GNU Multiple Precision

Introduction

These functions allow you to work with arbitrary-length integers using the GNU MP library.

These functions have been added in PHP 4.0.4.

Note
Most GMP functions accept GMP number arguments, defined as *resource* below. However, most of these functions will also accept numeric and string arguments, given that it is possible to convert the latter to a number. Also, if there is a faster function that can operate on integer arguments, it would be used instead of the slower function when the supplied arguments are integers. This is done transparently, so the bottom line is that you can use integers in every function that expects GMP number. See also the gmp_init() function.

Warning
If you want to explicitly specify a large integer, specify it as a string. If you don't do that, PHP will interpret the integer-literal first, possibly resulting in loss of precision, even before *GMP* comes into play.

Note
This extension is available on Windows platforms since PHP 5.1.0.

Installing/Configuring

Requirements

You can download the GMP library from » http://www.swox.com/gmp/. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

Installation

In order to have these functions available, you must compile PHP with GMP support by using the *--with-gmp* option.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

Most GPM functions operate on or return GMP number resources.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

GMP_ROUND_ZERO (integer)

GMP_ROUND_PLUSINF (integer)

GMP_ROUND_MINUSINF (integer)

GMP_VERSION (string)
The GMP library version

Examples

Example #1 - Factorial function using GMP

```
<?php
function fact($x)
{
   $return = 1;
   for ($i=2; $i < $x; $i++) {
       $return = gmp_mul($return, $i);
   }
   return $return;
}

echo gmp_strval(fact(1000)) . "\n";
?>
```

This will calculate factorial of 1000 (pretty big number) very fast.

GMP Functions

See Also

More mathematical functions can be found in the Mathematical Extensions section

gmp_abs

gmp_abs -- Absolute value

Description

resource **gmp_abs** (resource $a)

Get the absolute value of a number.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Returns the absolute value of *a*, as a GMP number.

Examples

Example #2 - gmp_abs() example

```
<?php
$abs1 = gmp_abs("274982683358");
$abs2 = gmp_abs("-274982683358");

echo gmp_strval($abs1) . "\n";
echo gmp_strval($abs2) . "\n";
?>
```

The above example will output:

```
274982683358
274982683358
```

gmp_add

gmp_add -- Add numbers

Description

resource **gmp_add** (resource `$a`, resource `$b`)

Add two numbers.

Parameters

a
: A number that will be added. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b
: A number that will be added. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number representing the sum of the arguments.

Examples

Example #3 - gmp_add() example

```
<?php
$sum = gmp_add("123456789012345", "76543210987655");
echo gmp_strval($sum) . "\n";
?>
```

The above example will output:

```
200000000000000
```

gmp_and

gmp_and -- Bitwise AND

Description

resource **gmp_and** (resource `$a`, resource `$b`)

Calculates bitwise AND of two GMP numbers.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number representing the bitwise *AND* comparison.

Examples

Example #4 - gmp_and() example

```
<?php
$and1 = gmp_and("0xfffffffff4", "0x4");
$and2 = gmp_and("0xfffffffff4", "0x8");
echo gmp_strval($and1) . "\n";
echo gmp_strval($and2) . "\n";
?>
```

The above example will output:

```
4
0
```

gmp_clrbit

gmp_clrbit -- Clear bit

Description

void **gmp_clrbit** (resource `&$a`, int `$index`)

Clears (sets to 0) bit *`index`* in *`a`*. The index starts at 0.

Parameters

`a`
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`index`
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #5 - gmp_clrbit() example

```
<?php
$a = gmp_init("0xff");
gmp_clrbit($a, 0); // index starts at 0, least significant bit
echo gmp_strval($a) . "\n";
?>
```

The above example will output:

```
254
```

Notes

Note

Unlike most of the other GMP functions, gmp_clrbit() must be called with a GMP resource that already exists (using gmp_init() for example). One will not be

automatically created.

See Also

- gmp_setbit()
- gmp_testbit()

gmp_cmp

gmp_cmp -- Compare numbers

Description

int **gmp_cmp** (resource `$a`, resource `$b`)

Compares two numbers.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Returns a positive value if *a* > *b*, zero if *a* = *b* and a negative value if *a* < *b*.

