Sections 5.2–5.7

***5.1** (*Count positive and negative numbers and compute the average of numbers*) Write a program that reads an unspecified number of integers, determines how many positive and negative values have been read, and computes the total and average of the input values (not counting zeros). Your program ends with the input **0**. Display the average as a floating-point number. Here is a sample run:

```
Enter an integer, the input ends if it is 0: 1 2 -1 3 0 Enter
The number of positives is 3
The number of negatives is 1
The total is 5.0
The average is 1.25
```

```
Enter an integer, the input ends if it is 0: 0 Enter
No numbers are entered except 0
```

5.2 (*Repeat additions*) Listing 5.4, SubtractionQuizLoop.java, generates five random subtraction questions. Revise the program to generate ten random addition questions for two integers between **1** and **15**. Display the correct count and test time.

5.3 (*Conversion from kilograms to pounds*) Write a program that displays the following table (note that **1** kilogram is **2.2** pounds):

```
Kilograms     Pounds
1                2.2
3                6.6
...
197            433.4
199            437.8
```

5.4 (*Conversion from miles to kilometers*) Write a program that displays the following table (note that 1 mile is 1.609 kilometers):

```
Miles       Kilometers
1           1.609
2           3.218
...
9           14.481
10          16.090
```

5.5 (*Conversion from kilograms to pounds and pounds to kilograms*) Write a program that displays the following two tables side by side:

```
Kilograms  Pounds    |      Pounds      Kilograms
1             2.2    |      20               9.09
3             6.6    |      25              11.36
...
197         433.4    |      510            231.82
199         437.8    |      515            234.09
```

5.6 (*Conversion from miles to kilometers*) Write a program that displays the following two tables side by side:

```
Miles        Kilometers   |  Kilometers        Miles
1            1.609        |  20                12.430
2            3.218        |  25                15.538
...
9            14.481       |  60                37.290
10           16.090       |  65                40.398
```

****5.7** (*Financial application: compute future tuition*) Suppose that the tuition for a university is \$10,000 this year and increases 5% every year. In one year, the tuition will be \$10,500. Write a program that computes the tuition in ten years and the total cost of four years' worth of tuition after the tenth year.

5.8 (*Find the highest score*) Write a program that prompts the user to enter the number of students and each student's name and score, and finally displays the name of the student with the highest score.

***5.9** (*Find the two highest scores*) Write a program that prompts the user to enter the number of students and each student's name and score, and finally displays the student with the highest score and the student with the second-highest score.

5.10 (*Find numbers divisible by 5 and 6*) Write a program that displays all the numbers from 100 to 1,000, ten per line, that are divisible by 5 and 6. Numbers are separated by exactly one space.

5.11 (*Find numbers divisible by 5 or 6, but not both*) Write a program that displays all the numbers from 100 to 200, ten per line, that are divisible by 5 or 6, but not both. Numbers are separated by exactly one space.

5.12 (*Find the smallest* `n` such that $n^2 > 12{,}000$) Use a `while` loop to find the smallest integer `n` such that n^2 is greater than 12,000.

5.13 (*Find the largest* `n` such that $n^3 < 12{,}000$) Use a `while` loop to find the largest integer `n` such that n^3 is less than 12,000.

Sections 5.8–5.10

***5.14** (*Compute the greatest common divisor*) Another solution for Listing 5.9 to find the greatest common divisor of two integers `n1` and `n2` is as follows: First find `d` to be the minimum of `n1` and `n2`, then check whether `d`, `d-1`, `d-2`, . . . , `2`, or `1` is a divisor for both `n1` and `n2` in this order. The first such common divisor is the greatest common divisor for `n1` and `n2`. Write a program that prompts the user to enter two positive integers and displays the gcd.

***5.15** (*Display the ASCII character table*) Write a program that prints the characters in the ASCII character table from `!` to `~`. Display ten characters per line. The ASCII table is shown in Appendix B. Characters are separated by exactly one space.

***5.16** (*Find the factors of an integer*) Write a program that reads an integer and displays all its smallest factors in increasing order. For example, if the input integer is `120`, the output should be as follows: `2`, `2`, `2`, `3`, `5`.

