e)
```
for ( x = 1; x <= 20; ++x )
{
   if ( x % 5 == 0 )
      cout << x << endl;
   else
      cout << x << '\t';
}
```

5.3 a) *Error:* The semicolon after the `while` header causes an infinite loop.
Correction: Replace the semicolon by a `{`, or remove both the `;` and the `}`.

b) *Error:* Using a floating-point number to control a `for` repetition statement.
Correction: Use an `int` and perform the proper calculation to get the values you desire.

```
for ( y = 1; y != 10; ++y )
   cout << ( static_cast< double >( y ) / 10 ) << endl;
```

c) *Error:* Missing `break` statement in the first `case`.
Correction: Add a `break` statement at the end of the first `case`. This is not an error if you want the statement of `case 2:` to execute every time the `case 1:` statement executes.

d) *Error:* Improper relational operator used in the loop-continuation condition.
Correction: Use `<=` rather than `<`, or change `10` to `11`.

Exercises

5.4 *(Find the Code Errors)* Find the error(s), if any, in each of the following:

a)
```
For ( x = 100, x >= 1, ++x )
   cout << x << endl;
```

b) The following code should print whether integer `value` is odd or even:

```
switch ( value % 2 )
{
   case 0:
      cout << "Even integer" << endl;
   case 1:
      cout << "Odd integer" << endl;
}
```

c) The following code should output the odd integers from 19 to 1:

```
for ( x = 19; x >= 1; x += 2 )
   cout << x << endl;
```

d) The following code should output the even integers from 2 to 100:

```
counter = 2;

do
{
   cout << counter << endl;
   counter += 2;
} While ( counter < 100 );
```

5.5 *(Summing Integers)* Write a program that uses a `for` statement to sum a sequence of integers. Assume that the first integer read specifies the number of values remaining to be entered. Your program should read only one value per input statement. A typical input sequence might be

```
5 100 200 300 400 500
```

where the `5` indicates that the subsequent `5` values are to be summed.

5.6 *(Averaging Integers)* Write a program that uses a `for` statement to calculate the average of several integers. Assume the last value read is the sentinel `9999`. For example, the sequence `10 8 11 7 9 9999` indicates that the program should calculate the average of all the values preceding `9999`.

5.7 *(What Does This Program Do?)* What does the following program do?

```
// Exercise 5.7: ex05_07.cpp
// What does this program print?
#include <iostream>
using namespace std;

int main()
{
   int x; // declare x
   int y; // declare y

   // prompt user for input
   cout << "Enter two integers in the range 1-20: ";
   cin >> x >> y;  // read values for x and y

   for ( int i = 1; i <= y; ++i ) // count from 1 to y
   {
      for ( int j = 1; j <= x; ++j ) // count from 1 to x
         cout << '@'; // output @

      cout << endl; // begin new line
   } // end outer for
} // end main
```

5.8 *(Find the Smallest Integer)* Write a program that uses a `for` statement to find the smallest of several integers. Assume that the first value read specifies the number of values remaining.

5.9 *(Product of Odd Integers)* Write a program that uses a `for` statement to calculate and print the product of the odd integers from 1 to 15.

5.10 *(Factorials)* The factorial function is used frequently in probability problems. Using the definition of factorial in Exercise 4.34, write a program that uses a `for` statement to evaluate the factorials of the integers from 1 to 5. Print the results in tabular format. What difficulty might prevent you from calculating the factorial of 20?

5.11 *(Compound Interest)* Modify the compound interest program of Section 5.4 to repeat its steps for the interest rates 5%, 6%, 7%, 8%, 9% and 10%. Use a `for` statement to vary the interest rate.

5.12 *(Drawing Patterns with Nested* `for` *Loops)* Write a program that uses `for` statements to print the following patterns separately, one below the other. Use `for` loops to generate the patterns. All asterisks (`*`) should be printed by a single statement of the form `cout << '*';` (this causes the asterisks to print side by side). [*Hint:* The last two patterns require that each line begin with an appropriate number of blanks. *Extra credit:* Combine your code from the four separate problems into a single program that prints all four patterns side by side by making clever use of nested `for` loops.]

```
(a)              (b)               (c)              (d)
*                **********        **********                *
**               *********          *********               **
***              ********            ********              ***
****             *******              *******             ****
*****            ******                ******            *****
******           *****                  *****           ******
*******          ****                    ****          *******
********         ***                      ***         ********
*********        **                        **        *********
**********       *                          *       **********
```

5.13 *(Bar Chart)* One interesting application of computers is drawing graphs and bar charts. Write a program that reads five numbers (each between 1 and 30). Assume that the user enters only valid values. For each number that is read, your program should print a line containing that number of adjacent asterisks. For example, if your program reads the number 7, it should print *******.

