Programming Exercises

The exercises in this section are optional and do not report to the performance dashboard. Instructors can decide whether to assign these exercises and students can check the correctness of their programs using the Check Exercise tool.

Pedagogical Note

For each exercise, you should 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.

Sections 3.2

***3.1** (*Algebra: solve quadratic equations*) The two roots of a quadratic equation, for example, $ax^2 + bx + c = 0$, 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"`.

***3.2** (*Game: add three numbers*) The program in Listing 3.1 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 3.3–3.8

***3.3** (*Algebra: solve* 2×2 *linear equations)* You can use Cramer's rule to solve the following 2×2 system of linear equation:

$$\begin{aligned} ax + by &= e \\ cx + dy &= f \end{aligned} \qquad x = \frac{ed - bf}{ad - bc} \qquad y = \frac{af - ec}{ad - bc}$$

Write a program that prompts the user to enter `a`, `b`, `c`, `d`, `e`, and `f` and display the result. If $ad - bc$ is `0`, report that `"The equation has no solution"`.

**3.4 (*Game: learn addition*) Write a program that generates two integers under 100 and prompts the user to enter the sum of these two integers. The program then reports true if the answer is correct, false otherwise. The program is similar to Listing 3.1.

*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.

*3.6 (*Health application: BMI*) Revise Listing 3.5, ComputeAndInterpretBMI.py, to let users enter their weight in pounds and their height in feet and inches. For example, if a person is `5` feet and `10` inches, you will enter `5` for feet and `10` for inches.

3.7 (*Financial application: monetary units*) Modify Listing 2.5, ComputeChange.py, 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.

*3.8 (*Sort three numbers*) Write a program that prompts the user to enter three numbers and displays them in increasing order.

*3.9 (*Financial: compare costs*) Suppose you shop for rice and find it in two different-sized packages. You would like to write a program to compare the cost of the packages. The program prompts the user to enter the weight and price of each package and then displays the one with the better price.

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

Sections 3.9–3.16

***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 `2000`, the program should display that February 2000 has 29 days. If the user entered month `3` and year `2005`, the program should display that March 2005 has 31 days.

3.12 (*Check a number*) Write a program that prompts the user to enter an integer and checks whether the number is divisible by both `5` and `6`, divisible by `5` or `6`, or just one of them (but not both).

***3.13** (*Financial application: compute taxes*) Listing 3.6, ComputeTax.py, gives the source code to compute taxes for single filers. Complete Listing 3.6 to give the complete source code for the other filing statuses.

3.14 (*Game: heads or tails*) Write a program that lets the user guess whether a flipped coin displays the head or the tail. 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 3.9, Lottery.py, to generate a three-digit lottery 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 the digits in the user input match all the 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 (*Reverse number)* Write a program that prompts the user to enter a four-digit integer and displays the number in reverse order.

***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.

***3.18** (*Financials: currency exchange*) Write a program that prompts the user to enter the currency exchange rate between U.S. dollars and Chinese Renminbi (RMB). Prompt the user to enter 0 to convert from U.S. dollars to Chinese RMB and 1 for vice versa. 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.

****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 2.9 gives a formula to compute the wind-chill temperature. The formula is valid for temperatures in the range between −58°F and 41°F and for 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

