Test Questions

Answer the quiz for this chapter online at www.cs.armstrong.edu/liang/intro10e/quiz.html.

Programming Exercises

think before coding

Pedagogical Note

For each exercise, carefully analyze the problem requirements and design strategies for solving the problem before coding.

Debugging Tip

Before you ask for help, read and explain the program to yourself, and trace it using several representative inputs by hand or using an IDE debugger. You learn how to program by debugging your own mistakes.

learn from mistakes

Section 3.2

***3.1** (*Algebra: solve quadratic equations*) The two roots of a quadratic equation $ax^2 + bx + c = 0$ can be obtained using the following formula:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$b^2 - 4ac$ is called the discriminant of the quadratic equation. If it is positive, the equation has two real roots. If it is zero, the equation has one root. If it is negative, the equation has no real roots.

Write a program that prompts the user to enter values for a, b, and c and displays the result based on the discriminant. If the discriminant is positive, display two roots. If the discriminant is `0`, display one root. Otherwise, display "The equation has no real roots".

Note that you can use `Math.pow(x, 0.5)` to compute $\sqrt{x}$. Here are some sample runs.

```
Enter a, b, c: 1.0 3 1 ↵Enter
The equation has two roots -0.381966 and -2.61803
```

```
Enter a, b, c: 1 2.0 1 ↵Enter
The equation has one root -1
```

```
Enter a, b, c: 1 2 3 ↵Enter
The equation has no real roots
```

3.2 (*Game: add three numbers*) The program in Listing 3.1, AdditionQuiz.java, generates two integers and prompts the user to enter the sum of these two integers. Revise the program to generate three single-digit integers and prompt the user to enter the sum of these three integers.

Sections 3.3–3.7

***3.3** (*Algebra: solve* 2 × 2 *linear equations*) A linear equation can be solved using Cramer's rule given in Programming Exercise 1.13. Write a program that prompts the user to enter *a*, *b*, *c*, *d*, *e*, and *f* and displays the result. If *ad* − *bc* is **0**, report that "The equation has no solution."

```
Enter a, b, c, d, e, f: 9.0 4.0 3.0 -5.0 -6.0 -21.0 ↵Enter
x is -2.0 and y is 3.0
```

```
Enter a, b, c, d, e, f: 1.0 2.0 2.0 4.0 4.0 5.0 ↵Enter
The equation has no solution
```

****3.4** (*Random month*) Write a program that randomly generates an integer between 1 and 12 and displays the English month name January, February, …, December for the number 1, 2, …, 12, accordingly.

***3.5** (*Find future dates*) Write a program that prompts the user to enter an integer for today's day of the week (Sunday is 0, Monday is 1, …, and Saturday is 6). Also prompt the user to enter the number of days after today for a future day and display the future day of the week. Here is a sample run:

```
Enter today's day: 1 ↵Enter
Enter the number of days elapsed since today: 3 ↵Enter
Today is Monday and the future day is Thursday
```

```
Enter today's day: 0 ↵Enter
Enter the number of days elapsed since today: 31 ↵Enter
Today is Sunday and the future day is Wednesday
```

***3.6** (*Health application: BMI*) Revise Listing 3.4, ComputeAndInterpretBMI.java, to let the user enter weight, feet, and inches. For example, if a person is 5 feet and 10 inches, you will enter **5** for feet and **10** for inches. Here is a sample run:

```
Enter weight in pounds: 140 ↵Enter
Enter feet: 5 ↵Enter
Enter inches: 10 ↵Enter
BMI is 20.087702275404553
Normal
```

3.7 (*Financial application: monetary units*) Modify Listing 2.10, ComputeChange .java, to display the nonzero denominations only, using singular words for single units such as 1 dollar and 1 penny, and plural words for more than one unit such as 2 dollars and 3 pennies.

VideoNote
Sort three integers

***3.8** (*Sort three integers*) Write a program that prompts the user to enter three integers and display the integers in non-decreasing order.

