TABLE 3.1

Recursive Definitions

What Is Being Defined	Characteristics
Recursive sequence	The first one or two values in the sequence are known; later items in the sequence are defined in terms of earlier items.
Recursive set	A few specific items are known to be in the set; other items in the set are built from combinations of items already in the set.
Recursive operation	A "small" case of the operation gives a specific value; other cases of the operation are defined in terms of smaller cases.
Recursive algorithm	For the smallest values of the arguments, the algorithm behavior is known; for larger values of the arguments, the algorithm invokes itself with smaller argument values.

SECTION 3.1 REVIEW

TECHNIQUES

- Generate values in a sequence defined recursively.
- Prove properties of the Fibonacci sequence.
- Recognize objects in a recursively defined collection of objects.
- Give recursive definitions for particular sets of objects.
- Give recursive definitions for certain operations on objects.
- Write recursive algorithms to generate sequences defined recursively.

MAIN IDEAS

- Recursive definitions can be given for sequences of objects, sets of objects, and operations on objects where basis information is known and new information depends on already known information.
- Recursive algorithms provide a natural way to solve certain problems by invoking the same task on a smaller version of the problem.

EXERCISES 3.1

For Exercises 1–12, write the first five values in the sequence.

1. $S(1) = 10$
   $S(n) = S(n-1) + 10$ for $n \geq 2$
2. $C(1) = 5$
   $C(n) = 2C(n-1) + 5$ for $n \geq 2$
3. $A(1) = 2$
   $A(n) = \dfrac{1}{A(n-1)}$ for $n \geq 2$
4. $B(1) = 1$
   $B(n) = B(n-1) + n^2$ for $n \geq 2$
5. $S(1) = 1$
   $S(n) = S(n-1) + \dfrac{1}{n}$ for $n \geq 2$
6. $T(1) = 1$
   $T(n) = nT(n-1)$ for $n \geq 2$

7. $P(1) = 1$
   $P(n) = n^2 P(n-1) + (n-1)$ for $n \geq 2$
8. $A(1) = 2$
   $A(n) = nA(n-1) + n$ for $n \geq 2$
9. $M(1) = 2$
   $M(2) = 2$
   $M(n) = 2M(n-1) + M(n-2)$ for $n > 2$
10. $D(1) = 3$
    $D(2) = 5$
    $D(n) = (n-1)D(n-1) + (n-2)D(n-2)$ for $n > 2$
11. $W(1) = 2$
    $W(2) = 3$
    $W(n) = W(n-1)W(n-2)$ for $n > 2$
12. $T(1) = 1$
    $T(2) = 2$
    $T(3) = 3$
    $T(n) = T(n-1) + 2T(n-2) + 3T(n-3)$ for $n > 3$

In Exercises 13–18, prove the given property of the Fibonacci numbers directly from the definition.

13. $F(n+1) + F(n-2) = 2F(n)$ for $n \geq 3$
14. $F(n) = 5F(n-4) + 3F(n-5)$ for $n \geq 6$
15. $F(n) = 3F(n-3) + 2F(n-4)$ for $n \geq 5$
16. $[F(n+1)]^2 = [F(n)]^2 + F(n-1)F(n+2)$ for $n \geq 2$
17. $F(n+3) = 2F(n+1) + F(n)$ for $n \geq 1$
18. $F(n+6) = 4F(n+3) + F(n)$ for $n \geq 1$

In Exercises 19–22, prove the given property of the Fibonacci numbers for all $n \geq 1$. (*Hint:* The first principle of induction will work.)

19. $F(1) + F(2) + \cdots + F(n) = F(n+2) - 1$
20. $F(2) + F(4) + \cdots + F(2n) = F(2n+1) - 1$
21. $F(1) + F(3) + \cdots + F(2n-1) = F(2n)$
22. $[F(1)]^2 + [F(2)]^2 + \cdots + [F(n)]^2 = F(n)F(n+1)$

In Exercises 23–26, prove the given property of the Fibonacci numbers using the second principle of induction.

23. Exercise 17
24. Exercise 18
25. $F(n) < 2^n$ for $n \geq 1$
26. $F(n) > \left(\frac{3}{2}\right)^{n-1}$ for $n \geq 6$

27. Write a pseudocode recursive algorithm for a function to compute $F(n)$, the nth Fibonacci number.

28. Walk through your recursive algorithm from Exercise 27 to compute $F(6)$.
    a. How many times is the function invoked?
    b. How many times is $F(4)$ computed?
    c. How many times is $F(3)$ computed?
    d. How many times is $F(2)$ computed?