5.14 *(Calculating Total Sales)* A mail order house sells five different products whose retail prices are: product 1 — $2.98, product 2—$4.50, product 3—$9.98, product 4—$4.49 and product 5—$6.87. Write a program that reads a series of pairs of numbers as follows:

a) product number
b) quantity sold

Your program should use a `switch` statement to determine the retail price for each product. Your program should calculate and display the total retail value of all products sold. Use a sentinel-controlled loop to determine when the program should stop looping and display the final results.

5.15 *(`GradeBook` Modification)* Modify the `GradeBook` program of Fig. 5.9–Fig. 5.11 to calculate the grade-point average. A grade of A is worth 4 points, B is worth 3 points, and so on.

5.16 *(Compound Interest Calculation)* Modify Fig. 5.6 so it uses only integers to calculate the compound interest. [*Hint:* Treat all monetary amounts as numbers of pennies. Then "break" the result into its dollar and cents portions by using the division and modulus operations. Insert a period.]

5.17 *(What Prints?)* Assume `i = 1`, `j = 2`, `k = 3` and `m = 2`. What does each statement print?

a) `cout << ( i == 1 ) << endl;`
b) `cout << ( j == 3 ) << endl;`
c) `cout << ( i >= 1 && j < 4 ) << endl;`
d) `cout << ( m <= 99 && k < m ) << endl;`
e) `cout << ( j >= i || k == m ) << endl;`
f) `cout << ( k + m < j || 3 - j >= k ) << endl;`
g) `cout << ( !m ) << endl;`
h) `cout << ( !( j - m ) ) << endl;`
i) `cout << ( !( k > m ) ) << endl;`

5.18 *(Number Systems Table)* Write a program that prints a table of the binary, octal and hexadecimal equivalents of the decimal numbers in the range 1–256. If you are not familiar with these number systems, read Appendix D, first. [*Hint:* You can use the stream manipulators `dec`, `oct` and `hex` to display integers in decimal, octal and hexadecimal formats, respectively.]

5.19 *(Calculating π)* Calculate the value of π from the infinite series

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

Print a table that shows the approximate value of π after each of the first 1000 terms of this series.

5.20 *(Pythagorean Triples)* A right triangle can have sides that are all integers. A set of three integer values for the sides of a right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse. Find all Pythagorean triples for `side1`, `side2` and `hypotenuse` all no larger than 500. Use a triple-nested `for` loop that tries all possibilities. This is an example of **brute force** computing. You'll learn in more advanced computer science courses that there are many interesting problems for which there's no known algorithmic approach other than sheer brute force.

5.21 *(Calculating Salaries)* A company pays its employees as managers (who receive a fixed weekly salary), hourly workers (who receive a fixed hourly wage for up to the first 40 hours they work and "time-and-a-half"—1.5 times their hourly wage—for overtime hours worked), commission workers (who receive $250 plus 5.7 percent of their gross weekly sales), or pieceworkers (who receive a fixed amount of money per item for each of the items they produce—each pieceworker in this company works on only one type of item). Write a program to compute the weekly pay for each employee. You do not know the number of employees in advance. Each type of employee has its own pay code: Managers have code 1, hourly workers have code 2, commission workers have code 3 and pieceworkers have code 4. Use a `switch` to compute each employee's pay according to that employee's paycode. Within the `switch`, prompt the user (i.e., the payroll clerk) to enter the appropriate facts your program needs to calculate each employee's pay according to that employee's paycode.

5.22 *(De Morgan's Laws)* In this chapter, we discussed the logical operators `&&`, `||` and `!`. De Morgan's laws can sometimes make it more convenient for us to express a logical expression. These laws state that the expression `!(` *condition1* `&&` *condition2* `)` is logically equivalent to the expression `( !`*condition1* `||` `!`*condition2* `)`. Also, the expression `!(` *condition1* `||` *condition2* `)` is logically equivalent to the expression `( !`*condition1* `&&` `!`*condition2* `)`. Use De Morgan's laws to write equivalent expressions for each of the following, then write a program to show that the original expression and the new expression in each case are equivalent:

a) `!( x < 5 ) && !( y >= 7 )`
b) `!( a == b ) || !( g != 5 )`
c) `!( ( x <= 8 ) && ( y > 4 ) )`
d) `!( ( i > 4 ) || ( j <= 6 ) )`

5.23 *(Diamond of Asterisks)* Write a program that prints the following diamond shape. You may use output statements that print a single asterisk (`*`), a single blank or a single newline. Maximize your use of repetition (with nested `for` statements) and minimize the number of output statements.

