C++ prototypes enable you to define default values for arguments. If a function call omits the corresponding argument, the program uses the default value. If the function includes an argument value, the program uses that value instead of the default. Default arguments can be provided only from right to left in the argument list. Thus, if you provide a default value for a particular argument, you also must provide default values for all arguments to the right of that argument.

A function's signature is its argument list. You can define two functions having the same name provided that they have different signatures. This is called function polymorphism, or function overloading. Typically, you overload functions to provide essentially the same service to different data types.

Function templates automate the process of overloading functions. You define a function using a generic type and a particular algorithm, and the compiler generates appropriate function definitions for the particular argument types you use in a program.

**Review Questions**

.1: What kinds of functions are good candidates for inline status?

.2: Suppose the `song()` function has this prototype:

```c
void song(char * name, int times);
```

a. How would you modify the prototype so that the default value for `times` is 1?

b. What changes would you make in the function definition?

c. Can you provide a default value of "O, My Papa" for `name`?

.3: Write overloaded versions of `iquote()`, a function that displays its argument enclosed in double quotation marks. Write three versions: one for an `int` argument, one for a `double` argument, and one for a `string` argument.