Exercises 29 and 30 concern a proof of correctness of the following iterative algorithm for a function to compute $F(n)$, the nth Fibonacci number.

```
F(positive integer n)
//function that iteratively computes the value of
//the nth Fibonacci number
Local variables:
positive integer i              //loop index
positive integers p, q, r       //terms in Fibonacci sequence

  if n = 1 then
     return 1
  else
     if n = 2 then
        return 1
     else
        i = 2
        p = 1                   //p = lagging term in Fibonacci sequence
        q = 1                   //q = leading term in Fibonacci sequence
        while i < n do
           r = p + q            //form the next term as the
                                //sum of the two previous terms
           p = q                //bump up p
           q = r                //bump up q
           i = i + 1
        end while
        //q now has the value F(n)
        return q
     end if
  end if
end function F
```

29. a. In the iterative Fibonacci algorithm, the condition B for loop continuation is $i < n$, so B' is $i \geq n$, but what is the exact value of i when the loop terminates?
    b. When the loop exits, you want $q = F(n)$; what do you want for the value of p at that point?

30. a. Write the loop invariant Q for the iterative Fibonacci algorithm.
    b. Prove that Q is a loop invariant.

31. The values p and q are defined as follows:

$$p = \frac{1 + \sqrt{5}}{2} \quad \text{and} \quad q = \frac{1 - \sqrt{5}}{2}$$

a. Prove that $1 + p = p^2$ and $1 + q = q^2$.

b. Prove that

$$F(n) = \frac{p^n - q^n}{p - q}$$

c. Use part (b) to prove that

$$F(n) = \frac{\sqrt{5}}{5}\left(\frac{1 + \sqrt{5}}{2}\right)^n - \frac{\sqrt{5}}{5}\left(\frac{1 - \sqrt{5}}{2}\right)^n$$

is a closed-form solution for the Fibonacci sequence.

32. The *Lucas sequence* is defined by

$$L(1) = 1$$
$$L(2) = 3$$
$$L(n) = L(n - 1) + L(n - 2) \text{ for } n \geq 2$$

a. Write the first five terms of the sequence.

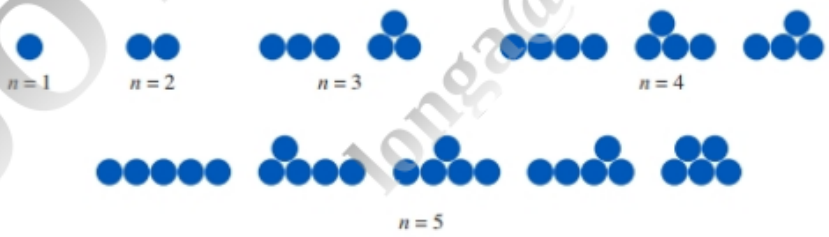
b. Prove that $L(n) = F(n + 1) + F(n - 1)$ for $n \geq 2$ where F is the Fibonacci sequence.

For Exercises 33–36, decide whether the sequences described are subsequences of the Fibonacci sequence, that is, whether their members are some or all of the members, in the right order, of the Fibonacci sequence.[3]

33. The sequence $A(n)$, where $A(n) = 1 +$ (the sum of the first n terms of the Fibonacci sequence), $n \geq 1$. The first four values are 2, 3, 5, 8, which—so far—form a subsequence of the Fibonacci sequence.

34. The sequence $B(n)$, where $B(n) = (n - 1)2^{n-2} + 1$, $n \geq 1$. The first four values are 1, 2, 5, 13, which—so far—form a subsequence of the Fibonacci sequence.

35. The sequence $C(n)$, where $C(n)$ is the number of ways in which n coins can be arranged in horizontal rows with all the coins in each row touching and every coin above the bottom row touching two coins in the row below it, $n \geq 1$. The first five values are 1, 1, 2, 3, 5, which—so far—form a subsequence of the Fibonacci sequence.



36. The sequence $D(n)$, where $D(n)$ describes the number of ways to paint the floors on an n-story building where each floor is painted yellow or blue and no two adjacent floors can be blue (although adjacent floors

[3]Exercises 33–36 are taken from "Mathematical Recreations" by Ian Stewart, *Scientific American*, May 1995.

can be yellow), $n \geq 1$. The first four values are 2, 3, 5, 8, which—so far—form a subsequence of the Fibonacci sequence. For example, $D(3) = 5$ because a three-story building can be painted

Y	Y	Y	B	B
Y	Y	B	Y	Y
Y	B	Y	B	Y

(*Hint:* think about a recursive expression for $D(n + 1)$.)

