5.1 RECURSIVE DEFINITIONS

The Handshake Problem
There are n guests at a sesquicentennial ball. Each person shakes hands with everybody else exactly once. How many handshakes are made?

Recursive (or inductive) Definition of a Function
Let \( a \in \mathbb{W} \) and \( X = \{a, a+1, a+2, \ldots\} \). The recursive definition of a function \( f \) with domain \( X \) consists of three parts:

- **Basis clause** A few initial values of the function \( f(a), f(a+1), \ldots, f(a+k-1) \) are specified. An equation that specifies such an initial value is an initial condition.

- **Recursive clause** A formula to compute \( f(n) \) from the \( k \) preceding functional values \( f(n-1), f(n-2), \ldots, f(n-k) \) is made. Such a formula is called a recurrence relation (or recursion formula).

- **Terminal clause** Only values thus obtained are valid functional values. (For convenience, we drop this clause from our recursive definition.)

**Example 1** Define recursively the factorial function \( f \).

**Example 2** Judy deposits $1,000 in a local bank at an annual interest of 8% compounded annually. Define recursively the compound \( A(n) \) she will receive at the end of \( n \) years.

**Example 3** Define recursively the number of handshakes \( h(n) \) made in the handshake problem.
Example 4  (The Tower of Brahma)  According to a legend, at the beginning of creation, God stacked 64 golden disks on one of three diamond pegs on a brass platform in the temple of Brahma at Benares, India. The priests on duty were asked to move the disks from peg X to peg Z using Y as an auxiliary peg under the following conditions:

1) Only one disk can be moved at a time.
2) No disk can be placed on the top of a smaller disk.

The priests were told the world would end when the job was completed!

Suppose there are $n$ disks on peg X. Let $b_n$ denote the number of moves needed to move them from peg X to peg Z, using peg Y as an intermediary. Define $b_n$ recursively.

Example 6  Let $a_n$ denote the number of times the assignment statement $x ← x + 1$ is executed by the following nested for loops. Define $a_n$ recursively.

```
for i = 1 to n do
    for j = 1 to i do
        for k = 1 to j do
            x ← x + 1
```

Example 7  (Fibonacci)  Leonardo Fibonacci (1170? - 1250?), the most outstanding Italian mathematician of the Middle Ages, proposed the following problem around 1202:

Suppose there are two newborn rabbits, one male and the other female. Find the number of rabbits produced in a year if:

1) each pair takes one month to become mature;
2) Each pair produces a mixed pair every month, from the second month, and
3) no rabbits die.
Example 8  Let $a_n$ denote the number of $n$-bit words containing no two consecutive 1's. Define $a_n$ recursively.

5.2 SOLVING RECURRENCE RELATIONS
finding an explicit formula for $f(n)$

Example 10  (The handshake problem continued)
$h(1) = 0$
$h(n) = h(n-1) + n-1, \quad n \geq 2$

solution
step 1  Use iteration to predict a formula for $h(n)$
step 2  Use induction to verify the formula.

Example 11  Solve the following recurrence relation:
\[ a_0 = 0 \]
\[ a_n = a_{n-1} + \frac{n(n + 1)}{2}, \quad n \geq 1 \]

Example 12  Solve the recurrence relation in the Tower of Brahma problem.
5.3 Solving Recurrence Relations Revisited

**kth order LHRRWCC**

\[ a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k} \]

where \( c_1, c_2, \ldots, c_k \in \mathbb{R} \) and \( c_k \neq 0 \).

**2nd order LHRRWCC**

\[ a_n = a a_{n-1} + b a_{n-2} \]

characteristic equation \( x^2 - ax - b = 0 \)

characteristic roots \( \alpha, \beta \)

basic solutions \( \alpha^n, \beta^n \)

general solution \( a_n = A\alpha^n + B\beta^n \)

**Theorem 2** Let \( \alpha \) and \( \beta \) be the distinct solutions of the equation \( x^2 - ax - b = 0 \), where \( a, b \in \mathbb{R} \). Then every solution of the LHRRWCC \( a_n = a a_{n-1} + b a_{n-2} \), where \( a_0 = C_0 \) and \( a_1 = C_1 \), is of the form \( a_n = A\alpha^n + B\beta^n \) for some constants \( A \) and \( B \).

**Example 14** Solve the recurrence relation \( a_n = 5a_{n-1} - 6a_{n-2} \), where \( a_0 = 4 \) and \( a_1 = 7 \).

**Example 15** Solve the Fibonacci recurrence relation \( F_n = F_{n-1} + F_{n-2} \), where \( F_1 = 1 = F_2 \).

**Example 16** Let \( a_n \) denote the number of subsets of the set \( S = \{1, 2, \ldots, n\} \) that do not contain consecutive integers, where \( n \geq 0 \). When \( n = 0 \), \( S = \emptyset \). Find an explicit formula for \( a_n \).