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

where

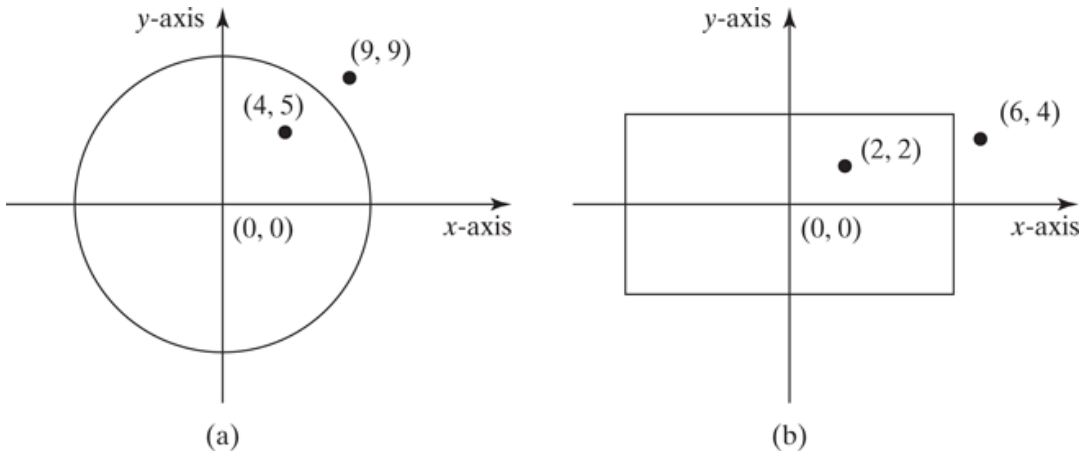
- `h` is the day of the week (0: Saturday, 1: Sunday, 2: Monday, 3: Tuesday, 4: Wednesday, 5: Thursday, 6: Friday).
- `q` is the day of the month.
- `m` is the month (3: March, 4: April, …, 12: December). January and February are counted as months 13 and 14 of the previous year.
- `j` is $\left[\frac{year}{100}\right]$.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

Write a program that prompts the user to enter a year, month, and day of the month, and then it displays the name of the day of the week.

(Hint: Use the // operator for integer division. 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.)

**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.

Figure 3.7



(a) Points inside and outside of the circle; (b) points inside and outside of the rectangle.

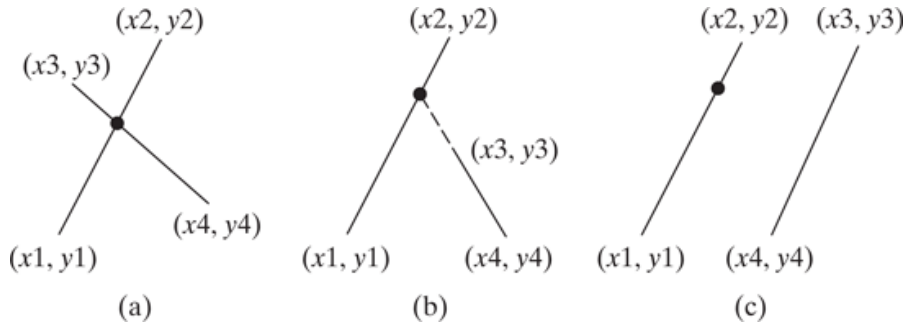
(*Hint*: A point is in the circle if its distance to (`0` , `0`) is less than or equal to `10` . The formula for computing the distance is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Test your program to cover all cases.)

**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.)

**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.

*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 and Figure 3.8b.

Figure 3.8



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

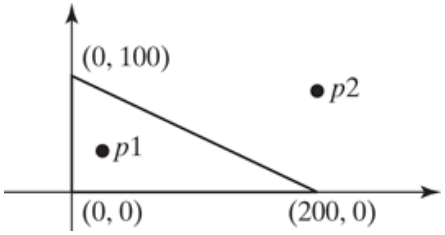
The intersecting point of the two lines can be found by solving the following linear equation:

$$(y_1 - y_2)x - (x_1 - x_2)y = (y_1 - y_2)x_1 - (x_1 - x_2)y_1$$
$$(y_3 - y_4)x - (x_3 - x_4)y = (y_3 - y_4)x_3 - (x_3 - x_4)y_3$$

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.

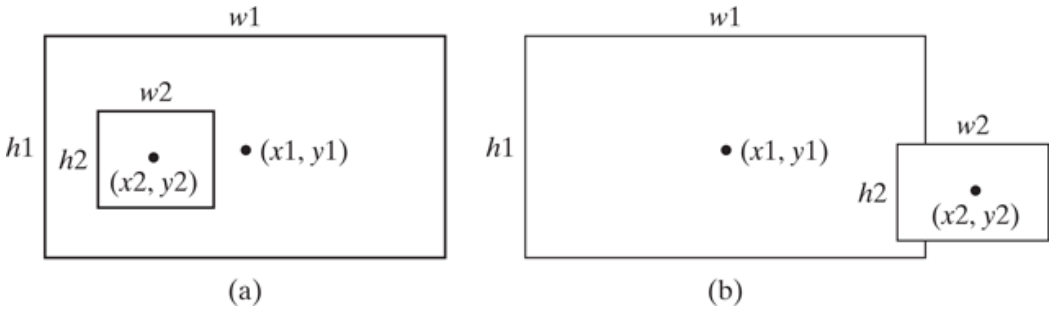
3.26 (*Palindrome number*) Write a program that prompts the user to enter a three-digit integer and determines whether it is a palindrome number. A number is palindrome if it reads the same from right to left and from left to right.