****3.9** (*Business: check ISBN-10*) An **ISBN-10** (International Standard Book Number) consists of 10 digits: $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}$. The last digit, d_{10}, is a checksum, which is calculated from the other nine digits using the following formula:

$$(d_1 \times 1 + d_2 \times 2 + d_3 \times 3 + d_4 \times 4 + d_5 \times 5 + \\ d_6 \times 6 + d_7 \times 7 + d_8 \times 8 + d_9 \times 9) \% 11$$

If the checksum is **10**, the last digit is denoted as X according to the ISBN-10 convention. Write a program that prompts the user to enter the first 9 digits and displays the 10-digit ISBN (including leading zeros). Your program should read the input as an integer. Here are sample runs:

```
Enter the first 9 digits of an ISBN as integer: 013601267 ↵
The ISBN-10 number is 0136012671
```

```
Enter the first 9 digits of an ISBN as integer: 013031997 ↵
The ISBN-10 number is 013031997X
```

3.10 (*Game: addition quiz*) Listing 3.3, SubtractionQuiz.java, randomly generates a subtraction question. Revise the program to randomly generate an addition question with two integers less than 100.

Sections 3.8–3.16

***3.11** (*Find the number of days in a month*) Write a program that prompts the user to enter the month and year and displays the number of days in the month. For example, if the user entered month **2** and year **2012**, the program should display that February 2012 had 29 days. If the user entered month **3** and year **2015**, the program should display that March 2015 had 31 days.

3.12 (*Palindrome number*) Write a program that prompts the user to enter a three-digit integer and determines whether it is a palindrome number. A number is palindrome if it reads the same from right to left and from left to right. Here is a sample run of this program:

```
Enter a three-digit integer: 121 ↵
121 is a palindrome
```

```
Enter a three-digit integer: 123 ↵
123 is not a palindrome
```

***3.13** (*Financial application: compute taxes*) Listing 3.5, ComputeTax.java, gives the source code to compute taxes for single filers. Complete Listing 3.5 to compute the taxes for all filing statuses.

3.14 (*Game: heads or tails*) Write a program that lets the user guess whether the flip of a coin results in heads or tails. The program randomly generates an integer **0** or **1**, which represents head or tail. The program prompts the user to enter a guess and reports whether the guess is correct or incorrect.

**3.15 (*Game: lottery*) Revise Listing 3.8, Lottery.java, to generate a lottery of a three-digit number. The program prompts the user to enter a three-digit number and determines whether the user wins according to the following rules:

1. If the user input matches the lottery number in the exact order, the award is $10,000.
2. If all digits in the user input match all digits in the lottery number, the award is $3,000.
3. If one digit in the user input matches a digit in the lottery number, the award is $1,000.

3.16 (*Random point*) Write a program that displays a random coordinate in a rectangle. The rectangle is centered at (0, 0) with width 100 and height 200.

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

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

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

*3.18 (*Cost of shipping*) A shipping company uses the following function to calculate the cost (in dollars) of shipping based on the weight of the package (in pounds).

$$c(w) = \begin{cases} 3.5, \text{ if } 0 < w <= 1 \\ 5.5, \text{ if } 1 < w <= 3 \\ 8.5, \text{ if } 3 < w <= 10 \\ 10.5, \text{ if } 10 < w <= 20 \end{cases}$$

Write a program that prompts the user to enter the weight of the package and display the shipping cost. If the weight is greater than 50, display a message "the package cannot be shipped."

**3.19 (*Compute the perimeter of a triangle*) Write a program that reads three edges for a triangle and computes the perimeter if the input is valid. Otherwise, display that the input is invalid. The input is valid if the sum of every pair of two edges is greater than the remaining edge.

*3.20 (*Science: wind-chill temperature*) Programming Exercise 2.17 gives a formula to compute the wind-chill temperature. The formula is valid for temperatures in the range between −58ºF and 41ºF and wind speed greater than or equal to `2`. Write a program that prompts the user to enter a temperature and a wind speed. The program displays the wind-chill temperature if the input is valid; otherwise, it displays a message indicating whether the temperature and/or wind speed is invalid.

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
Enter month: 1-12:  1 ↵Enter
Enter the day of the month: 1-31: 25 ↵Enter
Day of the week is Sunday
```

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

VideoNote
Check point location

****3.22** (*Geometry: point in a circle?*) Write a program that prompts the user to enter a point (`x`, `y`) and checks whether the point is within the circle centered at (`0`, `0`) with radius `10`. For example, (`4`, `5`) is inside the circle and (`9`, `9`) is outside the circle, as shown in Figure 3.7a.

(*Hint*: A point is in the circle if its distance to (`0`, `0`) is less than or equal to `10`. The formula for computing the distance is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Test your program to cover all cases.) Two sample runs are shown below.

