lround
llround

The `lround` function rounds its argument to the nearest integer value, returning it as a `long int` value. Like `round`, it rounds away from zero. `llround` is the same as `lround`, except that it returns a `long long int` value.

trunc

The `trunc` function rounds its argument to the nearest integer not larger in magnitude. (In other words, it truncates the argument toward zero.) `trunc` returns the result as a floating-point number.

Remainder Functions

```
float fmodf(float x, float y);                          see fmod
long double fmodl(long double x,
                  long double y);                       see fmod

double remainder(double x, double y);
float remainderf(float x, float y);
long double remainderl(long double x,
                       long double y);

double remquo(double x, double y, int *quo);
float remquof(float x, float y, int *quo);
long double remquol(long double x, long double y,
                    int *quo);
```

Besides additional versions of `fmod`, this category includes new remainder functions named `remainder` and `remquo`.

remainder

The `remainder` function returns `x` REM `y`, where REM is a function defined in the IEEE standard. For $y \neq 0$, the value of `x` REM `y` is $r = x - ny$, where n is the integer nearest the exact value of `x/y`. (If `x/y` is halfway between two integers, n is even.) If $r = 0$, it has the same sign as `x`.

remquo

The `remquo` function returns the same value as `remainder` when given the same first two arguments. In addition, `remquo` modifies the object pointed to by the `quo` parameter so that it contains n low-order bits of the integer quotient |`x/y`|, where n depends on the implementation but must be at least three. The value stored in this object will be negative if $x/y < 0$.

Manipulation Functions

```
double copysign(double x, double y);
float copysignf(float x, float y);
long double copysignl(long double x, long double y);

double nan(const char *tagp);
float nanf(const char *tagp);
long double nanl(const char *tagp);

double nextafter(double x, double y);
float nextafterf(float x, float y);
```