : 'a tt -> 'b tt -> int -> bool
val _powm : t -> 'a tt -> 'b tt -> 'c tt -> unit
val _powm_ui : t -> 'a tt -> int -> 'b tt -> unit
val powm : t -> 'a tt -> 'b tt -> modulo:'c tt -> unit
val powm_ui : t -> 'a tt -> int -> modulo:'b tt -> unit
val pow_ui : t -> 'a tt -> int -> unit
val ui_pow_ui : t -> int -> int -> unit
val root : t -> 'a tt -> int -> bool
val sqrt : t -> 'a tt -> unit
val _sqrtrem : t -> t -> 'a tt -> unit
val sqrtrem : t -> remainder:t -> 'a tt -> unit
val perfect_power_p : 'a tt -> bool
val perfect_square_p : 'a tt -> bool
val probab_prime_p : 'a tt -> int -> int
val nextprime : t -> 'a tt -> unit
val gcd : t -> 'a tt -> 'b tt -> unit
val gcd_ui : t option -> 'a tt -> int -> int
val _gcdext : t -> t -> t -> 'a tt -> 'b tt -> unit
val gcdext : gcd:t -> alpha:t -> beta:t -> 'a tt -> 'b tt -> unit
val lcm : t -> 'a tt -> 'b tt -> unit
val lcm_ui : t -> 'a tt -> int -> unit
val invert : t -> 'a tt -> 'b tt -> bool

```

---

```

val jacobi : 'a tt -> 'b tt -> int
val legendre : 'a tt -> 'b tt -> int
val kronecker : 'a tt -> 'b tt -> int
val kronecker_si : 'a tt -> int -> int
val si_kronecker : int -> 'a tt -> int
val remove : t -> 'a tt -> 'b tt -> int
val fac_ui : t -> int -> unit
val bin_ui : t -> 'a tt -> int -> unit
val bin_uiui : t -> int -> int -> unit
val fib_ui : t -> int -> unit
val fib2_ui : t -> t -> int -> unit
val lucnum_ui : t -> int -> unit
val lucnum2_ui : t -> t -> int -> unit
val cmp : 'a tt -> 'b tt -> int
val cmp_d : 'a tt -> float -> int
val cmp_si : 'a tt -> int -> int
val cmpabs : 'a tt -> 'b tt -> int
val cmpabs_d : 'a tt -> float -> int
val cmpabs_ui : 'a tt -> int -> int
val sgn : 'a tt -> int
val gand : t -> 'a tt -> 'b tt -> unit
val ior : t -> 'a tt -> 'b tt -> unit
val xor : t -> 'a tt -> 'b tt -> unit
val com : t -> 'a tt -> unit
val popcount : 'a tt -> int
val hamdist : 'a tt -> 'b tt -> int
val scan0 : 'a tt -> int -> int
val scan1 : 'a tt -> int -> int
val setbit : t -> int -> unit
val clrbit : t -> int -> unit
val tstbit : 'a tt -> int -> bool
val _import :
  t ->
    (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t ->
      int -> int -> unit
val _export :
  'a tt ->
    int -> int -> (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t
val import :
  dest:t ->
    (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t ->
      order:int -> endian:int -> unit
val export :
  'a tt ->
    order:int ->
      endian:int -> (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t
val fits_int_p : 'a tt -> bool

```



```
val odd_p : 'a tt -> bool
val even_p : 'a tt -> bool
val size : 'a tt -> int
val sizeinbase : 'a tt -> int -> int
val fits_ulong_p : 'a tt -> bool
val fits_slong_p : 'a tt -> bool
val fits_uint_p : 'a tt -> bool
val fits_sint_p : 'a tt -> bool
val fits_ushort_p : 'a tt -> bool
val fits_sshort_p : 'a tt -> bool
```

## Chapter 3

# Module Mpq : GMP multi-precision rationals

```
type 'a tt
type m
    Mutable tag

type f
    Functional (immutable) tag

type t = m tt
    Mutable multi-precision rationals

val canonicalize : 'a tt -> unit
val print : Format.formatter -> 'a tt -> unit
val init : unit -> 'a tt
val set : t -> 'a tt -> unit
val set_z : t -> 'a Mpz.tt -> unit
val set_si : t -> int -> int -> unit
val _set_str : t -> string -> int -> unit
val set_str : t -> string -> base:int -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> 'b tt
val init_set_z : 'a Mpz.tt -> 'b tt
val init_set_si : int -> int -> 'a tt
val init_set_str : string -> base:int -> 'a tt
val init_set_d : float -> 'a tt
val get_d : 'a tt -> float
val set_d : t -> float -> unit
val get_z : Mpz.t -> 'a tt -> unit
val _get_str : int -> 'a tt -> string
val get_str : base:int -> t -> string
val to_string : 'a tt -> string
val to_float : 'a tt -> float
```