```
Enter a point with two coordinates: 4 5 ↵Enter
Point (4.0, 5.0) is in the circle
```

```
Enter a point with two coordinates: 9 9 ↵Enter
Point (9.0, 9.0) is not in the circle
```

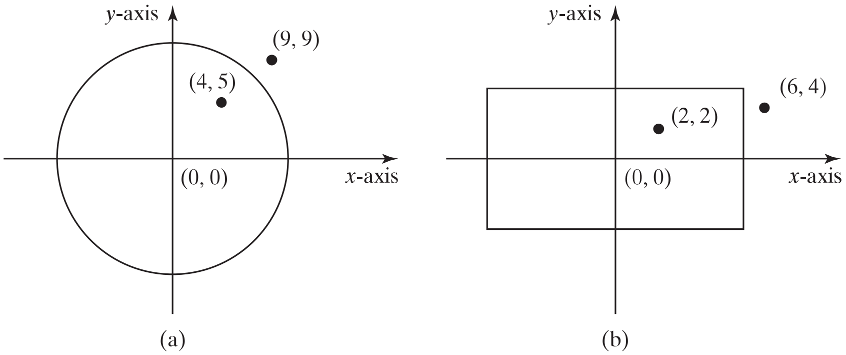


FIGURE 3.7 (a) Points inside and outside of the circle. (b) Points inside and outside of the rectangle.

****3.23** (*Geometry: point in a rectangle?*) Write a program that prompts the user to enter a point (x, y) and checks whether the point is within the rectangle centered at (0, 0) with width 10 and height 5. For example, (2, 2) is inside the rectangle and (6, 4) is outside the rectangle, as shown in Figure 3.7b. (*Hint*: A point is in the rectangle if its horizontal distance to (0, 0) is less than or equal to 10 / 2 and its vertical distance to (0, 0) is less than or equal to 5.0 / 2. Test your program to cover all cases.) Here are two sample runs.

```
Enter a point with two coordinates: 2 2 ↵Enter
Point (2.0, 2.0) is in the rectangle
```

```
Enter a point with two coordinates: 6 4 ↵Enter
Point (6.0, 4.0) is not in the rectangle
```

****3.24** (*Game: pick a card*) Write a program that simulates picking a card from a deck of 52 cards. Your program should display the rank (Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King) and suit (Clubs, Diamonds, Hearts, Spades) of the card. Here is a sample run of the program:

```
The card you picked is Jack of Hearts
```

***3.25** (*Geometry: intersecting point*) Two points on line 1 are given as (x1, y1) and (x2, y2) and on line 2 as (x3, y3) and (x4, y4), as shown in Figure 3.8a–b.

The intersecting point of the two lines can be found by solving the following linear equation:

$$(y_1 - y_2)x - (x_1 - x_2)y = (y_1 - y_2)x_1 - (x_1 - x_2)y_1$$

$$(y_3 - y_4)x - (x_3 - x_4)y = (y_3 - y_4)x_3 - (x_3 - x_4)y_3$$

This linear equation can be solved using Cramer's rule (see Programming Exercise 3.3). If the equation has no solutions, the two lines are parallel (Figure 3.8c).

Write a program that prompts the user to enter four points and displays the intersecting point. Here are sample runs:

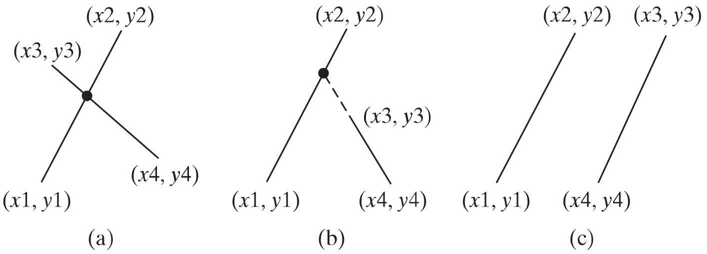


FIGURE 3.8 Two lines intersect in (a and b) and two lines are parallel in (c).

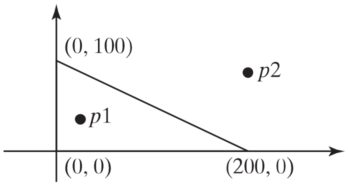
```
Enter x1, y1, x2, y2, x3, y3, x4, y4: 2 2 5 -1.0 4.0 2.0 -1.0 -2.0 ↵Enter
The intersecting point is at (2.88889, 1.1111)
```

```
Enter x1, y1, x2, y2, x3, y3, x4, y4: 2 2 7 6.0 4.0 2.0 -1.0 -2.0 ↵Enter
The two lines are parallel
```

3.26 (*Use the* **&&**, **||** *and* **^** *operators*) Write a program that prompts the user to enter an integer and determines whether it is divisible by 5 and 6, whether it is divisible by 5 or 6, and whether it is divisible by 5 or 6, but not both. Here is a sample run of this program:

