*1.3 (*Display a pattern*) Write a program that displays the following pattern:

```
   J     A     V     V     A
   J    A A    V   V    A A
J  J   AAAAA    V V    AAAAA
 J J   A   A     V     A   A
```

1.4 (*Print a table*) Write a program that displays the following table:

```
a     a*2   a*3
1     2     3
2     4     6
3     6     9
4     8     12
```

1.5 (*Compute expressions*) Write a program that displays the result of

$$\frac{7.5 \times 2.5 - 1.5 \times 3}{35.5 - 2.5}$$

**2.13 (*Financial application: compound value*) Suppose you save $100 *each* month into a savings account with the annual interest rate 5%. Thus, the monthly interest rate is 0.05/12 = 0.00417. After the first month, the value in the account becomes

```
100 * (1 + 0.00417) = 100.417
```

After the second month, the value in the account becomes

```
(100 + 100.417) * (1 + 0.00417) = 201.252
```

After the third month, the value in the account becomes

```
(100 + 201.252) * (1 + 0.00417) = 302.507
```

and so on.

Write a program that prompts the user to enter a monthly saving amount and displays the account value after the sixth month. (In Exercise 4.30, you will use a loop to simplify the code and display the account value for any month.)

```
Enter the monthly saving amount: 100
After the sixth month, the account value is $608.81
```

2.25 (*Financial application: payroll*) Write a program that reads the following information and prints a payroll statement:

Employee's name (e.g., Smith)
Number of hours worked in a week (e.g., 10)
Hourly pay rate (e.g., 6.75)
Federal tax withholding rate (e.g., 20%)
State tax withholding rate (e.g., 9%)

```
Enter employee's name: Smith
Enter number of hours worked in a week: 10
Enter hourly pay rate: 6.75
Enter federal tax withholding rate: 0.20
Enter state tax withholding rate: 0.09

Employee Name: Smith
Hours Worked:  10.0
Pay Rate:  $6.75
Gross Pay:  $67.5
Deductions:
  Federal Withholding (20.0%):  $13.5
  State Withholding (9.0%):  $6.07
  Total Deduction:  $19.57
Net Pay:  $47.93
```

*3.17 (*Game: scissor, rock, paper*) Write a program that plays the popular scissor-roc paper game. (A scissor can cut a paper, a rock can knock a scissor, and a paper ca wrap a rock.) The program randomly generates a number 0, 1, or 2 representin scissor, rock, and paper. The program prompts the user to enter a number 0, 1, 2 and displays a message indicating whether the user or the computer wins, lose or draws. Here are sample runs:

```
scissor (0), rock (1), paper (2): 1
The computer is scissor. You are rock. You won
```

```
scissor (0), rock (1), paper (2): 2
The computer is paper. You are paper too. It is a draw
```

Comprehensive

**3.21 (*Science: day of the week*) Zeller's congruence is an algorithm developed by Christian Zeller to calculate the day of the week. The formula is

$$h = \left(q + \frac{26(m+1)}{10} + k + \frac{k}{4} + \frac{j}{4} + 5j\right) \% 7$$

where

- h is the day of the week (0: Saturday, 1: Sunday, 2: Monday, 3: Tuesday, 4: Wednesday, 5: Thursday, 6: Friday).
- q is the day of the month.
- m is the month (3: March, 4: April, . . ., 12: December). January and February are counted as months 13 and 14 of the previous year.
- j is the century (i.e., $\frac{year}{100}$).
- k is the year of the century (i.e., *year* % 100).

Note that the division in the formula performs an integer division. Write a program that prompts the user to enter a year, month, and day of the month, and displays the name of the day of the week. Here are some sample runs:

```
Enter year: (e.g., 2012): 2015
Enter month: 1-12: 1
Enter the day of the month: 1-31: 25
Day of the week is Sunday
```

```
Enter year: (e.g., 2012): 2012
Enter month: 1-12: 5
Enter the day of the month: 1-31: 12
Day of the week is Saturday
```

(*Hint*: January and February are counted as 13 and 14 in the formula, so you need to convert the user input 1 to 13 and 2 to 14 for the month and change the year to the previous year.)

4.16 (*Find the prime factors of an integer*) Write a program that reads an integer and displays its smallest prime factors in ascending order. For example, if the integer is 60, the output should be as follows: 2, 3, 5.

4.25 (*Sum a series*) Write a program to sum the following series:

$$1 + (1 + 2) + (1 + 2 + 3) + \ldots + (1 + 2 + 3 + \ldots + n)$$

Here n is given by the user at the time of execution.

***5.5** (*Find largest number*) Write a method with the following header to display the largest number of three numbers:

```
public static void displayLargestNumber(
  double num1, double num2, double num3)
```

Write a test program that displays the largest of three numbers.

5.9 (*Conversions between feet and meters*) Write a class that contains the following two methods:

```
/** Convert from feet to meters */
public static double footToMeter(double foot)

/** Convert from meters to feet */
public static double meterToFoot(double meter)
```

The formula for the conversion is:

```
meter = 0.305 * foot
foot = 3.279 * meter
```

Write a test program that invokes these methods to display the following tables:

Feet	Meters		Meters	Feet
1.0	0.305		20.0	65.574
2.0	0.610		25.0	81.967
...				
9.0	2.745		60.0	196.721
10.0	3.050		65.0	213.115

5.10 (*Use the `isPrime` Method*) Listing 5.7, PrimeNumberMethod.java, provides the `isPrime(int number)` method for testing whether a number is prime. Use this method to find the number of prime numbers less than 20000.