**Theorem 3** Let \( a, b \in \mathbb{R} \) and \( b \neq 0 \). Let \( \alpha \) be a solution of the equation \( x^2 - ax - b = 0 \) with degree of multiplicity two. Then \( a_n = A\alpha^n + Bn\alpha^n \) is the general solution of the LHRRWCC \( a_n = aa_{n-1} + ba_{n-2} \).
Example 17  Solve the recurrence relation $a_n = 6a_{n-1} - 9a_{n-2}$, where $a_0 = 2$ and $a_1 = 3$.

Theorem 4  Let $\alpha$ be a characteristic root of the LHRRWCC (1).
1) If the degree of multiplicity of $\alpha$ is 1, then $\alpha^n$ is a basic solution of the LHRRWCC.
2) If the degree of multiplicity of $\alpha$ is $m$, then $\alpha^n, n\alpha^n, \ldots, n^{m-1}\alpha^n$ are basic solutions of the LHRRWCC. (Note: A kth order LHRRWCC has k basic solutions.)
3) The general solution of the LHRRWCC is a linear combination of all basic solutions.

Example 18  Solve the recurrence relation $a_n = 7a_{n-1} - 13a_{n-2} - 3a_{n-3} + 18a_{n-4}$ where $a_0 = 5$, $a_1 = 3$, $a_2 = 6$, and $a_3 = -21$.

5.5 RECURSIVE ALGORITHMS

recursive algorithm
    invokes a simpler version of itself.

factorial algorithm
handshake algorithm
tower of Brahma algorithm
Fibonacci algorithm

```plaintext
Algorithm Fibonacci(n)
(* This algorithm computes the nth Fibonacci number using recursion.*)
0. Begin (* algorithm *)
1. if n = 1 or n = 2 then (* base cases *)
2.   Fibonacci ← 1
3. else (* general case *)
4.   Fibonacci ← Fibonacci(n-1) + Fibonacci(n-2)
5. End (* algorithm *)
```

Algorithm 4
Example 32  Write a recursive algorithm to compute the gcd of two positive integers \( x \) and \( y \).

Example 33  (Binary Search Algorithm)  Write a recursive algorithm to search an ordered list \( X \) of \( n \) items and determine if a certain item (key) occurs in the list. Return the location of key if the search is successful.

solution  Since the algorithm is extremely useful, first we outline it:

- compute the middle index.
- if key = middle value then
  - we are done and exit
- else if key < middle value then
  - search the lower half
- else
  - search the upper half.

| Algorithm binary search(X,low,high,key,found,mid) |
| (* The algorithm returns the location of key in the variable mid in the list X if the search is successful. Low, mid, and high denote the lowest, middle, and highest indices of the list. Found is a boolean variable; it is true if key is found and false otherwise. *) |
| 0. Begin (* algorithm *) |
| 1. if low ≤ high then (* list is nonempty *) |
| 2. begin (* if *) |
| 3. found ← false (* boolean flag *) |
| 4. mid ← ⌊(low + high)/2⌋ |
| 5. if key = x_mid then |
| 6. found ← true (* we are done. *) |
| 7. else |
| 8. if key < x_mid then (* search the lower half *) |
| 9. binary search(X,low,mid-1,key,found,mid) |
| 10. else (* search the upper half *) |
| 11. binary search(X,mid+1,high,key,found,mid) |
| 12. endif |
| 13. End (* algorithm *) |

Algorithm 6
5.6 CORRECTNESS OF RECURSIVE ALGORITHMS

Example 26   Establish the correctness of the recursive linear search algorithm in Algorithm 8.

```plaintext
Algorithm linear search (X,n,key,location)
0. Begin (* algorithm *)
1. if n = 0 then (* unsuccessful search *)
2. location ← 0
3. else if x_n = key then
4. location ← n
5. else
6. linear search(X,n-1,key,location)
7. End (* algorithm *)
```

Algorithm 5

Example 27   Establish the correctness of the bubble sort algorithm in Algorithm 9.

```plaintext
Algorithm Bubble Sort(X,n)
0. Begin (* algorithm *)
1. if n > 1 then (* list contains at least two elements *)
2. begin (* if *)
3. for i = 1 to n-1 do
4. if x_i > x_{i+1} then (* they are out of order *)
5. swap x_i and x_{i+1}
6. bubble sort(X,n-1)
7. endif
8. End (* algorithm *)
```

Algorithm 9