****5.17** (*Display pyramid*) Write a program that prompts the user to enter an integer from `1` to `15` and displays a pyramid, as shown in the following sample run:

```
Enter the number of lines: 7 Enter
                        1
                     2  1  2
                  3  2  1  2  3
               4  3  2  1  2  3  4
            5  4  3  2  1  2  3  4  5
         6  5  4  3  2  1  2  3  4  5  6
      7  6  5  4  3  2  1  2  3  4  5  6  7
```

***5.18** (*Display four patterns using loops*) Use nested loops that display the following patterns in four separate programs:

```
Pattern A         Pattern B         Pattern C         Pattern D
1                 1 2 3 4 5 6                 1       1 2 3 4 5 6
1 2               1 2 3 4 5                 2 1         1 2 3 4 5
1 2 3             1 2 3 4                 3 2 1           1 2 3 4
1 2 3 4           1 2 3                 4 3 2 1             1 2 3
1 2 3 4 5         1 2                 5 4 3 2 1               1 2
1 2 3 4 5 6       1                 6 5 4 3 2 1                 1
```

****5.19** (*Display numbers in a pyramid pattern*) Write a nested `for` loop that prints the following output:

```
                              1
                          1   2   1
                      1   2   4   2   1
                  1   2   4   8   4   2   1
              1   2   4   8  16   8   4   2   1
          1   2   4   8  16  32  16   8   4   2   1
      1   2   4   8  16  32  64  32  16   8   4   2   1
  1   2   4   8  16  32  64 128  64  32  16   8   4   2   1
```

***5.20** (*Display prime numbers between 2 and 1,000*) Modify Listing 5.15 to display all the prime numbers between 2 and 1,000, inclusive. Display eight prime numbers per line. Numbers are separated by exactly one space.

Comprehensive

****5.21** (*Financial application: compare loans with various interest rates*) Write a program that lets the user enter the loan amount and loan period in number of years and displays the monthly and total payments for each interest rate starting from 5% to 8%, with an increment of 1/8. Here is a sample run:

```
Loan Amount: 10000 ↵Enter
Number of Years: 5 ↵Enter
Interest Rate      Monthly Payment      Total Payment

5.000%             188.71               11322.74
5.125%             189.29               11357.13
5.250%             189.86               11391.59
...
7.875%             202.17               12129.97
8.000%             202.76               12165.84
```

For the formula to compute monthly payment, see Listing 2.9, ComputeLoan.java.

****5.22** (*Financial application: loan amortization schedule)* The monthly payment for a given loan pays the principal and the interest. The monthly interest is computed by multiplying the monthly interest rate and the balance (the remaining principal). The principal paid for the month is therefore the monthly payment minus the monthly interest. Write a program that lets the user enter the loan amount,

VideoNote
Display loan schedule

number of years, and interest rate and displays the amortization schedule for the loan. Here is a sample run:

```
Loan Amount: 10000 ↵Enter
Number of Years: 1 ↵Enter
Annual Interest Rate: 7 ↵Enter

Monthly Payment: 865.26
Total Payment: 10383.21

Payment#     Interest     Principal     Balance
1            58.33        806.93        9193.07
2            53.62        811.64        8381.43
...
11           10.0         855.26         860.27
12            5.01        860.25           0.01
```

Note
The balance after the last payment may not be zero. If so, the last payment should be the normal monthly payment plus the final balance.

Hint: Write a loop to display the table. Since the monthly payment is the same for each month, it should be computed before the loop. The balance is initially the loan amount. For each iteration in the loop, compute the interest and principal, and update the balance. The loop may look like this:

```
for (i = 1; i <= numberOfYears * 12; i++) {
  interest = monthlyInterestRate * balance;
  principal = monthlyPayment - interest;
  balance = balance - principal;
  System.out.println(i + "\t\t" + interest
    + "\t\t" + principal + "\t\t" + balance);
}
```

***5.23** (*Demonstrate cancellation errors*) A cancellation error occurs when you are manipulating a very large number with a very small number. The large number may cancel out the smaller number. For example, the result of `100000000.0 + 0.000000001` is equal to `100000000.0`. To avoid cancellation errors and obtain more accurate results, carefully select the order of computation. For example, in computing the following series, you will obtain more accurate results by computing from right to left rather than from left to right:

$$1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

Write a program that compares the results of the summation of the preceding series, computing from left to right and from right to left with `n = 50000`.

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

VideoNote
Sum a series

$$\frac{1}{3} + \frac{3}{5} + \frac{5}{7} + \frac{7}{9} + \frac{9}{11} + \frac{11}{13} + \cdots + \frac{95}{97} + \frac{97}{99}$$

****5.25** (*Compute* π) You can approximate π by using the following series:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots + \frac{(-1)^{i+1}}{2i - 1}\right)$$

Write a program that displays the π value for `i = 10000`, `20000`, …, and `100000`.

****5.26** (*Compute e*) You can approximate `e` using the following series:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots + \frac{1}{i!}$$

Write a program that displays the `e` value for `i = 10000`, `20000`, …, and `100000`. (*Hint*: Because $i! = i \times (i - 1) \times \ldots \times 2 \times 1$, then

$$\frac{1}{i!} \text{ is } \frac{1}{i(i-1)!}$$

Initialize `e` and `item` to be `1` and keep adding a new `item` to `e`. The new item is the previous item divided by `i` for `i = 2`, `3`, `4`, ...)