```
val of_string : string -> 'a tt
val of_float : float -> 'a tt
val of_int : int -> 'a tt
val of_frac : int -> int -> 'a tt
val of_mpz : 'a Mpz.tt -> 'b tt
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> 'c tt
val add : t -> 'a tt -> 'b tt -> unit
val sub : t -> 'a tt -> 'b tt -> unit
val mul : t -> 'a tt -> 'b tt -> unit
val mul_2exp : t -> 'a tt -> int -> unit
val div : t -> 'a tt -> 'b tt -> unit
val div_2exp : t -> 'a tt -> int -> unit
val neg : t -> 'a tt -> unit
val abs : t -> 'a tt -> unit
val inv : t -> 'a tt -> unit
val cmp : 'a tt -> 'b tt -> int
val cmp_si : 'a tt -> int -> int -> int
val sgn : 'a tt -> int
val equal : 'a tt -> 'b tt -> bool
val get_num : Mpz.t -> 'a tt -> unit
val get_den : Mpz.t -> 'a tt -> unit
val set_num : t -> 'a Mpz.tt -> unit
val set_den : t -> 'a Mpz.tt -> unit
```

## Chapter 4

# Module Mpf : GMP multi-precision floating-point numbers

```
type 'a tt
type m
    Mutable tag

type f
    Functional (immutable) tag

type t = m tt
    Mutable multi-precision floating-point numbers

val print : Format.formatter -> 'a tt -> unit
val set_default_prec : int -> unit
val get_default_prec : unit -> int
val init : unit -> 'a tt
val init2 : int -> 'a tt
val get_prec : 'a tt -> int
val set_prec : t -> int -> unit
val set_prec_raw : t -> int -> unit
val set : t -> 'a tt -> unit
val set_si : t -> int -> unit
val set_d : t -> float -> unit
val set_z : t -> 'a Mpz.tt -> unit
val set_q : t -> 'a Mpq.tt -> unit
val _set_str : t -> string -> int -> unit
val set_str : t -> string -> base:int -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> 'b tt
val init_set_si : int -> 'a tt
val init_set_d : float -> 'a tt
val _init_set_str : string -> int -> 'a tt
val init_set_str : string -> base:int -> 'a tt
```

```

val get_d : 'a tt -> float
val get_d_2exp : 'a tt -> float * int
val get_si : 'a tt -> nativeint
val get_int : 'a tt -> int
val get_z : Mpz.t -> 'a tt -> unit
val get_q : Mpq.t -> 'a tt -> unit
val _get_str : int -> int -> 'a tt -> string * int
val get_str : base:int -> digits:int -> 'a tt -> string * int
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val of_string : string -> 'a tt
val of_float : float -> 'a tt
val of_int : int -> 'a tt
val of_mpz : 'a Mpz.tt -> 'b tt
val of_mpq : 'a Mpq.tt -> 'b tt
val is_integer : 'a tt -> bool
val add : t -> 'a tt -> 'b tt -> unit
val add_ui : t -> 'a tt -> int -> unit
val sub : t -> 'a tt -> 'b tt -> unit
val ui_sub : t -> int -> 'a tt -> unit
val sub_ui : t -> 'a tt -> int -> unit
val mul : t -> 'a tt -> 'b tt -> unit
val mul_ui : t -> 'a tt -> int -> unit
val mul_2exp : t -> 'a tt -> int -> unit
val div : t -> 'a tt -> 'b tt -> unit
val ui_div : t -> int -> 'a tt -> unit
val div_ui : t -> 'a tt -> int -> unit
val div_2exp : t -> 'a tt -> int -> unit
val sqrt : t -> 'a tt -> unit
val pow_ui : t -> 'a tt -> int -> unit
val neg : t -> 'a tt -> unit
val abs : t -> 'a tt -> unit
val cmp : 'a tt -> 'b tt -> int
val cmp_d : 'a tt -> float -> int
val cmp_si : 'a tt -> int -> int
val sgn : 'a tt -> int
val _equal : 'a tt -> 'b tt -> int -> bool
val equal : 'a tt -> 'a tt -> bits:int -> bool
val reldiff : t -> 'a tt -> 'b tt -> unit
val ceil : t -> 'a tt -> unit
val floor : t -> 'a tt -> unit
val trunc : t -> 'a tt -> unit
val integer_p : 'a tt -> bool
val fits_int_p : 'a tt -> bool
val fits_ulong_p : 'a tt -> bool

```