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

5.24 *(Diamond of Asterisks Modification)* Modify Exercise 5.23 to read an odd number in the range 1 to 19 to specify the number of rows in the diamond, then display a diamond of the appropriate size.

5.25 *(Removing* **`break`** *and* **`continue`***)* A criticism of the `break` and `continue` statements is that each is unstructured. These statements can always be replaced by structured statements. Describe in general how you'd remove any `break` statement from a loop in a program and replace it with some structured equivalent. [*Hint:* The `break` statement leaves a loop from within the body of the loop. Another way to leave is by failing the loop-continuation test. Consider using in the loop-continuation test a second test that indicates "early exit because of a 'break' condition."] Use the technique you developed here to remove the `break` statement from the program of Fig. 5.13.

5.26 What does the following program segment do?

```cpp
for ( int i = 1; i <= 5; ++i )
{
   for ( int j = 1; j <= 3; ++j )
   {
      for ( int k = 1; k <= 4; ++k )
         cout << '*';

      cout << endl;
   } // end inner for

   cout << endl;
} // end outer for
```

5.27 *(Removing the `continue` Statement)* Describe in general how you'd remove any `continue` statement from a loop in a program and replace it with some structured equivalent. Use the technique you developed here to remove the `continue` statement from the program of Fig. 5.14.

5.28 *("The Twelve Days of Christmas" Song)* Write a program that uses repetition and `switch` statements to print the song "The Twelve Days of Christmas." One `switch` statement should be used to print the day (i.e., "first," "second," etc.). A separate `switch` statement should be used to print the remainder of each verse. Visit the website `www.12days.com/library/carols/12daysofxmas.htm` for the complete lyrics to the song.

5.29 *(Peter Minuit Problem)* Legend has it that, in 1626, Peter Minuit purchased Manhattan Island for $24.00 in barter. Did he make a good investment? To answer this question, modify the compound interest program of Fig. 5.6 to begin with a principal of $24.00 and to calculate the amount of interest on deposit if that money had been kept on deposit until this year (e.g., 384 years through 2010). Place the `for` loop that performs the compound interest calculation in an outer `for` loop that varies the interest rate from 5% to 10% to observe the wonders of compound interest.

Making a Difference

5.30 *(Global Warming Facts Quiz)* The controversial issue of global warming has been widely publicized by the film *An Inconvenient Truth*, featuring former Vice President Al Gore. Mr. Gore and a U.N. network of scientists, the Intergovernmental Panel on Climate Change, shared the 2007 Nobel Peace Prize in recognition of "their efforts to build up and disseminate greater knowledge about man-made climate change." Research *both* sides of the global warming issue online (you might want to search for phrases like "global warming skeptics"). Create a five-question multiple-choice quiz on global warming, each question having four possible answers (numbered 1–4). Be objective and try to fairly represent both sides of the issue. Next, write an application that administers the quiz, calculates the number of correct answers (zero through five) and returns a message to the user. If the user correctly answers five questions, print "Excellent"; if four, print "Very good"; if three or fewer, print "Time to brush up on your knowledge of global warming," and include a list of the websites where you found your facts.

5.31 *(Tax Plan Alternatives; The "FairTax")* There are many proposals to make taxation fairer. Check out the FairTax initiative in the United States at

```
www.fairtax.org/site/PageServer?pagename=calculator
```

Research how the proposed FairTax works. One suggestion is to eliminate income taxes and most other taxes in favor of a 23% consumption tax on all products and services that you buy. Some FairTax opponents question the 23% figure and say that because of the way the tax is calculated, it would be more accurate to say the rate is 30%—check this carefully. Write a program that prompts the user to enter expenses in various expense categories they have (e.g., housing, food, clothing, transportation, education, health care, vacations), then prints the estimated FairTax that person would pay.

5.32 *(Facebook User Base Growth)* According to CNNMoney.com, Facebook hit 500 million users in July of 2010 and its user base has been growing at a rate of 5% per month. Using the compound-growth technique you learned in Fig. 5.6 and assuming this growth rate continues, how many months will it take for Facebook to grow its user base to one billion users? How many months will it take for Facebook to grow its user base to two billion users (which, at the time of this writing, was the total number of people on the Internet)?