****5.27** (*Display leap years*) Write a program that displays all the leap years, ten per line, from 101 to 2100, separated by exactly one space. Also display the number of leap years in this period.

****5.28** (*Display the first days of each month*) Write a program that prompts the user to enter the year and first day of the year, and displays the first day of each month in the year. For example, if the user entered the year `2013`, and `2` for Tuesday, January 1, 2013, your program should display the following output:

```
January 1, 2013 is Tuesday
...
December 1, 2013 is Sunday
```

****5.29** (*Display calendars*) Write a program that prompts the user to enter the year and first day of the year and displays the calendar table for the year on the console. For example, if the user entered the year `2013`, and `2` for Tuesday, January 1, 2013, your program should display the calendar for each month in the year, as follows:

January 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

December 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

***5.30** (*Financial application: compound value*) Suppose you save $100 *each* month into a savings account with the annual interest rate 5%. So, 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 an amount (e.g., `100`), the annual interest rate (e.g., `5`), and the number of months (e.g., `6`) and displays the amount in the savings account after the given month.

***5.31** (*Financial application: compute CD value*) Suppose you put $10,000 into a CD with an annual percentage yield of 5.75%. After one month, the CD is worth

```
10000 + 10000 * 5.75 / 1200 = 10047.92
```

After two months, the CD is worth

```
10047.91 + 10047.91 * 5.75 / 1200 = 10096.06
```

After three months, the CD is worth

```
10096.06 + 10096.06 * 5.75 / 1200 = 10144.44
```

and so on.

Write a program that prompts the user to enter an amount (e.g., `10000`), the annual percentage yield (e.g., `5.75`), and the number of months (e.g., `18`) and displays a table as shown in the sample run.

```
Enter the initial deposit amount: 10000 ↵Enter
Enter annual percentage yield: 5.75 ↵Enter
Enter maturity period (number of months): 18 ↵Enter

Month  CD Value
1      10047.92
2      10096.06
...
17     10846.57
18     10898.54
```

****5.32** (*Game: lottery*) Revise Listing 3.8, Lottery.java, to generate a lottery of a two-digit number. The two digits in the number are distinct. (*Hint*: Generate the first digit. Use a loop to continuously generate the second digit until it is different from the first digit.)

****5.33** (*Perfect number*) A positive integer is called a *perfect number* if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number because `6 = 3 + 2 + 1`. The next is `28 = 14 + 7 + 4 + 2 + 1`. There are four perfect numbers less than 10,000. Write a program to find all these four numbers.

*****5.34** (*Game: scissor, rock, paper*) Programming Exercise 3.17 gives a program that plays the scissor-rock-paper game. Revise the program to let the user continuously play until either the user or the computer wins more than two times than its opponent.

***5.35** (*Summation*) Write a program to compute the following summation.

$$\frac{1}{1+\sqrt{2}} + \frac{1}{\sqrt{2}+\sqrt{3}} + \frac{1}{\sqrt{3}+\sqrt{4}} + \ldots + \frac{1}{\sqrt{624}+\sqrt{625}}$$

****5.36** (*Business application: checking ISBN*) Use loops to simplify Programming Exercise 3.9.

****5.37** (*Decimal to binary*) Write a program that prompts the user to enter a decimal integer and displays its corresponding binary value. Don't use Java's `Integer.toBinaryString(int)` in this program.

****5.38** (*Decimal to octal*) Write a program that prompts the user to enter a decimal integer and displays its corresponding octal value. Don't use Java's `Integer.toOctalString(int)` in this program.

***5.39** (*Financial application: find the sales amount*) You have just started a sales job in a department store. Your pay consists of a base salary and a commission. The base salary is $5,000. The scheme shown below is used to determine the commission rate.

Sales Amount	Commission Rate
$0.01–$5,000	8 percent
$5,000.01–$10,000	10 percent
$10,000.01 and above	12 percent

Note that this is a graduated rate. The rate for the first $5,000 is at 8%, the next $5000 is at 10%, and the rest is at 12%. If the sales amount is 25,000, the commission is 5,000 * 8% + 5,000 * 10% + 15,000 * 12% = 2,700.

Your goal is to earn $30,000 a year. Write a program that finds the minimum sales you have to generate in order to make $30,000.

5.40 (*Simulation: heads or tails*) Write a program that simulates flipping a coin one million times and displays the number of heads and tails.