```
val fits_slong_p : 'a tt -> bool
val fits_uint_p : 'a tt -> bool
val fits_sint_p : 'a tt -> bool
val fits_ushort_p : 'a tt -> bool
val fits_sshort_p : 'a tt -> bool
```

## Chapter 5

# Module Mpfr : MPFR multi-precision floating-point numbers

```
type 'a tt
type round =
  | Near
  | Zero
  | Up
  | Down
  | Away
  | Faith
  | NearAway
type m
  Mutable tag

type f
  Functional (immutable) tag

type t = m tt
  Mutable multi-precision floating-point numbers

val print : Format.formatter -> 'a tt -> unit
val print_round : Format.formatter -> round -> unit
val string_of_round : round -> string
val set_default_rounding_mode : round -> unit
val get_default_rounding_mode : unit -> round
val round_prec : t -> round -> int -> int
val get_emin : unit -> int
val get_emax : unit -> int
val set_emin : int -> unit
val set_emax : int -> unit
val check_range : t -> int -> round -> int
val clear_underflow : unit -> unit
```

```

val clear_overflow : unit -> unit
val clear_nanflag : unit -> unit
val clear_inexflag : unit -> unit
val clear_flags : unit -> unit
val underflow_p : unit -> bool
val overflow_p : unit -> bool
val nanflag_p : unit -> bool
val inexflag_p : unit -> bool
val set_default_prec : int -> unit
val get_default_prec : unit -> int
val init : unit -> 'a tt
val init2 : int -> 'a tt
val get_prec : 'a tt -> int
val set_prec : t -> int -> unit
val set_prec_raw : t -> int -> unit
val set : t -> 'a tt -> round -> int
val set_si : t -> int -> round -> int
val set_d : t -> float -> round -> int
val set_z : t -> 'a Mpz.tt -> round -> int
val set_q : t -> 'a Mpq.tt -> round -> int
val _set_str : t -> string -> int -> round -> unit
val set_str : t -> string -> base:int -> round -> unit
val _strtoufr : t -> string -> int -> round -> int * int
val strtoufr : t -> string -> base:int -> round -> int * int
val set_f : t -> 'a Mpf.tt -> round -> int
val set_si_2exp : t -> int -> int -> round -> int
val set_inf : t -> int -> unit
val set_nan : t -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> round -> int * 'b tt
val init_set_si : int -> round -> int * 'a tt
val init_set_d : float -> round -> int * 'a tt
val init_set_f : 'a Mpf.tt -> round -> int * 'b tt
val init_set_z : 'a Mpz.tt -> round -> int * 'b tt
val init_set_q : 'a Mpq.tt -> round -> int * 'b tt
val _init_set_str : string -> int -> round -> 'a tt
val init_set_str : string -> base:int -> round -> 'a tt
val get_d : 'a tt -> round -> float
val get_d1 : 'a tt -> float
val get_z_exp : Mpz.t -> 'a tt -> int
val get_z : Mpz.t -> 'a tt -> round -> unit
val _get_str : int -> int -> 'a tt -> round -> string * int
val get_str : base:int -> digits:int -> t -> round -> string * int
val to_string : 'a tt -> string
val to_float : ?round:round -> 'a tt -> float

```



```

val to_mpq : 'a tt -> 'b Mpq.tt
val of_string : string -> round -> 'a tt
val of_float : float -> round -> 'a tt
val of_int : int -> round -> 'a tt
val of_frac : int -> int -> round -> 'a tt
val of_mpz : 'a Mpz.tt -> round -> 'b tt
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> round -> 'c tt
val of_mpq : 'a Mpq.tt -> round -> 'b tt
val add : t -> 'a tt -> 'b tt -> round -> int
val add_ui : t -> 'a tt -> int -> round -> int
val add_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val add_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val sub : t -> 'a tt -> 'b tt -> round -> int
val ui_sub : t -> int -> 'a tt -> round -> int
val sub_ui : t -> 'a tt -> int -> round -> int
val sub_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val sub_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val mul : t -> 'a tt -> 'b tt -> round -> int
val mul_ui : t -> 'a tt -> int -> round -> int
val mul_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val mul_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val mul_2ui : t -> 'a tt -> int -> round -> int
val mul_2si : t -> 'a tt -> int -> round -> int
val mul_2exp : t -> 'a tt -> int -> round -> int
val div : t -> 'a tt -> 'b tt -> round -> int
val ui_div : t -> int -> 'a tt -> round -> int
val div_ui : t -> 'a tt -> int -> round -> int
val div_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val div_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val div_2ui : t -> 'a tt -> int -> round -> int
val div_2si : t -> 'a tt -> int -> round -> int
val div_2exp : t -> t -> int -> round -> int
val sqrt : t -> 'a tt -> round -> bool
val sqrt_ui : t -> int -> round -> bool
val pow_ui : t -> 'a tt -> int -> round -> bool
val pow_si : t -> 'a tt -> int -> round -> bool
val ui_pow_ui : t -> int -> int -> round -> bool
val ui_pow : t -> int -> 'a tt -> round -> bool
val pow : t -> 'a tt -> 'b tt -> round -> bool
val neg : t -> 'a tt -> round -> int
val abs : t -> 'a tt -> round -> int
val cmp : 'a tt -> 'b tt -> int
val cmp_si : 'a tt -> int -> int
val cmp_si_2exp : 'a tt -> int -> int -> int
val sgn : 'a tt -> int

```