Examples

Example #6 - gmp_cmp() example

```
<?php
$cmp1 = gmp_cmp("1234", "1000"); // greater than
$cmp2 = gmp_cmp("1000", "1234"); // less than
$cmp3 = gmp_cmp("1234", "1234"); // equal to

echo "$cmp1 $cmp2 $cmp3\n";
?>
```

The above example will output:

```
1 -1 0
```

gmp_com

gmp_com -- Calculates one's complement

Description

resource **gmp_com** (resource `$a`)

Returns the one's complement of *a*.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Returns the one's complement of *a*, as a GMP number.

Examples

Example #7 - gmp_com() example

```
<?php
$com = gmp_com("1234");
echo gmp_strval($com) . "\n";
?>
```

The above example will output:

```
-1235
```

gmp_div_q

gmp_div_q -- Divide numbers

Description

resource **gmp_div_q** (resource `$a`, resource `$b` [, int `$round`])

Divides *a* by *b* and returns the integer result.

Parameters

a
: The number being divided. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b
: The number that *a* is being divided by. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

round
: The result rounding is defined by the *round*, which can have the following values:

  - **GMP_ROUND_ZERO**: The result is truncated towards 0.
  - **GMP_ROUND_PLUSINF**: The result is rounded towards +*infinity*.
  - **GMP_ROUND_MINUSINF**: The result is rounded towards -*infinity*.

  It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #8 - gmp_div_q() example

```
<?php
$div1 = gmp_div_q("100", "5");
echo gmp_strval($div1) . "\n";

$div2 = gmp_div_q("1", "3");
echo gmp_strval($div2) . "\n";
```

```
$div3 = gmp_div_q("1", "3", GMP_ROUND_PLUSINF);
echo gmp_strval($div3) . "\n";

$div4 = gmp_div_q("-1", "4", GMP_ROUND_PLUSINF);
echo gmp_strval($div4) . "\n";

$div5 = gmp_div_q("-1", "4", GMP_ROUND_MINUSINF);
echo gmp_strval($div5) . "\n";
?>
```

The above example will output:

```
20
0
1
0
-1
```

Notes

Note
This function can also be called as gmp_div().

See Also

- gmp_div_r()
- gmp_div_qr()

gmp_div_qr

gmp_div_qr -- Divide numbers and get quotient and remainder

Description

array **gmp_div_qr** (resource `$n`, resource `$d` [, int `$round`])

The function divides *n* by *d*.

Parameters

n
: The number being divided. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d
: The number that *n* is being divided by. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

round
: See the gmp_div_q() function for description of the *round* argument.

Return Values

Returns an array, with the first element being *[n/d]* (the integer result of the division) and the second being *(n - [n/d] * d)* (the remainder of the division).

Examples

Example #9 - Division of GMP numbers

```
<?php
$a = gmp_init("0x41682179fbf5");
$res = gmp_div_qr($a, "0xDEFE75");
printf("Result is: q - %s, r - %s",
      gmp_strval($res[0]), gmp_strval($res[1]));
?>
```

See Also

- gmp_div_q()
- gmp_div_r()

gmp_div_r

gmp_div_r -- Remainder of the division of numbers

Description

resource **gmp_div_r** (resource `$n`, resource `$d` [, int `$round`])

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

Parameters

n
: The number being divided. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d
: The number that *n* is being divided by. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

round
: See the gmp_div_q() function for description of the *round* argument.

Return Values

The remainder, as a GMP number.

Examples

Example #10 - gmp_div_r() example

```
<?php
$div = gmp_div_r("105", "20");
echo gmp_strval($div) . "\n";
?>
```

The above example will output:

```
5
```

See Also

- gmp_div_q()
- gmp_div_qr()

gmp_div

gmp_div -- Alias of gmp_div_q()

Description

This function is an alias of: gmp_div_q().

gmp_divexact

gmp_divexact -- Exact division of numbers

Description

resource **gmp_divexact** (resource $n, resource $d)

Divides *n* by *d*, using fast "exact division" algorithm. This function produces correct results only when it is known in advance that *d* divides *n*.