37 a. The original problem posed by Fibonacci concerned pairs of rabbits. Two rabbits do not breed until they are 2 months old. After that, each pair of rabbits produces a new pair each month. No rabbits ever die. Let $R(n)$ denote the number of rabbit pairs at the end of n months if you start with a single rabbit pair. Show that $R(n)$ is the Fibonacci sequence.

b. Write 27 and 62 as the sum of distinct nonconsecutive Fibonacci numbers.

38. a. The sequence of *Catalan numbers* is defined recursively by

$$C(0) = 1$$
$$C(1) = 1$$
$$C(n) = \sum_{k=1}^{n} C(k-1)C(n-k) \text{ for } n \geq 2$$

Compute the values of $C(2)$, $C(3)$, and $C(4)$ using this recurrence relation.

b. Frank and Jody are both candidates for president of the County Council. The number of votes cast equals $2n$, where n votes are cast for Frank and n for Jody. Votes are counted sequentially. The *ballot problem* asks: In how many ways can the votes be counted so that Jody's total is never ahead of Frank's total? The answer, as it turns out, is $C(n)$, the nth Catalan number. For example, if $n = 5$, one possible counting sequence that meets this requirement is

FFJJFJFFJJ

Using $n = 3$, write down all the satisfactory counting sequences and compare the result to the Catalan number $C(3)$.

39. A sequence is recursively defined by

$$S(1) = 2$$
$$S(2) = 2$$
$$S(3) = 6$$
$$S(n) = 3S(n-3) \text{ for } n \geq 3$$

Prove that $S(n)$ is an even number for $n \geq 1$.

40. A sequence is recursively defined by

$$T(5) = 6$$
$$T(6) = 10$$
$$T(n) = 2T(n-2) + 2 \text{ for } n \geq 7$$

Prove that $T(n) \geq 2n$ for $n \geq 7$.

41. A sequence is recursively defined by

$$S(0) = 1$$
$$S(1) = 1$$
$$S(n) = 2S(n-1) + S(n-2) \text{ for } n \geq 2$$

a. Prove that $S(n)$ is an odd number for $n \geq 0$.

b. Prove that $S(n) < 6S(n-2)$ for $n \geq 4$.

42. A sequence is recursively defined by

$$T(0) = 1$$
$$T(1) = 2$$
$$T(n) = 2T(n-1) + T(n-2) \text{ for } n \geq 2$$

Prove that $T(n) \leq \left(\frac{5}{2}\right)^n$ for $n \geq 0$.

43. Write a recursive definition for a geometric progression with initial term a and common ratio r (see Exercise 27, Section 2.2.).

44. Write a recursive definition for an arithmetic progression with initial term a and common difference d (see Exercise 28, Section 2.2.).

45. In an experiment, a certain colony of bacteria initially has a population of 50,000. A reading is taken every 2 hours, and at the end of every 2-hour interval, there are 3 times as many bacteria as before.

a. Write a recursive definition for $A(n)$, the number of bacteria present at the beginning of the nth time period.

b. At the beginning of which interval are there 1,350,000 bacteria present?

46. An amount of $500 is invested in an account paying 1.2% interest compounded annually.

a. Write a recursive definition for $P(n)$, the amount in the account at the beginning of the nth year.

b. After how many years will the account balance exceed $570?

47. A set T of numbers is defined recursively by

1. 2 belongs to T.
2. If x belongs to T, so does $x + 3$ and $2 * x$.

Which of the following numbers belong to T?

a. 6 b. 7 c. 19 d. 12

48. A set M of numbers is defined recursively by

1. 2 and 3 belong to M.
2. If x and y belong to M, so does $x * y$.

Which of the following numbers belong to M?

a. 6 b. 9 c. 16 d. 21 e. 26 f. 54 g. 72 h. 218

49. A set S of strings of characters is defined recursively by

    1. a and b belong to S.
    2. If x belongs to S, so does xb.

    Which of the following strings belong to S?

    a. a b. ab c. aba d. $aaab$ e. $bbbbb$

50. A set W of strings of symbols is defined recursively by

    1. a, b, and c belong to W.
    2. If x belongs to W, so does $a(x)c$.

