calculation:

```c
long double result = 1.0;
for (n = numbers, p = picks; p > 0; n--, p--)
    result = result * n / p ;
```

Rather than multiplying all the numerator terms first, the loop begins by multiplying 1.0 by
the first numerator term and then dividing by the first denominator term. Then, in the next
cycle, the loop multiplies and divides by the second numerator and denominator terms.
This keeps the running product smaller than if you did all the multiplication first. For
example, compare

```
(10 * 9) / (2 * 1)
```

with the following:

```
(10 / 2) * (9 / 1)
```

The first evaluates to 90 / 2 and then to 45, whereas the second evaluates to 5 * 9 and then
to 45. Both give the same answer, but the first method produces a larger intermediate
value (90) than does the second. The more factors you have, the bigger the difference
gets. For large numbers, this strategy of alternating multiplication with division can keep the
calculation from overflowing the maximum possible floating-point value.

Listing 7.4 incorporates this formula into an `odds()` function. Because the number of picks
and the total number of choices should be positive values, the program uses the `unsigned
int` type (unsigned, for short) for those quantities. Multiplying several integers together can
produce pretty large results, so `lotto.cpp` uses the `long double` type for the function's
return value. Also, terms such as 49 / 6 produce a truncation error for integer types.

**Compatibility Note**

Some C++ implementations don't support type `long
double`. If your implementation falls into that category, try
ordinary `double` instead.

**Listing 7.4** `lotto.cpp`