Parameters

n
: The number being divided. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d
: The number that *a* is being divided by. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #11 - gmp_divexact() example

```
<?php
$div1 = gmp_divexact("10", "2");
echo gmp_strval($div1) . "\n";

$div2 = gmp_divexact("10", "3"); // bogus result
echo gmp_strval($div2) . "\n";
?>
```

The above example will output:

```
5
2863311534
```

gmp_fact

gmp_fact -- Factorial

Description

resource **gmp_fact** (int $a)

Calculates factorial (*a!*) of *a*.

Parameters

a

The factorial number. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #12 - gmp_fact() example

```
<?php
$fact1 = gmp_fact(5); // 5 * 4 * 3 * 2 * 1
echo gmp_strval($fact1) . "\n";

$fact2 = gmp_fact(50); // 50 * 49 * 48, ... etc
echo gmp_strval($fact2) . "\n";
?>
```

The above example will output:

```
120
30414093201713378043612608166064768844377641568960512000000000000
```

gmp_gcd

gmp_gcd -- Calculate GCD

Description

resource **gmp_gcd** (resource `$a`, resource `$b`)

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of, or both, input operands are negative.

Parameters

a
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A positive GMP number that divides into both *a* and *b*.

Examples

Example #13 - gmp_gcd() example

```
<?php
$gcd = gmp_gcd("12", "21");
echo gmp_strval($gcd) . "\n";
?>
```

The above example will output:

```
3
```

gmp_gcdext

gmp_gcdext -- Calculate GCD and multipliers

Description

array **gmp_gcdext** (resource `$a`, resource `$b`)

Calculates g, s, and t, such that *a*s + b*t = g = gcd(a,b)*, where gcd is the greatest common divisor. Returns an array with respective elements g, s and t.

This function can be used to solve linear Diophantine equations in two variables. These are equations that allow only integer solutions and have the form: *a*x + b*y = c*. For more information, go to the » "Diophantine Equation" page at MathWorld

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

An array of GMP numbers.

Examples

Example #14 - Solving a linear Diophantine equation

```
<?php
// Solve the equation a*s + b*t = g
// where a = 12, b = 21, g = gcd(12, 21) = 3
$a = gmp_init(12);
$b = gmp_init(21);
$g = gmp_gcd($a, $b);
$r = gmp_gcdext($a, $b);

$check_gcd = (gmp_strval($g) == gmp_strval($r['g']));
$eq_res = gmp_add(gmp_mul($a, $r['s']), gmp_mul($b, $r['t']));
$check_res = (gmp_strval($g) == gmp_strval($eq_res));

if ($check_gcd && $check_res) {
```

```
   $fmt = "Solution: %d*%d + %d*%d = %d\n";
   printf($fmt, gmp_strval($a), gmp_strval($r['s']), gmp_strval($b),
   gmp_strval($r['t']), gmp_strval($r['g']));
} else {
   echo "Error while solving the equation\n";
}

// output: Solution: 12*2 + 21*-1 = 3
?>
```

gmp_hamdist

gmp_hamdist -- Hamming distance

Description

int **gmp_hamdist** (resource $a, resource $b)

Returns the hamming distance between *a* and *b*. Both operands should be non-negative.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number. It should be positive.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number. It should be positive.

Return Values

A GMP number resource.

Examples

Example #15 - gmp_hamdist() example

```
<?php
$ham1 = gmp_init("1001010011", 2);
$ham2 = gmp_init("1011111100", 2);
echo gmp_hamdist($ham1, $ham2) . "\n";

/* hamdist is equivilent to: */
echo gmp_popcount(gmp_xor($ham1, $ham2)) . "\n";
?>
```

The above example will output:

```
6
6
```

See Also

- gmp_popcount()
- gmp_xor()

gmp_init

gmp_init -- Create GMP number

Description

resource **gmp_init** (mixed `$number` [, int `$base`])

Creates a GMP number from an integer or string.

Parameters

`number`
: An integer or a string. The string representation can be decimal, hexadecimal or octal.

`base`
: The base. Defaults to 0. The base may vary from 2 to 36. If base is 0 (default value), the actual base is determined from the leading characters: if the first two characters are *0x* or *0X*, hexadecimal is assumed, otherwise if the first character is "0", octal is assumed, otherwise decimal is assumed.