5.11 (*Financial application: compute commissions*) Write a method that computes the commission, using the scheme in Programming Exercise 4.39. The header of the method is as follows:

```
public static double computeCommission(double salesAmount)
```

Write a test program that displays the following table:

Sales Amount	Commission
10000	900.0
15000	1500.0
...	
95000	11100.0
100000	11700.0

5.12 (*Display characters*) Write a method that prints characters using the following header:

```
public static void printChars(char ch1, char ch2, int
  numberPerLine)
```

This method prints the characters between `ch1` and `ch2` with the specified number per line. Write a test program

6.13 (*Random number chooser*) Write a method that returns a random number between 1 and 54, excluding the numbers passed in the argument. The method header is specified as follows:

```
public static int getRandom(int... numbers)
```

***6.21** (*Game: bean machine*) The bean machine, also known as a quincunx or the Galton box, is a device for statistics experiments named after English scientist Sir Francis Galton. It consists of an upright board with evenly spaced nails (or pegs) in a triangular form, as shown in Figure 6.15.

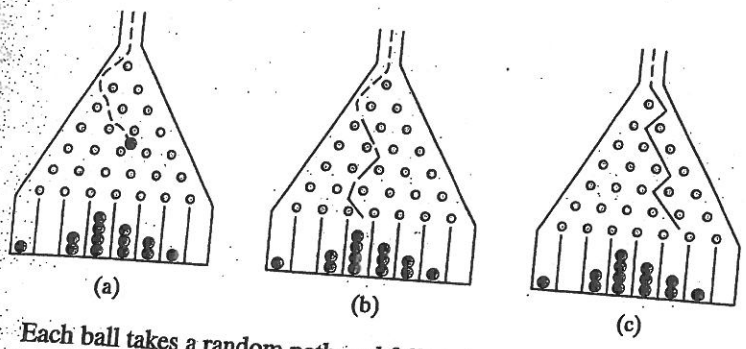

FIGURE 6.15 Each ball takes a random path and falls into a slot.

Balls are dropped from the opening of the board. Every time a ball hits a nail, it has a 50% chance of falling to the left or to the right. The piles of balls are accumulated in the slots at the bottom of the board.

Write a program that simulates the bean machine. Your program should prompt the user to enter the number of the balls and the number of the slots in the machine. Simulate the falling of each ball by printing its path. For example, the path for the ball in Figure 6.15b is LLRRLLR and the path for the ball in Figure 6.15c is RLRRLRR. Display the final buildup of the balls in the slots in a histogram. Here is a sample run of the program:

```
Enter the number of balls to drop: 5
Enter the number of slots in the bean machine: 7

LRLRLRR
RRLLLRR
LLRLLRR
RRLLLLL
LRLRRLR

  0
  0
 000
```

(*Hint*: Create an array named `slots`. Each element in `slots` stores the number of balls in a slot. Each ball falls into a slot via a path. The number of Rs in a path is the position of the slot where the ball falls. For example, for the path LRLRLRR, the ball falls into `slots[4]`, and for the path is RRLLLLL, the ball falls into `slots[2]`.)

7.1 (*Sum elements column by column*) Write a method that returns the sum of all the elements in a specified column in a matrix using the following header:

```
public static double sumColumn(double[][] m, int columnIndex)
```

Write a test program that reads a 3-by-4 matrix and displays the sum of each column. Here is a sample run:

```
Enter a 3-by-4 matrix row by row:
1 5 2 3.4
5.5 6 7.8 1
0.5 1 3 1
Sum of the elements at column 0 is 16.5
Sum of the elements at column 1 is 9.0
Sum of the elements at column 2 is 13.0
Sum of the elements at column 3 is 13.0
```

***7.2** (*Product of the main diagonal*) ...

*****7.9** (*Game: play a tic-tac-toe game*) In a game of tic-tac-toe, two players take turns marking an available cell in a 3 × 3 grid with their respective tokens (either X or O). When one player has placed three tokens in a horizontal, vertical, or diagonal row on the grid, the game is over and that player has won. A draw (no winner) occurs when all the cells on the grid have been filled with tokens and neither player has achieved a win. Create a program for playing tic-tac-toe.

The program prompts two players to enter an X token and O token alternately. Whenever a token is entered, the program redisplays the board on the console and determines the status of the game (win, draw, or continue). Here is a sample run:

```
 |   |   |
 |   |   |
 |   |   |
Enter a row (0, 1, or 2) for player X: 1
Enter a column (0, 1, or 2) for player X: 1
 |   |   |
 | X |   |
 |   |   |
Enter a row (0, 1, or 2) for player O: 1
Enter a column (0, 1, or 2) for player O: 2
 |   |   |
 | X | O |
 |   |   |
Enter a row (0, 1, or 2) for player X:
. . .
 | X |   |
 | O | X | O |
 |   |   | X |
X player won
```

***7.10** (*Largest row and column*) ...