```

val _equal : 'a tt -> 'b tt -> int -> bool
val equal : 'a tt -> 'b tt -> bits:int -> bool
val nan_p : 'a tt -> bool
val inf_p : 'a tt -> bool
val number_p : 'a tt -> bool
val reldiff : t -> 'a tt -> 'b tt -> round -> unit
val log : t -> 'a tt -> round -> int
val log2 : t -> 'a tt -> round -> int
val log10 : t -> 'a tt -> round -> int
val exp : t -> 'a tt -> round -> int
val exp2 : t -> 'a tt -> round -> int
val exp10 : t -> 'a tt -> round -> int
val cos : 'a tt -> 'b tt -> round -> int
val sin : 'a tt -> 'b tt -> round -> int
val tan : 'a tt -> 'b tt -> round -> int
val sec : 'a tt -> 'b tt -> round -> int
val csc : 'a tt -> 'b tt -> round -> int
val cot : 'a tt -> 'b tt -> round -> int
val sin_cos : 'a tt -> 'b tt -> 'c tt -> round -> bool
val acos : t -> 'a tt -> round -> int
val asin : t -> 'a tt -> round -> int
val atan : t -> 'a tt -> round -> int
val atan2 : t -> 'a tt -> 'b tt -> round -> int
val cosh : 'a tt -> 'b tt -> round -> int
val sinh : 'a tt -> 'b tt -> round -> int
val tanh : 'a tt -> 'b tt -> round -> int
val sech : 'a tt -> 'b tt -> round -> int
val csch : 'a tt -> 'b tt -> round -> int
val coth : 'a tt -> 'b tt -> round -> int
val acosh : t -> 'a tt -> round -> int
val asinh : t -> 'a tt -> round -> int
val atanh : t -> 'a tt -> round -> int
val fac_ui : t -> int -> round -> int
val log1p : t -> 'a tt -> round -> int
val expm1 : t -> 'a tt -> round -> int
val eint : t -> 'a tt -> round -> int
val gamma : t -> 'a tt -> round -> int
val lngamma : t -> 'a tt -> round -> int
val zeta : t -> 'a tt -> round -> int
val erf : t -> 'a tt -> round -> int
val erfc : t -> 'a tt -> round -> int
val j0 : t -> 'a tt -> round -> int
val j1 : t -> 'a tt -> round -> int
val jn : t -> int -> 'a tt -> round -> int
val y0 : t -> 'a tt -> round -> int

```

```

val y1 : t -> 'a tt -> round -> int
val yn : t -> int -> 'a tt -> round -> int
val fma : t -> 'a tt -> 'b tt -> 'c tt -> round -> int
val fms : t -> 'a tt -> 'b tt -> 'c tt -> round -> int
val agm : t -> 'a tt -> 'b tt -> round -> int
val hypot : t -> 'a tt -> 'b tt -> round -> int
val const_log2 : t -> round -> int
val const_pi : t -> round -> int
val const_euler : t -> round -> int
val const_catalan : t -> round -> int
val rint : t -> 'a tt -> round -> int
val ceil : t -> 'a tt -> int
val floor : t -> 'a tt -> int
val round : t -> 'a tt -> int
val trunc : t -> 'a tt -> int
val frac : t -> 'a tt -> round -> int
val modf : t -> t -> 'a tt -> round -> int
val fmod : t -> 'a tt -> 'b tt -> round -> int
val remainder : t -> 'a tt -> 'b tt -> round -> int
val integer_p : 'a tt -> bool
val nexttoward : t -> 'a tt -> unit
val nextabove : t -> unit
val nextbelow : t -> unit
val min : t -> 'a tt -> 'b tt -> round -> int
val max : t -> 'a tt -> 'b tt -> round -> int
val get_exp : 'a tt -> int
val set_exp : t -> int -> int

```

## Chapter 6

# Module Gmp\_random : GMP random generation functions