Return Values

A GMP number resource.

ChangeLog

Version	Description
4.1.0	The optional *`base`* parameter was added.

Examples

Example #16 - Creating GMP number

```
<?php
$a = gmp_init(123456);
$b = gmp_init("0xFFFFDEBACDFEDF7200");
?>
```

Notes

Note
It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like gmp_add(). Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as gmp_init().

gmp_intval

gmp_intval -- Convert GMP number to integer

Description

int **gmp_intval** (resource `$gmpnumber`)

This function allows to convert GMP number to integer.

Parameters

`gmpnumber`
A GMP number.

Return Values

An integer value of *`gmpnumber`*.

Examples

Example #17 - gmp_intval() example

```
<?php
// displays correct result
echo gmp_intval("2147483647") . "\n";

// displays wrong result, above PHP integer limit
echo gmp_intval("2147483648") . "\n";

// displays correct result
echo gmp_strval("2147483648") . "\n";
?>
```

The above example will output:

```
2147483647
2147483647
2147483648
```

Notes

Warning

This function returns a useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use gmp_strval().

gmp_invert

gmp_invert -- Inverse by modulo

Description

resource **gmp_invert** (resource `$a`, resource `$b`)

Computes the inverse of *a* modulo *b*.

Parameters

a
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number on success or **FALSE** if an inverse does not exist.

Examples

Example #18 - gmp_invert() example

```
<?php
echo gmp_invert("5", "10"); // no inverse, outputs nothing, result is FALSE
$invert = gmp_invert("5", "11");
echo gmp_strval($invert) . "\n";
?>
```

The above example will output:

```
9
```

gmp_jacobi

gmp_jacobi -- Jacobi symbol

Description

int **gmp_jacobi** (resource $a, resource $p)

Computes » Jacobi symbol of *a* and *p*. *p* should be odd and must be positive.

Parameters

a
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

p
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number. Should be odd and must be positive.

Return Values

A GMP number resource.

Examples

Example #19 - gmp_jacobi() example

```
<?php
echo gmp_jacobi("1", "3") . "\n";
echo gmp_jacobi("2", "3") . "\n";
?>
```

The above example will output:

```
1
0
```

gmp_legendre

gmp_legendre -- Legendre symbol

Description

int **gmp_legendre** (resource `$a`, resource `$p`)

Compute the » Legendre symbol of *a* and *p*. *p* should be odd and must be positive.

Parameters

a
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

p
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number. Should be odd and must be positive.

Return Values

A GMP number resource.

Examples

Example #20 - gmp_legendre() example

```
<?php
echo gmp_legendre("1", "3") . "\n";
echo gmp_legendre("2", "3") . "\n";
?>
```

The above example will output:

```
1
0
```

gmp_mod

gmp_mod -- Modulo operation

Description

resource **gmp_mod** (resource $n, resource $d)

Calculates *n* modulo *d*. The result is always non-negative, the sign of *d* is ignored.

Parameters

n

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d

The modulo that is being evaluated. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #21 - gmp_mod() example

```
<?php
$mod = gmp_mod("8", "3");
echo gmp_strval($mod) . "\n";
?>
```

The above example will output:

```
2
```

gmp_mul

gmp_mul -- Multiply numbers

Description

resource **gmp_mul** (resource $a, resource $b)

Multiplies *a* by *b* and returns the result.

Parameters

a

A number that will be multiplied by *b*. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

A number that will be multiplied by *a*. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #22 - gmp_mul() example

```
<?php
$mul = gmp_mul("12345678", "2000");
echo gmp_strval($mul) . "\n";
?>
```

The above example will output:

```
24691356000
```

gmp_neg

gmp_neg -- Negate number

Description

resource **gmp_neg** (resource `$a`)

Returns the negative value of a number.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Returns - *a*, as a GMP number.

Examples

Example #23 - gmp_neg() example

```
<?php
$neg1 = gmp_neg("1");
echo gmp_strval($neg1) . "\n";
$neg2 = gmp_neg("-1");
echo gmp_strval($neg2) . "\n";
?>
```

The above example will output:

```
-1
1
```

gmp_nextprime

gmp_nextprime -- Find next prime number

Description

resource **gmp_nextprime** (int `$a`)

Find next prime number

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Return the next prime number greater than *a*, as a GMP number.

