in `main` is equivalent to

`exit(`*expression*`);`

The difference between `return` and `exit` is that `exit` causes program termination regardless of which function calls it. The `return` statement causes program termination only when it appears in the `main` function. Some programmers use `exit` exclusively to make it easier to locate all exit points in a program.

9.6 Recursion

A function is ***recursive*** if it calls itself. For example, the following function computes $n!$ recursively, using the formula $n! = n \times (n - 1)!$:

```
int fact(int n)
{
  if (n <= 1)
    return 1;
  else
    return n * fact(n - 1);
}
```

Some programming languages rely heavily on recursion, while others don't even allow it. C falls somewhere in the middle: it allows recursion, but most C programmers don't use it that often.

To see how recursion works, let's trace the execution of the statement

```
i = fact(3);
```

Here's what happens:

```
fact(3) finds that 3 is not less than or equal to 1, so it calls
   fact(2), which finds that 2 is not less than or equal to 1, so it calls
      fact(1), which finds that 1 is less than or equal to 1, so it returns 1, causing
   fact(2) to return 2 × 1 = 2, causing
fact(3) to return 3 × 2 = 6.
```

Notice how the unfinished calls of `fact` "pile up" until `fact` is finally passed 1. At that point, the old calls of `fact` begin to "unwind" one by one, until the original call—`fact(3)`—finally returns with the answer, 6.

Here's another example of recursion: a function that computes x^n, using the formula $x^n = x \times x^{n-1}$.

```
int power(int x, int n)
{
  if (n == 0)
    return 1;
  else
    return x * power(x, n - 1);
}
```