```
type state
val init_default : unit -> state
val init_lc_2exp : 'a Mpz.tt -> int -> int -> state
val init_lc_2exp_size : int -> state
val seed : state -> 'a Mpz.tt -> unit
val seed_ui : state -> int -> unit
module Mpz :
  sig
    val urandomb : Mpz.t -> Gmp_random.state -> int -> unit
    val urandomm : Mpz.t -> Gmp_random.state -> 'a Mpz.tt -> unit
    val rrandomb : Mpz.t -> Gmp_random.state -> int -> unit
  end
module Mpf :
  sig
    val urandomb : Mpf.t -> Gmp_random.state -> int -> unit
  end
module Mpfr :
  sig
    val urandomb : Mpfr.t -> Gmp_random.state -> unit
    val urandom : 'a Mpfr.tt -> Gmp_random.state -> Mpfr.round -> unit
  end
```

## Chapter 7

# Module Mpzf : GMP multi-precision integers, functional version

```
type 'a tt = 'a Mpz.tt
type t = Mpz.f tt
    multi-precision integer

val _mpz : t -> Mpz.t
val _mpzf : Mpz.t -> t
val to_mpz : t -> 'a Mpz.tt
val of_mpz : 'a Mpz.tt -> t
    Safe conversion from and to Mpz.t.
    There is no sharing between the argument and the result.

val print : Format.formatter -> 'a tt -> unit
val of_string : string -> t
val of_float : float -> t
val of_int : int -> t
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val add : 'a tt -> 'b tt -> t
val add_int : 'a tt -> int -> t
val sub : 'a tt -> 'b tt -> t
val sub_int : 'a tt -> int -> t
val mul : 'a tt -> 'b tt -> t
val mul_int : 'a tt -> int -> t
val cdiv_q : 'a tt -> 'b tt -> t
val cdiv_r : 'a tt -> 'b tt -> t
val cdiv_qr : 'a tt -> 'b tt -> t * t
val fdiv_q : 'a tt -> 'b tt -> t
val fdiv_r : 'a tt -> 'b tt -> t
val fdiv_qr : 'a tt -> 'b tt -> t * t
val tdiv_q : 'a tt -> 'b tt -> t
val tdiv_r : 'a tt -> 'b tt -> t
```

```
val tdiv_qr : 'a tt -> 'b tt -> t * t
val divexact : 'a tt -> 'b tt -> t
val gmod : 'a tt -> 'b tt -> t
val gcd : 'a tt -> 'b tt -> t
val lcm : 'a tt -> 'b tt -> t
val neg : 'a tt -> t
val abs : 'a tt -> t
val cmp : 'a tt -> 'b tt -> int
val cmp_int : 'a tt -> int -> int
val sgn : 'a tt -> int
```

## Chapter 8

# Module Mpqf : GMP multi-precision rationals, functional version

```
type 'a tt = 'a Mpq.tt
type t = Mpq.f tt
    multi-precision rationals

val to_mpq : t -> 'a Mpq.tt
val of_mpq : 'a Mpq.tt -> t
    Safe conversion from and to Mpq.t.
    There is no sharing between the argument and the result.

val _mpq : t -> Mpq.t
val _mpqf : Mpq.t -> t
    Unsafe conversion from and to Mpq.t.
    Sharing between the argument and the result.

val print : Format.formatter -> 'a tt -> unit
val of_string : string -> t
val of_float : float -> t
val of_int : int -> t
val of_frac : int -> int -> t
val of_mpz : 'a Mpz.tt -> t
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> t
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val to_mpzf2 : 'a tt -> Mpzf.t * Mpzf.t
val add : 'a tt -> 'b tt -> t
val sub : 'a tt -> 'b tt -> t
val mul : 'a tt -> 'b tt -> t
val div : 'a tt -> 'b tt -> t
val neg : 'a tt -> t
val abs : 'a tt -> t
val inv : 'a tt -> t
```

```
val equal : 'a tt -> 'b tt -> bool
val cmp : 'a tt -> 'b tt -> int
val cmp_int : 'a tt -> int -> int
val cmp_frac : 'a tt -> int -> int -> int
val sgn : 'a tt -> int
val get_num : t -> Mpzf.t
val get_den : t -> Mpzf.t
```



## Chapter 9

# Module Mpfrf : MPFR multi-precision floating-point version, functional version