```
Enter an integer: 10 ↵Enter
Is 10 divisible by 5 and 6? false
Is 10 divisible by 5 or 6? true
Is 10 divisible by 5 or 6, but not both? true
```

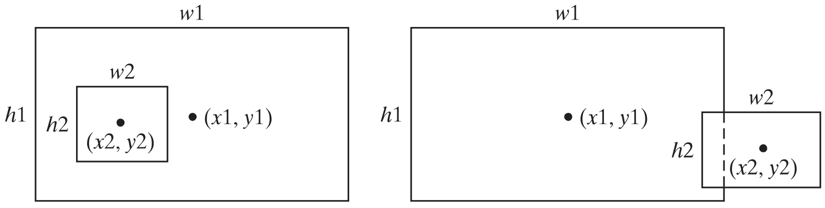
****3.27** (*Geometry: points in triangle?*) Suppose a right triangle is placed in a plane as shown below. The right-angle point is placed at (0, 0), and the other two points are placed at (200, 0), and (0, 100). Write a program that prompts the user to enter a point with *x*- and *y*-coordinates and determines whether the point is inside the triangle. Here are the sample runs:



```
Enter a point's x- and y-coordinates: 100.5 25.5 ↵Enter
The point is in the triangle
```

```
Enter a point's x- and y-coordinates: 100.5 50.5
The point is not in the triangle
```

****3.28** (*Geometry: two rectangles*) Write a program that prompts the user to enter the center *x*-, *y*-coordinates, width, and height of two rectangles and determines whether the second rectangle is inside the first or overlaps with the first, as shown in Figure 3.9. Test your program to cover all cases.



(a) (b)

FIGURE 3.9 (a) A rectangle is inside another one. (b) A rectangle overlaps another one.

Here are the sample runs:

```
Enter r1's center x-, y-coordinates, width, and height: 2.5 4 2.5 43
Enter r2's center x-, y-coordinates, width, and height: 1.5 5 0.5 3
r2 is inside r1
```

```
Enter r1's center x-, y-coordinates, width, and height: 1 2 3 5.5
Enter r2's center x-, y-coordinates, width, and height: 3 4 4.5 5
r2 overlaps r1
```

```
Enter r1's center x-, y-coordinates, width, and height: 1 2 3 3
Enter r2's center x-, y-coordinates, width, and height: 40 45 3 2
r2 does not overlap r1
```

****3.29** (*Geometry: two circles*) Write a program that prompts the user to enter the center coordinates and radii of two circles and determines whether the second circle is inside the first or overlaps with the first, as shown in Figure 3.10. (*Hint*: circle2 is inside circle1 if the distance between the two centers <= |r1 - r2| and circle2 overlaps circle1 if the distance between the two centers <= r1 + r2. Test your program to cover all cases.)

Here are the sample runs:

```
Enter circle1's center x-, y-coordinates, and radius: 0.5 5.1 13
Enter circle2's center x-, y-coordinates, and radius: 1 1.7 4.5
circle2 is inside circle1
```

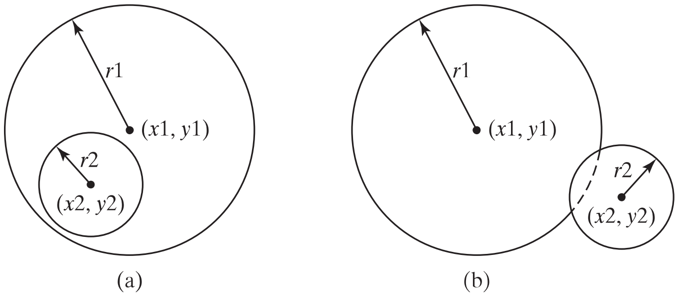


FIGURE 3.10 (a) A circle is inside another circle. (b) A circle overlaps another circle.

