EX 12.6 Write a recursive method that returns the value of $N!$ ($N$ factorial) using the definition given in this chapter. Explain why you would not normally use recursion to solve this problem.

EX 12.7 Write a recursive method to reverse a string. Explain why you would not normally use recursion to solve this problem.

EX 12.8 Design or generate a new maze for the MazeSearch program in this chapter and rerun the program. Explain the processing in terms of your new maze, giving examples of a path that was tried but failed, a path that was never tried, and the ultimate solution.

EX 12.9 Annotate the lines of output of the SolveTowers program in this chapter to show the recursive steps.

EX 12.10 Produce a chart showing the number of moves required to solve the Towers of Hanoi puzzle using the following number of disks: 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, and 25.

EX 12.11 How many line segments are used to construct a Koch snowflake of order $N$? Produce a chart showing the number of line segments that make up a Koch snowflake for orders 1 through 9.

**Programming Projects**

PP 12.1 Design and implement a recursive version of the PalindromeTester program from Chapter 5.

PP 12.2 Design and implement a program that implements Euclid’s algorithm for finding the greatest common divisor of two positive integers. The greatest common divisor is the largest integer that divides both values without producing a remainder. An iterative version of this method was part of the RationalNumber class presented in Chapter 7. In a class called DivisorCalc, define a static method called gcd that accepts two integers, num1 and num2. Create a driver to test your implementation. The recursive algorithm is defined as follows:

- $\text{gcd}(\text{num1}, \text{num2})$ is $\text{num1}$ if $\text{num1}$ divides into $\text{num2}$ evenly
- $\text{gcd}(\text{num1}, \text{num2})$ is $\text{gcd}(\text{num2}, \text{num1}\%\text{num2})$ otherwise

PP 12.3 Modify the Maze class so that it prints out the path of the final solution as it is discovered without storing it.
PP 12.4 Design and implement a program that traverses a 3D maze.

PP 12.5 Modify the TiledPictures program so that the repeated images appear in the lower-right quadrant.

PP 12.6 Design and implement a recursive program that solves the Non-Attacking Queens problem. That is, write a program to determine all ways in which eight queens can be positioned on an eight-by-eight chessboard so that none of them are in the same row, column, or diagonal as any other queen. There are no other chess pieces on the board.

PP 12.7 In the language of an alien race, all words take the form of Blurbs. A Blurb is a Whoozit followed by one or more Whatzits. A Whoozit is the character ‘x’ followed by zero or more ‘y’s. A Whatzit is a ‘q’ followed by either a ‘z’ or a ‘d’, followed by a Whoozit. Design and implement a recursive program that generates random Blurbs in this alien language.

PP 12.8 Design and implement a recursive program to determine whether a string is a valid Blurb as defined in PP12.7.

PP 12.9 Design and implement a recursive program to determine and print the Nth line of Pascal’s Triangle, as shown below. Each interior value is the sum of the two values above it. *Hint*: Use an array to store the values on each line.

\[
\begin{array}{cccccc}
1 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
1 & 5 & 10 & 10 & 5 & 1 \\
1 & 6 & 15 & 20 & 15 & 6 & 1 \\
1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1 \\
\end{array}
\]

PP 12.10 Design and implement an applet that generalizes the KochSnowflake program. Allow the user to choose a fractal design from a menu item and to pick the background and drawing colors. The buttons to increase and decrease the order of the fractal will apply to whichever fractal design is chosen. In addition to the Koch snowflake, include a C-curve fractal whose order 1 is a straight line. Each successive order is created by