    Which of the following strings belong to W?

    a. $a(b)c$ b. $a(a(b)c)c$ c. $a(abc)c$ d. $a(a(a(a)c)c)c$ e. $a(aacc)c$

51. A set S of integers is defined recursively by

    1. 0 and 3 belong to S.
    2. If x and y belong to S, so does $x + y$.

    Use structural induction to prove that every integer in S is a multiple of 3.

52. A set T of strings is defined recursively by

    1. pqq belongs to T.
    2. If x and y belong to T, so do $pxqq$, $qqxp$, and xy.

    Use structural induction to prove that every string in T has twice as many q's as p's.

53. Give a recursive definition for the set of all unary predicate wffs in x.
54. Give a recursive definition for the set of all well-formed formulas of integer arithmetic, involving integers together with the arithmetic operations of $+$, $-$, $*$, and $/$.
55. Give a recursive definition for the set of all odd integers.
56. Give a recursive definition for the set of all strings of well-balanced parentheses.
57. Give a recursive definition for the set of all binary strings containing an odd number of 0s.
58. Give a recursive definition for the set of all binary strings containing an even number of 1s.
59. Give a recursive definition for the set of all binary strings ending with 0.
60. Give a recursive definition for the set of all binary strings with an equal number of 0s and 1s.
61. Use BNF notation to define the set of positive integers.
62. Use BNF notation to define the set of decimal numbers, which consist of an optional sign ($+$ or $-$), followed by one or more digits, followed by a decimal point, followed by zero or more digits.
63. Give a recursive definition for x^R, the reverse of the string x.
64. Give a recursive definition for $|x|$, the length of the string x.
65. Give a recursive definition for the factorial operation $n!$ for $n \geq 1$.

66. Give a recursive definition for the addition of two nonnegative integers m and n.
67. a. Write a recursive definition for the operation of taking the maximum of n integers $a_1, \ldots, a_n$, $n \geq 2$.
    b. Write a recursive definition for the operation of taking the minimum of n integers $a_1, \ldots, a_n$, $n \geq 2$.
68. a. Give a recursive definition for the conjunction of n statement letters in propositional logic, $n \geq 2$.
    b. Write a generalization of the associative property of conjunction (tautological equivalence 2b of Section 1.1) and use induction to prove it.
69. Let A and $B_1, B_2, \ldots, B_n$ be statement letters. Prove the finite extension of the distributive equivalences of propositional logic:

$$A \vee (B_1 \wedge B_2 \wedge \cdots \wedge B_n) \Leftrightarrow (A \vee B_1) \wedge (A \vee B_2) \wedge \cdots \wedge (A \vee B_n)$$

and

$$A \wedge (B_1 \vee B_2 \vee \cdots \vee B_n) \Leftrightarrow (A \wedge B_1) \vee (A \wedge B_2) \vee \cdots \vee (A \wedge B_n)$$

for $n \geq 2$.

70. Let $B_1, B_2, \ldots, B_n$ be statement letters. Prove the finite extension of De Morgan's laws:

$$(B_1 \vee B_2 \vee \cdots \vee B_n)' \Leftrightarrow B'_1 \wedge B'_2 \wedge \cdots \wedge B'_n$$

and

$$(B_1 \wedge B_2 \wedge \cdots \wedge B_n)' \Leftrightarrow B'_1 \vee B'_2 \vee \cdots \vee B'_n$$

for $n \geq 2$.

In Exercises 71–76, write the body of a recursive function to compute $S(n)$ for the given sequence S.

71. 1, 3, 9, 27, 81, ...
72. 2, 1, 1/2, 1/4, 1/8, ...
73. 1, 2, 4, 7, 11, 16, 22, ...
74. 2, 4, 16, 256, ...
75. $a, b, a + b, a + 2b, 2a + 3b, 3a + 5b, \ldots$
76. $p, p - q, p + q, p - 2q, p + 2q, p - 3q, \ldots$
77. What value is returned by the following recursive function Mystery for an input value of n?