```
Enter circle1's center x-, y-coordinates, and radius: 3.4 5.7 5.5 ↵Enter
Enter circle2's center x-, y-coordinates, and radius: 6.7 3.5 3 ↵Enter
circle2 overlaps circle1
```

```
Enter circle1's center x-, y-coordinates, and radius: 3.4 5.5 1 ↵Enter
Enter circle2's center x-, y-coordinates, and radius: 5.5 7.2 1 ↵Enter
circle2 does not overlap circle1
```

***3.30** (*Current time*) Revise Programming Exercise 2.8 to display the hour using a 12-hour clock. Here is a sample run:

```
Enter the time zone offset to GMT: -5 ↵Enter
The current time is 4:50:34 AM
```

***3.31** (*Financials: currency exchange*) Write a program that prompts the user to enter the exchange rate from currency in U.S. dollars to Chinese RMB. Prompt the user to enter **0** to convert from U.S. dollars to Chinese RMB and **1** to convert from Chinese RMB and U.S. dollars. Prompt the user to enter the amount in U.S. dollars or Chinese RMB to convert it to Chinese RMB or U.S. dollars, respectively. Here are the sample runs:

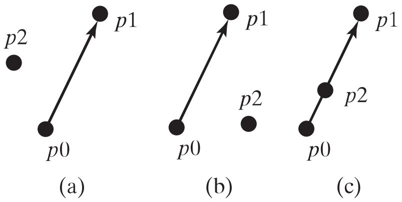
```
Enter the exchange rate from dollars to RMB: 6.81 ↵Enter
Enter 0 to convert dollars to RMB and 1 vice versa: 0 ↵Enter
Enter the dollar amount:  100 ↵Enter
$100.0 is 681.0 yuan
```

```
Enter the exchange rate from dollars to RMB: 6.81 ↵Enter
Enter 0 to convert dollars to RMB and 1 vice versa: 5 ↵Enter
Enter the RMB amount:  10000 ↵Enter
10000.0 yuan is $1468.43
```

```
        Enter the exchange rate from dollars to RMB: 6.81 ↵Enter
        Enter 0 to convert dollars to RMB and 1 vice versa: 5 ↵Enter
        Incorrect input
```

***3.32** (*Geometry: point position*) Given a directed line from point p0(x0, y0) to p1(x1, y1), you can use the following condition to decide whether a point p2(x2, y2) is on the left of the line, on the right, or on the same line (see Figure 3.11):

$$(x1 - x0)*(y2 - y0) - (x2 - x0)*(y1 - y0) \begin{cases} >0 \text{ p2 is on the left side of the line} \\ =0 \text{ p2 is on the same line} \\ <0 \text{ p2 is on the right side of the line} \end{cases}$$



(a) (b) (c)

FIGURE 3.11 (a) p2 is on the left of the line. (b) p2 is on the right of the line. (c) p2 is on the same line.

Write a program that prompts the user to enter the three points for p0, p1, and p2 and displays whether p2 is on the left of the line from p0 to p1, on the right, or on the same line. Here are some sample runs:

```
Enter three points for p0, p1, and p2: 4.4 2 6.5 9.5 -5 4 ↵Enter
(-5.0, 4.0) is on the left side of the line from (4.4, 2.0) to (6.5, 9.5)
```

```
        Enter three points for p0, p1, and p2: 1 1 5 5 2 2 ↵Enter
        (2.0, 2.0) is on the line from (1.0, 1.0) to (5.0, 5.0)
```

```
Enter three points for p0, p1, and p2: 3.4 2 6.5 9.5 5 2.5 ↵Enter
(5.0, 2.5) is on the right side of the line from (3.4, 2.0) to (6.5, 9.5)
```

***3.33** (*Financial: compare costs*) Suppose you shop for rice in two different packages. You would like to write a program to compare the cost. The program prompts the user to enter the weight and price of the each package and displays the one with the better price. Here is a sample run:

```
        Enter weight and price for package 1: 50 24.59 ↵Enter
        Enter weight and price for package 2: 25 11.99 ↵Enter
        Package 2 has a better price.
        Enter weight and price for package 1: 50 25 ↵Enter
        Enter weight and price for package 2: 25 12.5 ↵Enter
        Two packages have the same price.
```

***3.34** (*Geometry: point on line segment*) Programming Exercise 3.32 shows how to test whether a point is on an unbounded line. Revise Programming Exercise 3.32 to test whether a point is on a line segment. Write a program that prompts the user to enter the three points for p0, p1, and p2 and displays whether p2 is on the line segment from p0 to p1. Here are some sample runs:

```
Enter three points for p0, p1, and p2: 1 1 2.5 2.5 1.5 1.5 ↵Enter
(1.5, 1.5) is on the line segment from (1.0, 1.0) to (2.5, 2.5) ↵Enter
```

```
Enter three points for p0, p1, and p2:  1 1 2 2 3.5 3.5 ↵Enter
(3.5, 3.5) is not on the line segment from (1.0, 1.0) to (2.0, 2.0)
```