```
type 'a tt = 'a Mpfr.tt
type t = Mpfr.f tt
    multi-precision floating-point numbers

val to_mpfr : t -> 'a Mpfr.tt
val of_mpfr : 'a Mpfr.tt -> t
    Safe conversion from and to Mpfr.t.
    There is no sharing between the argument and the result.

val _mpfr : t -> Mpfr.t
val _mpfrf : Mpfr.t -> t
    Unsafe conversion from and to Mpfr.t.
    The argument and the result actually share the same number: be cautious !

val print : Format.formatter -> t -> unit
val of_string : string -> Mpfr.round -> t
val of_float : float -> Mpfr.round -> t
val of_int : int -> Mpfr.round -> t
val of_frac : int -> int -> Mpfr.round -> t
val of_mpz : 'a Mpz.tt -> Mpfr.round -> t
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> Mpfr.round -> t
val of_mpq : 'a Mpq.tt -> Mpfr.round -> t
val to_string : t -> string
val to_float : ?round:Mpfr.round -> t -> float
val to_mpqf : t -> Mpqf.t
val add : 'a tt -> 'b tt -> Mpfr.round -> t
val add_int : 'a tt -> int -> Mpfr.round -> t
val sub : 'a tt -> 'b tt -> Mpfr.round -> t
val sub_int : 'a tt -> int -> Mpfr.round -> t
```