Examples

Example #24 - gmp_nextprime() example

```
<?php
$prime1 = gmp_nextprime(10); // next prime number greater than 10
$prime2 = gmp_nextprime(-1000); // next prime number greater than -1000

echo gmp_strval($prime1) . "\n";
echo gmp_strval($prime2) . "\n";
?>
```

The above example will output:

```
11
-997
```

Notes

Note

This function uses a probabilistic algorithm to identify primes and chances to get a composite number are extremely small.

gmp_or

gmp_or -- Bitwise OR

Description

resource **gmp_or** (resource `$a`, resource `$b`)

Calculates bitwise inclusive OR of two GMP numbers.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #25 - gmp_or() example

```
<?php
$or1 = gmp_or("0xfffffff2", "4");
echo gmp_strval($or1, 16) . "\n";
$or2 = gmp_or("0xfffffff2", "2");
echo gmp_strval($or2, 16) . "\n";
?>
```

The above example will output:

```
fffffff6
fffffff2
```

gmp_perfect_square

gmp_perfect_square -- Perfect square check

Description

bool **gmp_perfect_square** (resource $a)

Check if a number is a perfect square.

Parameters

a

The number being checked as a perfect square. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Returns **TRUE** if *a* is a perfect square, **FALSE** otherwise.

Examples

Example #26 - gmp_perfect_square() example

```
<?php
// 3 * 3, perfect square
var_dump(gmp_perfect_square("9"));

// not a perfect square
var_dump(gmp_perfect_square("7"));

// 1234567890 * 1234567890, perfect square
var_dump(gmp_perfect_square("152415787501905210 0"));
?>
```

The above example will output:

```
bool(true)
bool(false)
bool(true)
```

See Also

- gmp_sqrt()
- gmp_sqrtrem()

gmp_popcount

gmp_popcount -- Population count

Description

int **gmp_popcount** (resource $a)

Get the population count.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

The population count of *a*, as an integer.

Examples

Example #27 - gmp_popcount() example

```
<?php
$pop1 = gmp_init("10000101", 2); // 3 1's
echo gmp_popcount($pop1) . "\n";
$pop2 = gmp_init("11111110", 2); // 7 1's
echo gmp_popcount($pop2) . "\n";
?>
```

The above example will output:

```
3
7
```

gmp_pow

gmp_pow -- Raise number into power

Description

resource **gmp_pow** (resource `$base`, int `$exp`)

Raise *`base`* into power *`exp`*.

Parameters

`base`
: The base number. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`exp`
: The positive power to raise the *`base`*.

Return Values

The new (raised) number, as a GMP number. The case of *0^0* yields 1.

Examples

Example #28 - gmp_pow() example

```
<?php
$pow1 = gmp_pow("2", 31);
echo gmp_strval($pow1) . "\n";
$pow2 = gmp_pow("0", 0);
echo gmp_strval($pow2) . "\n";
$pow3 = gmp_pow("2", -1); // Negative exp, generates warning
echo gmp_strval($pow3) . "\n";
?>
```

The above example will output:

```
2147483648
1
```

gmp_powm

gmp_powm -- Raise number into power with modulo

Description

resource **gmp_powm** (resource `$base`, resource `$exp`, resource `$mod`)

Calculate (*`base`* raised into power *`exp`*) modulo *`mod`*. If *`exp`* is negative, result is undefined.

Parameters

`base`
: The base number. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`exp`
: The positive power to raise the *`base`*. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`mod`
: The modulo. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

The new (raised) number, as a GMP number.

Examples

Example #29 - gmp_powm() example

```
<?php
$pow1 = gmp_powm("2", "31", "2147483649");
echo gmp_strval($pow1) . "\n";
?>
```

The above example will output:

```
2147483648
```

gmp_prob_prime

gmp_prob_prime -- Check if number is "probably prime"

Description

int **gmp_prob_prime** (resource `$a` [, int `$reps`])

The function uses Miller-Rabin's probabilistic test to check if a number is a prime.

