36. Binomial coefficients can also be defined as follows:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Write a nonrecursive function to calculate binomial coefficients using this definition.

7.2 Examples of Recursion: Towers of Hanoi; Parsing

Towers of Hanoi

The Towers of Hanoi problem is a classic example of a problem for which a recursive algorithm is especially appropriate. It can be solved easily using recursion, but a nonrecursive solution is considerably more difficult. The problem is to solve the puzzle shown in Fig. 7.1, in which one must move the disks from the left peg to the right peg according to the following rules:

1. When a disk is moved, it must be placed on one of the three pegs.
2. Only one disk may be moved at a time, and it must be the top disk on one of the pegs.
3. A larger disk may never be placed on top of a smaller one.

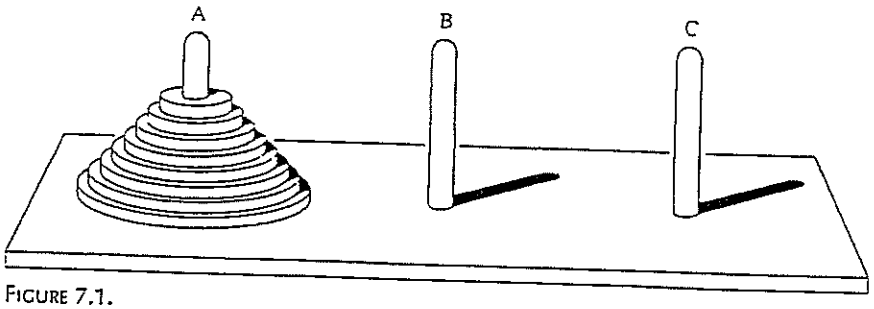

Figure 7.1.

Legend has it that the priests in the Temple of Bramah were given a puzzle consisting of a golden platform with three diamond needles on which were placed sixty-four golden disks. The priests were to move one disk per day, following these rules, and when they had successfully finished moving the disks to another needle, time would end. (*Question:* If the priests moved one disk per day and began their work in year 0, when would time end?)

Novices usually find the puzzle easy to solve for a small number of disks, but they have more difficulty as the number of disks grows to seven, eight, and beyond. To a computer scientist, however, the Towers of Hanoi puzzle is easy: We begin by identifying a base case, for which the problem is trivial to solve:

If there is one disk, then move it from Peg A to Peg C.

The puzzle is thus easily solved for $n = 1$ disk. We then give an inductive solution for $n > 1$ disks, in which we assume that a solution exists for $n - 1$ disks:

1. *Move the topmost $n - 1$ disks from Peg A to Peg B, using Peg C for temporary storage.*
2. *Move the final disk remaining on Peg A to Peg C.*
3. *Move the $n - 1$ disks from Peg B to Peg C, using Peg A for temporary storage.*

This scheme is implemented by the following recursive function:

```
/* Move is a recursive function to solve the
 * Towers of Hanoi puzzle.
 *
 *  Receive: n, the number of disks to be moved;
 *           source, peg containing disk to move
 *           destination, peg where to move disk
 *           spare, peg to store disks temporarily,
 *  Output:  A message describing the move
 ************************************************/

void Move(int n,
          char source, char destination, char spare)
{
  if (n >= 1)                                  // anchor
    cout << "Move the top disk from " << source
         << " to " << destination << endl;
  else
  {                                            // inductive case
    Move(n-1, source, spare, destination);
    Move(1, source, destination, spare);
    Move(n-1, spare, destination, source);
  }
}
```

Figure 7.2 presents a driver program that uses `Move()` to solve the Hanoi Towers problem, and an execution in which the problem is solved for four disks.

FIGURE 7.2 SOLVING THE TOWERS OF HANOI PROBLEM RECURSIVELY

```
/* Program to solve the Towers of Hanoi puzzle recursively.
 *
 * Input:  numDisks, the number of disks to be moved
 * Output: A sequence of moves that solve the puzzle
 ****************************************************************/

#include <iostream>
using namespace std;
```

```
void Move(int n, char source, char destination, char spare);
{
  const char PEG1 = 'A',              // the three pegs
             PEG2 = 'B',
             PEG3 = 'C';

  cout << "This program solves the Hanoi Towers puzzle.\n\n";

  cout << "Enter the number of disks: ";
  int numDisks;                       // the number of disks to be moved
  cin >> numDisks;
  cout << endl;

  Move(numDisks, PEG1, PEG2, PEG3); // the solution
}

/*** Insert definition of function Move() here. ***/
```

SAMPLE RUN:

```
This program solves the Hanoi Towers puzzle.

Enter the number of disks: 4

Move the top disk from A to B
Move the top disk from A to C
Move the top disk from B to C
Move the top disk from A to B
Move the top disk from C to A
Move the top disk from C to B
Move the top disk from A to B
Move the top disk from A to C
Move the top disk from B to C
Move the top disk from B to A
Move the top disk from C to A
Move the top disk from B to C
Move the top disk from A to B
Move the top disk from A to C
Move the top disk from B to C
```

PARSING

All the examples of recursion that we have given thus far have used **direct recursion**: that is, the functions have called themselves directly. **Indirect recursion** occurs when a function calls other functions, and some chain of function calls eventually results in a call to the first function again. For example, function `F()` may call function `G()`, which calls function `H()`, which calls `F()` again.

To illustrate indirect recursion, we consider the compiler problem of processing arithmetic expressions. In particular, we consider the specific problem of parsing arithmetic expressions, that is, determining whether they are well formed and, if so, what their structure is.

```
Input two letters, range (eg. c,j)
a
aba
abcba
abcdcba
abcdedcba
abcdefedcba
abcdefgfedcba
abcdefghgfedcba
abcdefghihgfedcba
abcdefghijihgfedcba
abcdefghijkjihgfedcba
abcdefghijklkjihgfedcba
abcdefghijklmlkjihgfedcba
Input two letters, range (eg. c,j)
```

III Write a pgm to print out any size triangle base on either one or two letter input.
Restriction NO LOOPS IN CODE!

```
A
ABA
ABCBA
ABCDCBA
ABCDEDCBA
ABCDEFEDCBA
ABCDEFGFEDCBA
ABCDEFGHGFEDCBA
ABCDEFGHIHGFEDCBA
ABCDEFGHIJIHGFEDCBA
ABCDEFGHIJKJIHGFEDCBA
ABCDEFGHIJKLKJIHGFEDCBA
ABCDEFGHIJKLMLKJIHGFEDCBA
ABCDEFGHIJKLMNMLKJIHGFEDCBA
ABCDEFGHIJKLMNONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`a`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`aba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA
```