```
val mul : 'a tt -> 'b tt -> Mpfr.round -> t
val mul_ui : 'a tt -> int -> Mpfr.round -> t
val ui_div : int -> 'b tt -> Mpfr.round -> t
val div : 'a tt -> 'b tt -> Mpfr.round -> t
val div_ui : 'a tt -> int -> Mpfr.round -> t
val sqrt : 'a tt -> Mpfr.round -> t
val ui_pow : int -> 'b tt -> Mpfr.round -> t
val pow : 'a tt -> 'b tt -> Mpfr.round -> t
val pow_int : 'a tt -> int -> Mpfr.round -> t
val neg : 'a tt -> Mpfr.round -> t
val abs : 'a tt -> Mpfr.round -> t
val equal : 'a tt -> 'b tt -> bits:int -> bool
val cmp : 'a tt -> 'b tt -> int
val cmp_int : 'a tt -> int -> int
val sgn : 'a tt -> int
val nan_p : 'a tt -> bool
val inf_p : 'a tt -> bool
val number_p : 'a tt -> bool
```

# Index

`_equal`, 12, 17  
`_export`, 7  
`_gcdext`, 6  
`_get_str`, 4, 9, 12, 15  
`_import`, 7  
`_init_set_str`, 4, 11, 15  
`_mpfr`, 24  
`_mpfrf`, 24  
`_mpq`, 22  
`_mpqf`, 22  
`_mpz`, 20  
`_mpzf`, 20  
`_powm`, 6  
`_powm_ui`, 6  
`_set_str`, 4, 9, 11, 15  
`_sqrtrem`, 6  
`_strtofr`, 15  
  
`abs`, 5, 10, 12, 16, 21, 22, 25  
`acos`, 17  
`acosh`, 17  
`add`, 5, 10, 12, 16, 20, 22, 24  
`add_int`, 20, 24  
`add_q`, 16  
`add_ui`, 5, 12, 16  
`add_z`, 16  
`addmul`, 5  
`addmul_ui`, 5  
`agm`, 18  
`asin`, 17  
`asinh`, 17  
`atan`, 17  
`atan2`, 17  
`atanh`, 17  
  
`bin_ui`, 7  
`bin_uiui`, 7  
  
`canonicalize`, 9  
`cdiv_q`, 5, 20  
`cdiv_q_2exp`, 5  
`cdiv_q_ui`, 5  
`cdiv_qr`, 5, 20  
`cdiv_qr_ui`, 5  
`cdiv_r`, 5, 20  
`cdiv_r_2exp`, 5  
`cdiv_r_ui`, 5  
  
`cdiv_ui`, 5  
`ceil`, 12, 18  
`check_range`, 14  
`clear_flags`, 15  
`clear_inexflag`, 15  
`clear_nanflag`, 15  
`clear_overflow`, 15  
`clear_underflow`, 14  
`clrbits`, 7  
`cmp`, 7, 10, 12, 16, 21, 23, 25  
`cmp_d`, 7, 12  
`cmp_frac`, 23  
`cmp_int`, 21, 23, 25  
`cmp_si`, 7, 10, 12, 16  
`cmp_si_2exp`, 16  
`cmpabs`, 7  
`cmpabs_d`, 7  
`cmpabs_ui`, 7  
`com`, 7  
`congruent_2exp_p`, 6  
`congruent_p`, 6  
`congruent_ui_p`, 6  
`const_catalan`, 18  
`const_euler`, 18  
`const_log2`, 18  
`const_pi`, 18  
`cos`, 17  
`cosh`, 17  
`cot`, 17  
`coth`, 17  
`csc`, 17  
`csch`, 17  
  
`div`, 10, 12, 16, 22, 25  
`div_2exp`, 10, 12, 16  
`div_2si`, 16  
`div_2ui`, 16  
`div_q`, 16  
`div_ui`, 12, 16, 25  
`div_z`, 16  
`divexact`, 6, 21  
`divexact_ui`, 6  
`divisible_2exp_p`, 6  
`divisible_p`, 6  
`divisible_ui_p`, 6  
  
`eint`, 17

equal, 10, 12, 17, 23, 25  
 erf, 17  
 erfc, 17  
 even\_p, 8  
 exp, 17  
 exp10, 17  
 exp2, 17  
 expm1, 17  
 export, 7

f, 4, 9, 11, 14  
 fac\_ui, 7, 17  
 fdiv\_q, 5, 20  
 fdiv\_q\_2exp, 6  
 fdiv\_q\_ui, 5  
 fdiv\_qr, 5, 20  
 fdiv\_qr\_ui, 6  
 fdiv\_r, 5, 20  
 fdiv\_r\_2exp, 6  
 fdiv\_r\_ui, 6  
 fdiv\_ui, 6  
 fib\_ui, 7  
 fib2\_ui, 7  
 fits\_int\_p, 7, 12  
 fits\_sint\_p, 8, 13  
 fits\_slong\_p, 8, 13  
 fits\_sshort\_p, 8, 13  
 fits\_uint\_p, 8, 13  
 fits\_ulong\_p, 8, 12  
 fits\_ushort\_p, 8, 13  
 floor, 12, 18  
 fma, 18  
 fmod, 18  
 fms, 18  
 frac, 18

gamma, 17  
 gand, 7  
 gcd, 6, 21  
 gcd\_ui, 6  
 gcdext, 6  
 get\_d, 4, 9, 12, 15  
 get\_d\_2exp, 4, 12  
 get\_d1, 15  
 get\_default\_prec, 11, 15  
 get\_default\_rounding\_mode, 14  
 get\_den, 10, 23  
 get\_emax, 14  
 get\_emin, 14  
 get\_exp, 18  
 get\_int, 4, 12  
 get\_num, 10, 23  
 get\_prec, 11, 15  
 get\_q, 12  
 get\_si, 4, 12  
 get\_str, 4, 9, 12, 15

get\_z, 9, 12, 15  
 get\_z\_exp, 15  
 gmod, 6, 21  
 gmod\_ui, 6  