Parameters

a
: The number being checked as a prime. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

reps
: Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable" prime. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably" prime. If it returns 2, then *a* is surely prime.

Examples

Example #30 - gmp_prob_prime() example

```
<?php
// definitely not a prime
echo gmp_prob_prime("6") . "\n";

// probably a prime
echo gmp_prob_prime("11111111111111111") . "\n";

// definitely a prime
echo gmp_prob_prime("11") . "\n";
?>
```

The above example will output:

```
0
1
```

2

gmp_random

gmp_random -- Random number

Description

resource **gmp_random** (int `$limiter`)

Generate a random number. The number will be between zero and the number of bits per limb multiplied by *`limiter`*. If *`limiter`* is negative, negative numbers are generated.

A limb is an internal GMP mechanism. The number of bits in a limb is not static, and can vary from system to system. Generally, the number of bits in a limb is either 16 or 32, but this is not guaranteed.

Parameters

`limiter`
: The limiter. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A random GMP number.

Examples

Example #31 - gmp_random() example

```
<?php
$rand1 = gmp_random(1); // random number from 0 to 1 * bits per limb
$rand2 = gmp_random(2); // random number from 0 to 2 * bits per limb

echo gmp_strval($rand1) . "\n";
echo gmp_strval($rand2) . "\n";
?>
```

The above example will output:

```
1915834968
8642564075890328087
```

gmp_scan0

gmp_scan0 -- Scan for 0

Description

int **gmp_scan0** (resource `$a`, int `$start`)

Scans *`a`*, starting with bit *`start`*, towards more significant bits, until the first clear bit is found.

Parameters

`a`
: The number to scan. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`start`
: The starting bit.

Return Values

Returns the index of the found bit, as an integer. The index starts from 0.

Examples

Example #32 - gmp_scan0() example

```
<?php
// "0" bit is found at position 3. index starts at 0
$s1 = gmp_init("10111", 2);
echo gmp_scan0($s1, 0) . "\n";

// "0" bit is found at position 7. index starts at 5
$s2 = gmp_init("101110000", 2);
echo gmp_scan0($s2, 5) . "\n";
?>
```

The above example will output:

```
3
7
```

gmp_scan1

gmp_scan1 -- Scan for 1

Description

int **gmp_scan1** (resource `$a`, int `$start`)

Scans *`a`*, starting with bit *`start`*, towards more significant bits, until the first set bit is found.

Parameters

`a`
: The number to scan. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`start`
: The starting bit.

Return Values

Returns the index of the found bit, as an integer. If no set bit is found, -1 is returned.

Examples

Example #33 - gmp_scan1() example

```
<?php
// "1" bit is found at position 3. index starts at 0
$s1 = gmp_init("01000", 2);
echo gmp_scan1($s1, 0) . "\n";

// "1" bit is found at position 9. index starts at 5
$s2 = gmp_init("01000001111", 2);
echo gmp_scan1($s2, 5) . "\n";
?>
```

The above example will output:

```
3
9
```

gmp_setbit

gmp_setbit -- Set bit

Description

void **gmp_setbit** (resource `&$a`, int `$index` [, bool `$set_clear`])

Sets bit *`index`* in *`a`*.

Parameters

`a`
: The number being set to. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`index`
: The set bit.

`set_clear`
: Defines if the bit is set to 0 or 1. By default the bit is set to 1. Index starts at 0.

Return Values

A GMP number resource.

Examples

Example #34 - gmp_setbit() example

```
<?php
$a = gmp_init("0xfd");
gmp_setbit($a, 1); // index starts at 0
echo gmp_strval($a) . "\n";
?>
```

The above example will output:

```
255
```

Notes

Note
Unlike most of the other GMP functions, gmp_setbit() must be called with a GMP resource that already exists (using gmp_init() for example). One will not be automatically created.

See Also

- gmp_clrbit()
- gmp_testbit()

gmp_sign

gmp_sign -- Sign of number

Description

int **gmp_sign** (resource $a)

Checks the sign of a number.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Returns 1 if *a* is positive, -1 if *a* is negative, and 0 if *a* is zero.

