typically go well; hence, many compilers will warn when the two types are mixed, which itself is problematic.

If we happen to be on a 64-bit platform, we might choose to return a **long**:

```
long volYMD(SecId stock, int year, int month, int day);  // (3) NOT A GOOD IDEA
```

The problems using **long** as the return type are that it (1) is not yet generally considered a **vocabulary type** (see *Appendix — Historical perspective on the evolution of use of fundamental integral types* on page 93) and (2) would reduce portability (see *Potential Pitfalls — Relying on the relative sizes of* ***int***, ***long***, *and* ***long long*** below).

Prior to C++11, we might have considered returning a **double**:

```
double volYMD(SecId stock, int year, int month, int day);  // (4) OK
```

At least with **double** we know that we will have sufficient precision (53 bits) to express integers accurately into the quadrillions, which will certainly cover us for any foreseeable future. The main drawback is that **double** doesn't properly describe the nature of the type that we are returning, i.e., a whole integer number of shares, and so its algebra, although not as dubious as **unsigned int**, isn't ideal either.

With the advent of C++11, we might consider using one of the type aliases in `<cstdint>`:

```
std::int64_t volYMD(SecId stock, int year, int month, int day);  // (4) OK
```

This choice addresses most of the issues discussed above except that, instead of being a specific C++ type, it is a platform-dependent alias that is likely to be a **long** on a 64-bit platform and almost certainly a **long long** on a 32-bit one. Such exact size requirements are often necessary for packing data in structures and arrays but are not as useful when reasoning about them in the interfaces of functions where having a common set of fundamental vocabulary types becomes much more important, e.g., for interoperability.

All of this leads us to our final alternative, **long long**:

```
long long volYMD(SecId stock, int year, int month, int day);  // (5) GOOD IDEA
```

In addition to being a signed fundamental integral type of sufficient capacity on all platforms, **long long** is the same C++ type *relative* to other C++ types on all platforms.

Potential Pitfalls

Relying on the relative sizes of `int`, `long`, and `long long`

As discussed at some length in *Appendix — Historical perspective on the evolution of use of fundamental integral types* on page 93, the fundamental integral types have historically been a moving target. On older, 32-bit platforms, a **long** was often 32 bits, and **long long**, which was nonstandard prior to C++11, or its platform-dependent equivalent was needed to ensure that 64 bits were available. When the correctness of code