 Gmp\_random, 19

hamdist, 7  
 hypot, 18

import, 7  
 inexflag\_p, 15  
 inf\_p, 17, 25  
 init, 4, 9, 11, 15  
 init\_default, 19  
 init\_lc\_2exp, 19  
 init\_lc\_2exp\_size, 19  
 init\_set, 4, 9, 11, 15  
 init\_set\_d, 4, 9, 11, 15  
 init\_set\_f, 15  
 init\_set\_q, 15  
 init\_set\_si, 4, 9, 11, 15  
 init\_set\_str, 4, 9, 11, 15  
 init\_set\_z, 9, 15  
 init2, 4, 11, 15  
 integer\_p, 12, 18  
 Introduction, 3  
 inv, 10, 22  
 invert, 6  
 ior, 7  
 is\_integer, 12

j0, 17  
 j1, 17  
 jacobi, 7  
 jn, 17

kronecker, 7  
 kronecker\_si, 7

lcm, 6, 21  
 lcm\_ui, 6  
 legendre, 7  
 lngamma, 17  
 log, 17  
 log10, 17  
 log1p, 17  
 log2, 17  
 lucnum\_ui, 7  
 lucnum2\_ui, 7

m, 4, 9, 11, 14  
 max, 18  
 min, 18  
 modf, 18  
 Mpf, 11, 19  
 Mpfr, 14, 19  
 Mpfrf, 24

---

Mpq, 9  
 Mpqf, 22  
 Mpz, 4, 19  
 Mpzf, 20  
 mul, 5, 10, 12, 16, 20, 22, 25  
 mul\_2exp, 5, 10, 12, 16  
 mul\_2si, 16  
 mul\_2ui, 16  
 mul\_int, 20  
 mul\_q, 16  
 mul\_si, 5  
 mul\_ui, 12, 16, 25  
 mul\_z, 16  
  
 nan\_p, 17, 25  
 nanflag\_p, 15  
 neg, 5, 10, 12, 16, 21, 22, 25  
 nextabove, 18  
 nextbelow, 18  
 nextprime, 6  
 nexttoward, 18  
 number\_p, 17, 25  
  
 odd\_p, 8  
 of\_float, 5, 10, 12, 16, 20, 22, 24  
 of\_frac, 10, 16, 22, 24  
 of\_int, 5, 10, 12, 16, 20, 22, 24  
 of\_mpfr, 24  
 of\_mpq, 12, 16, 22, 24  
 of\_mpz, 10, 12, 16, 20, 22, 24  
 of\_mpz2, 10, 16, 22, 24  
 of\_string, 5, 10, 12, 16, 20, 22, 24  
 overflow\_p, 15  
  
 perfect\_power\_p, 6  
 perfect\_square\_p, 6  
 popcount, 7  
 pow, 16, 25  
 pow\_int, 25  
 pow\_si, 16  
 pow\_ui, 6, 12, 16  
 powm, 6  
 powm\_ui, 6  
 print, 4, 9, 11, 14, 20, 22, 24  
 print\_round, 14  
 probab\_prime\_p, 6  
  
 realloc2, 4  
 reldiff, 12, 17  
 remainder, 18  
 remove, 7  
 rint, 18  
 root, 6  
 round, 14, 18  
 round\_prec, 14  
 rrandomb, 19  
  
 scan0, 7  
 scan1, 7  
 sec, 17  
 sech, 17  
 seed, 19  
 seed\_ui, 19  
 set, 4, 9, 11, 15  
 set\_d, 4, 9, 11, 15  
 set\_default\_prec, 11, 15  
 set\_default\_rounding\_mode, 14  
 set\_den, 10  
 set\_emax, 14  
 set\_emin, 14  
 set\_exp, 18  
 set\_f, 15  
 set\_inf, 15  
 set\_nan, 15  
 set\_num, 10  
 set\_prec, 11, 15  
 set\_prec\_raw, 11, 15  
 set\_q, 11, 15  
 set\_si, 4, 9, 11, 15  
 set\_si\_2exp, 15  
 set\_str, 4, 9, 11, 15  
 set\_z, 9, 11, 15  
 setbit, 7  
 sgn, 7, 10, 12, 16, 21, 23, 25  
 si\_kronecker, 7  
 sin, 17  
 sin\_cos, 17  
 sinh, 17  
 size, 8  
 sizeinbase, 8  
 sqrt, 6, 12, 16, 25  
 sqrt\_ui, 16  
 sqrtrem, 6  
 state, 19  
 string\_of\_round, 14  
 strtouf, 15  
 sub, 5, 10, 12, 16, 20, 22, 24  
 sub\_int, 20, 24  
 sub\_q, 16  
 sub\_ui, 5, 12, 16  
 sub\_z, 16  
 submul, 5  
 submul\_ui, 5  
 swap, 4, 9, 11, 15  
  
 t, 4, 9, 11, 14, 20, 22, 24  
 tan, 17  
 tanh, 17  
 tdiv\_q, 6, 20  
 tdiv\_q\_2exp, 6  
 tdiv\_q\_ui, 6  
 tdiv\_qr, 6, 21  
 tdiv\_qr\_ui, 6

---

tdiv\_r, 6, 20  
tdiv\_r\_2exp, 6  
tdiv\_r\_ui, 6  
tdiv\_ui, 6  
to\_float, 5, 9, 12, 15, 20, 22, 24  
to\_mpfr, 24  
to\_mpq, 16, 22  
to\_mpqf, 24  
to\_mpz, 20  
to\_mpzf2, 22  
to\_string, 5, 9, 12, 15, 20, 22, 24  
trunc, 12, 18  
tstbit, 7  
tt, 4, 9, 11, 14, 20, 22, 24  
  
ui\_div, 12, 16, 25  
ui\_pow, 16, 25  
ui\_pow\_ui, 6, 16  
ui\_sub, 5, 12, 16  
underflow\_p, 15  
urandom, 19  
urandomb, 19  
urandomm, 19  
  
xor, 7  
  
y0, 17  
y1, 18  
yn, 18  
  
zeta, 17