As shown in Program 6-30, a driver can be used to thoroughly test a function. It can repeatedly call the function with different test values as arguments. When the function performs as desired, it can be placed into the actual program it will be part of.

6.17 Little Lotto Case Study

Problem Statement

The mathematics department of Jefferson Junior High School wants a program developed that will illustrate basic probability for their students in an entertaining way. In particular they want a program called "Little Lotto" that simulates a lottery. In this program students can specify the number of numbers in the selection set (1–12) and the number of numbers patrons must pick and match to the winning numbers (between 1 and the size of the selection set). The order of the selected numbers is not significant. It will report the patron's chances of winning as both a number and a probability.

Example Output

This example output clarifies exactly what the department wants the program to do.

```
This program will tell you your probability of winning "Little Lotto".

How many numbers (1-12) are there to pick from? 12
How many numbers must you pick to play? 5

Your chance of winning the lottery is 1 chance in 792.
This is a probability of 0.0013
```

Program Design

Program Steps

The program must carry out the following general steps:

1. Get and validate how many numbers there are to choose from (n).
2. Get and validate how many of these numbers must be selected (k).
3. Compute the number of ways a set of k items can be selected from a set of n items.
4. Report to the player his chance of winning and his probability of winning.

Program Modules

The program will be designed as a set of modules, each having a specific function. Table 6-1 describes the modules that will be used:

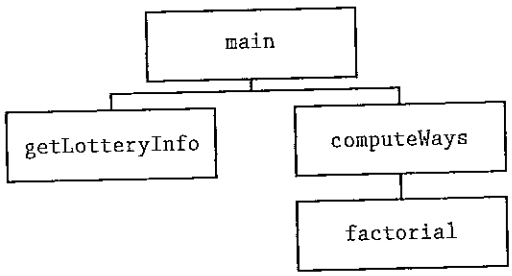
Table 6-1 Little Lotto Program Modules

Function	Description
`main`	This function explains the "game", organizes calls to other functions, and reports results.
`getLotteryInfo`	This function gets and validates the number of numbers to select from (n) and the number that must be chosen (k).
`computeWays`	This function computes the number of different sets of size k that can be chosen from n numbers.
`factorial`	This function computes factorials. It is used by `computeWays`.

Program Organization

In previous chapters hierarchy charts were used to illustrate the relationship of actions that a program must carry out. However, they are more commonly used to illustrate the relationship of program modules in a program that is organized into a set of functions. The hierarchy chart in Figure 6-15 illustrates the organization of the Little Lotto program. Notice that it clarifies which functions call which other functions.

Figure 6-15



Variables whose values will be input

```
int  pickFrom          // Number of numbers available to select from
int  numPicks          // Number of numbers that must be chosen
```

Variables and values whose values will be output

```
long int ways          // Number of different possible selections
                       // Only 1 of these can "win"
1.0 / ways             // Probability of winning
```

Detailed Pseudocode for Each Module

In a modular program, a separate pseudocode routine should be created to capture the logic of each function. Here is the pseudocode for each function in the Little Lotto program.

```
main
   Display information on what the program does
   Call getLotteryInfo  // Puts value in pickFrom and numPicks variables
   Call computeWays     // Returns number of ways numbers can be selected
   Store the returned result in the ways variable
   Display ways and 1 / ways
End main

getLotteryInfo          // Places inputs in reference variables
   Input pickFrom
   While pickFrom < 1 or pickFrom > 12
      Display an error message
      Input pickFrom
   End while
   Input numPicks
   While numPicks < 1 or numPicks > pickFrom
      Display an error message
      Input pickFrom
   End while
End getLotteryInfo

computeWays             // Receives pickFrom as n and numPicks as k
   Call factorial 3 times to get information for its calculations
   Return      factorial(n)
            factorial(k) *  factorial (n-k)
End computeWays

factorial      // Receives number whose factorial is to be calculated
   factTotal = 1
   Loop for count = number down to 1
      factTotal = factTotal * count
   End Loop
   Return factTotal
End factorial
```

The Program

The next step, after the pseudocode has been checked for logic errors, is to expand the pseudocode into the final program. This is shown in Program 6-31.