Examples

Example #35 - gmp_sign() example

```
<?php
// positive
echo gmp_sign("500") . "\n";

// negative
echo gmp_sign("-500") . "\n";

// zero
echo gmp_sign("0") . "\n";
?>
```

The above example will output:

```
1
-1
0
```

gmp_sqrt

gmp_sqrt -- Calculate square root

Description

resource **gmp_sqrt** (resource `$a`)

Calculates square root of *a*.

Parameters

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

The integer portion of the square root, as a GMP number.

Examples

Example #36 - gmp_sqrt() example

```
<?php
$sqrt1 = gmp_sqrt("9");
$sqrt2 = gmp_sqrt("7");
$sqrt3 = gmp_sqrt("1524157875019052100");

echo gmp_strval($sqrt1) . "\n";
echo gmp_strval($sqrt2) . "\n";
echo gmp_strval($sqrt3) . "\n";
?>
```

The above example will output:

```
3
2
1234567890
```

gmp_sqrtrem

gmp_sqrtrem -- Square root with remainder

Description

array **gmp_sqrtrem** (resource $a)

Calculate the square root of a number, with remainder.

Parameters

a

The number being square rooted. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

Returns array where first element is the integer square root of *a* and the second is the remainder (i.e., the difference between *a* and the first element squared).

Examples

Example #37 - gmp_sqrtrem() example

```
<?php
list($sqrt1, $sqrt1rem) = gmp_sqrtrem("9");
list($sqrt2, $sqrt2rem) = gmp_sqrtrem("7");
list($sqrt3, $sqrt3rem) = gmp_sqrtrem("1048576");

echo gmp_strval($sqrt1) . ", " . gmp_strval($sqrt1rem) . "\n";
echo gmp_strval($sqrt2) . ", " . gmp_strval($sqrt2rem) . "\n";
echo gmp_strval($sqrt3) . ", " . gmp_strval($sqrt3rem) . "\n";
?>
```

The above example will output:

```
3, 0
2, 3
1024, 0
```

gmp_strval

gmp_strval -- Convert GMP number to string

Description

string **gmp_strval** (resource `$gmpnumber` [, int `$base`])

Convert GMP number to string representation in base *`base`*. The default base is 10.

Parameters

`gmpnumber`
: The GMP number that will be converted to a string. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

`base`
: The base of the returned number. The default base is 10. Allowed values for the base are from 2 to 36.

Return Values

The number, as a string.

Examples

Example #38 - Converting a GMP number to a string

```
<?php
$a = gmp_init("0x41682179fbf5");
printf("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_sub

gmp_sub -- Subtract numbers

Description

resource **gmp_sub** (resource `$a`, resource `$b`)

Subtracts *b* from *a* and returns the result.

Parameters

a

The number being subtracted from. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

The number subtracted from *a*. It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #39 - gmp_sub() example

```
<?php
$sub = gmp_sub("281474976710656", "4294967296"); // 2^48 - 2^32
echo gmp_strval($sub) . "\n";
?>
```

The above example will output:

```
281470681743360
```

gmp_testbit

gmp_testbit -- Tests if a bit is set

Description

bool **gmp_testbit** (resource `$a`, int `$index`)

Tests if the specified bit is set.

Parameters

a
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

index
: The bit to test

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Errors/Exceptions

E_WARNING is issued when *index* is less than zero.

Examples

Example #40 - gmp_testbit() example

```
<?php
$n = gmp_init("1000000");
var_dump(gmp_testbit($n, 1));
gmp_setbit($n, 1);
var_dump(gmp_testbit($n, 1));
?>
```

The above example will output:

```
bool(false)
bool(true)
```

See Also

- gmp_setbit()
- gmp_clrbit()

gmp_xor

gmp_xor -- Bitwise XOR

Description

resource **gmp_xor** (resource `$a`, resource `$b`)

Calculates bitwise exclusive OR (XOR) of two GMP numbers.

Parameters

a
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b
: It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

Return Values

A GMP number resource.

Examples

Example #41 - gmp_xor() example

```
<?php
$xor1 = gmp_init("1101101110011101", 2);
$xor2 = gmp_init("0110011001011001", 2);

$xor3 = gmp_xor($xor1, $xor2);

echo gmp_strval($xor3, 2) . "\n";
?>
```

The above example will output:

```
1011110111000100
```