****3.27** (*Geometry: points in triangle?*) Suppose a right triangle is placed in a plane as shown below. The right-angle point is at (0, 0), and the other two points are 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.



****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.

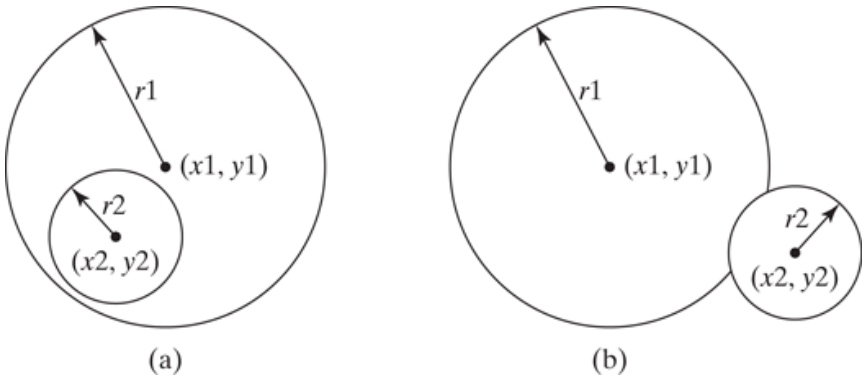
Figure 3.9



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

**3.29 (*Geometry: two* circles) Wrfite 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.)

Figure 3.10



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

*3.30 (*Current time)* Revise Programming Exercise 2.18 to display the hour using a 12-hour clock.

*3.31 (*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 side of the line, on the right side of the line, or on the same line (see Figure 3.11):

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

Figure 3.11

(a) p2 is on the left side of the line. (b) p2 is on the right side 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 side of the line from p0 to p1, on the right side, or on the same line.

***3.32** (*Geometry: point on line segment*) Programming Exercise 3.31 shows how to test whether a point is on an unbounded line. Revise Programming Exercise 3.31 to test whether a point in 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.

****3.33** (*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 checksum. Your program should read the input as an integer. If the integer starts with 0s, don't enter these zeros in the input.

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

***3.35** (*Turtle: point position*) Write a program that prompts the user to enter three points p0, p1, and p2, and displays a message to indicate whether p2 is on the left side, the right side, or on the line from p0 to p1, as shown in Figure 3.12. See Programming Exercise 3.31 for determining the point position.

Figure 3.12

The program displays the point position graphically.

(Screenshots courtesy of Apple.)

**3.36 (*Turtle: point in a circle?*) Modify LiveExample 3.10 to let the program randomly generate a point within the square whose center is the same as the circle center and whose side is the diameter of the circle. Draw the circle and the point. Display a message to indicate whether the point is inside the circle.

**3.37 (*Turtle: 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 `100` and height `50` . Display the point, the rectangle, and a message indicating whether the point is inside the rectangle in the window, as shown in Figure 3.13.

Figure 3.13

The program displays the rectangle, a point, and a message whether a point is in or outside of the rectangle.

(Screenshots courtesy of Apple.)

*3.38 (*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 3.14.

Figure 3.14

The program checks whether a rectangle is inside another one, overlaps another one, or does not overlap.

(Screenshots courtesy of Apple.)

*3.39 (*Turtle: 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 3.15.

Figure 3.15

The program displays two circles and a status message

(Screenshots courtesy of Apple.)

Note

Additional programming exercises with solutions are provided to the instructors on the Instructor Resource Website.