***5.41** (*Occurrence of max numbers*) Write a program that reads integers, finds the largest of them, and counts its occurrences. Assume that the input ends with number `0`. Suppose that you entered `3 5 2 5 5 5 0`; the program finds that the largest is `5` and the occurrence count for `5` is `4`.

(*Hint*: Maintain two variables, `max` and `count`. `max` stores the current max number, and `count` stores its occurrences. Initially, assign the first number to `max` and `1` to `count`. Compare each subsequent number with `max`. If the number is greater than `max`, assign it to `max` and reset `count` to `1`. If the number is equal to `max`, increment `count` by `1`.)

```
Enter numbers: 3 5 2 5 5 5 0 ↵Enter
The largest number is 5
The occurrence count of the largest number is 4
```

***5.42** (*Financial application: find the sales amount*) Rewrite Programming Exercise 5.39 as follows:

- Use a `for` loop instead of a `do-while` loop.
- Let the user enter `COMMISSION_SOUGHT` instead of fixing it as a constant.

***5.43** (*Math: combinations*) Write a program that displays all possible combinations for picking two numbers from integers `1` to `7`. Also display the total number of all combinations.

```
1 2
1 3
...
...

The total number of all combinations is 21
```

***5.44** (*Computer architecture: bit-level operations*) A `short` value is stored in `16` bits. Write a program that prompts the user to enter a short integer and displays the `16` bits for the integer. Here are sample runs:

```
Enter an integer: 5 ↵Enter
The bits are 0000000000000101
```

```
Enter an integer: -5 ↵Enter
The bits are 1111111111111011
```

(*Hint*: You need to use the bitwise right shift operator (`>>`) and the bitwise `AND` operator (`&`), which are covered in Appendix G, Bitwise Operations.)

****5.45** (*Statistics: compute mean and standard deviation*) In business applications, you are often asked to compute the mean and standard deviation of data. The mean is simply the average of the numbers. The standard deviation is a statistic that tells you how tightly all the various data are clustered around the mean in a set of data. For example, what is the average age of the students in a class? How close are the ages? If all the students are the same age, the deviation is 0.

Write a program that prompts the user to enter ten numbers, and displays the mean and standard deviations of these numbers using the following formula:

$$\text{mean} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \cdots + x_n}{n} \qquad \text{deviation} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n-1}}$$

Here is a sample run:

```
Enter ten numbers: 1 2 3 4.5 5.6 6 7 8 9 10 ↵Enter
The mean is 5.61
The standard deviation is 2.99794
```

***5.46** (*Reverse a string*) Write a program that prompts the user to enter a string and displays the string in reverse order.

```
Enter a string: ABCD ↵Enter
The reversed string is DCBA
```

***5.47** (*Business: check ISBN-13*) **ISBN-13** is a new standard for indentifying books. It uses 13 digits $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}d_{11}d_{12}d_{13}$. The last digit d_{13} is a checksum, which is calculated from the other digits using the following formula:

$$10 - (d_1 + 3d_2 + d_3 + 3d_4 + d_5 + 3d_6 + d_7 + 3d_8 + d_9 + 3d_{10} + d_{11} + 3d_{12})\%10$$

If the checksum is **10**, replace it with **0**. Your program should read the input as a string. Here are sample runs:

```
Enter the first 12 digits of an ISBN-13 as a string: 978013213080 ↵Enter
The ISBN-13 number is 9780132130806
```

```
Enter the first 12 digits of an ISBN-13 as a string: 978013213079 ↵Enter
The ISBN-13 number is 9780132130790
```

```
Enter the first 12 digits of an ISBN-13 as a string: 97801320 ↵Enter
97801320 is an invalid input
```

***5.48** (*Process string*) Write a program that prompts the user to enter a string and displays the characters at odd positions. Here is a sample run:

```
Enter a string: Beijing Chicago ↵Enter
BiigCiao
```

***5.49** (*Count vowels and consonants*) Assume letters `A`, `E`, `I`, `O`, and `U` as the vowels. Write a program that prompts the user to enter a string and displays the number of vowels and consonants in the string.

```
Enter a string: Programming is fun ↵Enter
The number of vowels is 5
The number of consonants is 11
```

***5.50** (*Count uppercase letters*) Write a program that prompts the user to enter a string and displays the number of the uppercase letters in the string.

```
Enter a string: Welcome to Java ↵Enter
The number of uppercase letters is 2
```

***5.51** (*Longest common prefix*) Write a program that prompts the user to enter two strings and displays the largest common prefix of the two strings. Here are some sample runs:

```
Enter the first string: Welcome to C++ ↵Enter
Enter the second string: Welcome to programming ↵Enter
The common prefix is Welcome to
```

```
Enter the first string: Atlanta ↵Enter
Enter the second string: Macon ↵Enter
Atlanta and Macon have no common prefix
```