Program 6-31

```
1  // This program finds the probability of winning a "mini" lottery when
2  // the user's set of numbers must exactly match the set drawn by the
3  // lottery organizers. In addition to main, it uses three functions.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  // Function prototypes
9  void getLotteryInfo(int&, int&);
10 long int computeWays(int, int);
11 long int factorial(int);
12
```

(program continues)

Program 6-31 *(continued)*

```
13 int main()
14 {
15    int pickFrom,              // The number of numbers to pick from
16        numPicks;              // The number of numbers to select
17    long int ways;             // The number of different possible
18                               // ways to pick the set of numbers
19
20    cout << "This program will tell you your probability of "
21         << "winning \"Little Lotto\". \n";
22    getLotteryInfo(pickFrom, numPicks);
23    ways = computeWays(pickFrom, numPicks);
24
25    cout << fixed << showpoint << setprecision(4);
26    cout << "\nYour chance of winning the lottery is "
27         << "1 chance in " << ways << ".\n";
28    cout << "This is a probability of " << (1.0 / ways)    << "\n";
29    return 0;
30 }
31
32 /*************************************************************************
33  *                        getLotteryInfo                                 *
34  * Gets and validates lottery info. from the user and places it in *
35  * reference parameters referencing variables in the main function.*
36  *************************************************************************/
37 void getLotteryInfo(int &pickFrom, int &numPicks)
38 {
39    cout << "\nHow many numbers (1-12) are there to pick from? ";
40    cin  >> pickFrom;
41    while (pickFrom < 1 || pickFrom > 12)
42    {
43       cout << "There must be between 1 and 12 numbers.\n"
44            << "How many numbers (1-12) are there to pick from? ";
45       cin  >> pickFrom;
46    }
47    cout << "How many numbers must you pick to play? ";
48    cin  >> numPicks;
49    while (numPicks < 1 || numPicks > pickFrom)
50    {
51       if (numPicks < 1)                          // too few picks
52          cout << "You must pick at least one number.\n";
53       else                                       // too many picks
54          cout << "You must pick " << pickFrom << " or fewer numbers.\n";
55
56       cout << "How many numbers must you pick to play? ";
57       cin  >> numPicks;
58    }
59 }
60
```

(program continues)

Program 6-31 *(continued)*

```
61 /*****************************************************************
62  *                        computeWays                            *
63  * Computes and returns the number of different possible sets    *
64  * of k numbers that can be chosen from a set of n numbers.      *
65  * The formula for this is   n!                                  *
66  *                          --------                             *
67  *                          k!(n-k)!                             *
68  *****************************************************************/
69 // Note that the computation is done in a way that does not require
70 // multiplying two factorials together. This is done to prevent any
71 // intermediate result becoming so large that it causes overflow.
72 long int computeWays(int n, int k)
73 {
74    return ( factorial(n) / factorial(k) / factorial (n-k) );
75 }
76
77 /*****************************************************************
78  *                           factorial                           *
79  * Computes and returns the factorial of the non-negative integer *
80  * passed to it. n! means n * (n-1) * (n-2) ... * 1              *
81  * 0! is a special case and is defined to be 1.                  *
82  *****************************************************************/
83 // Notice that if number equals 0, the loop condition will
84 // initially be false and the loop will never be executed.
85 // This will, correctly, leave factTotal = 1.
86
87 long int factorial(int number)
88 {
89    long int factTotal = 1;
90
91    for (int count = number; count >= 1; count--)
92    {
93       factTotal *= count;
94    }
95    return factTotal;
96 }
```

Program Output with Example Input Shown in Bold

```
This program will tell you your probability of winning "Little Lotto".

How many numbers (1-12) are there to pick from? 10[Enter]
How many numbers must you pick to play? 3[Enter]

Your chance of winning the lottery is 1 chance in 120.
This is a probability of 0.0083
```

High Adventure Travel Agency Case Study

The following additional case study, which contain applications of material introduced in Chapter 6, can be found on the book's companion website. It demonstrates all the steps needed to develop a modular program that calculates and itemizes charges for the vacation packages offered by the High Adventure Travel Agency.