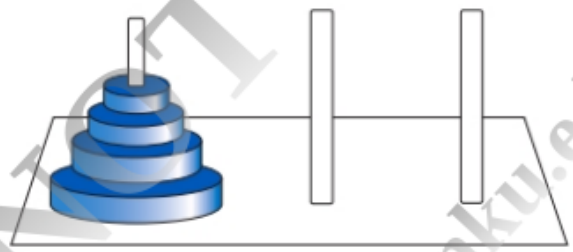
```
Mystery (positive integer n)
    if n = 1 then
        return 1
    else
        return Mystery(n − 1) + 1
    end if
end function Mystery
```

78. The following recursive function is initially invoked with an i value of 1. L is a list (array) of 10 integers. What does the function do?

```
g(list L; positive integer i; integer x)
    if i > 10 then
        return 0
    else
```

```
        if L[ i ] = x then
            return 10
        else
            return g(L, i + 1, x)
        end if
    end if
end function g
```

79. Informally describe a recursive algorithm to reverse the entries in a list of items.
80. Informally describe a recursive algorithm to compute the sum of the digits of a positive integer.
81. Informally describe a recursive algorithm to compute the greatest common divisor of two positive integers a and b where $a > b$. (*Hint*: The solution is based on the Euclidean algorithm, discussed in Section 2.3. In particular, make use of expression (5) on page 134.)
82. The famous Towers of Hanoi puzzle involves 3 pegs with n disks of varying sizes stacked in order from the largest (on the bottom) to the smallest (on the top) on 1 of the pegs. The puzzle requires that the disks end up stacked the same way on a different peg; only one disk at a time can be moved to another peg, and no disk can ever be stacked on top of a smaller disk. Informally describe a recursive algorithm to solve the Towers of Hanoi puzzle.



83. Simulate the execution of algorithm *SelectionSort* on the following list L; write the list after every exchange that changes the list.

4, 10, −6, 2, 5

84. Simulate the execution of algorithm *SelectionSort* on the following list L; write the list after every exchange that changes the list.

9, 0, 2, 6, 4

85. The binary search algorithm is used with the following list; x has the value "Chicago." Name the elements against which x is compared.

Boston, Charlotte, Indianapolis, New Orleans, Philadelphia, San Antonio, Yakima

86. The binary search algorithm is used with the following list; x has the value "flour." Name the elements against which x is compared.

butter, chocolate, eggs, flour, shortening, sugar

87. Do a proof of correctness for the iterative function given in this section to compute $S(n)$ of Example 1, $S(n) = 2^n$.
88. The *Online Encyclopedia of Integer Sequences* (OEIS) was originated and maintained for many years by Neil Sloane, a mathematician at AT&T who has also written several books about sequences. The OEIS Foundation now manages the database, which contains more than 200,000 sequences of integers that have

been submitted and studied by many people. (See oeis.org). There is even a YouTube movie about the OEIS! *Recaman's sequence* (number A005132 in the OEIS catalog) is a recursive sequence defined as follows:

$$a(1) = 1$$

For $n > 1$,

$$a(n) = \begin{cases} a(n-1) - n \text{ if that number is positive and not already in the sequence,} \\ \quad \text{otherwise} \\ a(n-1) + n \end{cases}$$

a. Confirm that the first few terms of this sequence are 1, 3, 6, 2, 7, 13.

b. It has been conjectured that every nonnegative integer will eventually appear in this sequence. Find the index of this sequence at which the following numbers appear: 10, 12, 23.

SECTION 3.2 RECURRENCE RELATIONS

We developed two algorithms, one iterative and one recursive, to compute a value $S(n)$ for the sequence S of Example 1. However, there is a still easier way to compute $S(n)$. Recall that

$$S(1) = 2 \tag{1}$$

$$S(n) = 2S(n-1) \text{ for } n \geq 2 \tag{2}$$

Because

$$S(1) = 2 = 2^1$$
$$S(2) = 4 = 2^2$$
$$S(3) = 8 = 2^3$$
$$S(4) = 16 = 2^4$$

and so on, we can see that

$$S(n) = 2^n \tag{3}$$

Using Equation (3), we can plug in a value for n and compute $S(n)$ without having to compute—either explicitly, or, through recursion, implicitly—all the lower values of S first. An equation such as (3), where we can substitute a value and get the output value back directly, is called a **closed-form solution** to the recurrence relation (2) subject to the basis step (1). Finding a closed-form solution is called **solving** the recurrence relation.

Recurrence relations can be used to describe a variety of things, from chemical degradation (see the opening problem for this chapter) to the amount in an interest-bearing account, from the growth of species to the spread of a computer virus. Clearly, it is nice to find a closed-form solution to a recurrence relation whenever possible.

Linear First-Order Recurrence Relations

Expand, Guess, and Verify

One technique for solving recurrence relations is an "expand, guess, and verify" approach that repeatedly uses the recurrence relation to expand the expression for the nth term until the general pattern can be guessed. Finally the guess is verified by mathematical induction.