From: C++: Early Objects, Gaddis et al, 9th Ed

CHAPTERS 1–8
PROGRAMMING CHALLENGES

Find the Error

33. The following pseudocode algorithm has an error. It is supposed to use values input for a rectangular room's length and width to calculate and display its area. Find the error.

```
area = width × length.
Display "What is the room's width?".
Input width.
Display "What is the room's length?".
Input length.
Display area.
```

Soft Skills

Before a programmer can design a program he or she must have some basic knowledge about the domain, or area, the program will deal with and must understand exactly what it is that the client wants the program to do. Otherwise the final program may not work correctly or may not meet the client's needs.

34. Suppose one of your friends, who paints the insides of houses, has asked you to develop a program that determines and displays how much paint is needed to paint a room if the length and width of the room are input. What information are you lacking that you need to write this program? Write at least three questions that you would need to ask your friend before starting the project.

Programming Challenges – CHAPTER 1

1. **1. Candy Bar Sales**

VideoNote
Solving the Candy Bar Sales Problem

Using Program 1-1 as an example, write a program that calculates how much a student organization earns during its fund-raising candy sale. The program should prompt the user to enter the number of candy bars sold and the amount the organization earns for each bar sold. It should then calculate and display the total amount earned.

1. **2. Baseball Costs**

Using Program 1-1 as an example, write a program that calculates how much a Little League baseball team spent last year to purchase new baseballs. The program should prompt the user to enter the number of baseballs purchased and the cost of each baseball. It should then calculate and display the total amount spent to purchase the baseballs.

1. **3. Flower Garden**

Write a program that calculates how much a garden center spent to make a flower garden display. The program should prompt the user to enter the cost of the soil, the flower seeds, and the fence. It should then calculate and display the total amount spent.

Programming Challenges – CHAPTER 2

2.1. **Sum of Two Numbers**

Write a program that stores the integers 50 and 100 in variables and stores the sum of these two in a variable named `total`. Display the total on the screen.

2.2. **Sales Prediction**

The East Coast sales division of a company generates 58 percent of total sales. Based on that percentage, write a program that will predict how much the East Coast division will generate if the company has $8.6 million in sales this year. Display the result on the screen.

2.3. **Sales Tax**

Write a program that computes the total sales tax on a $95 purchase. Assume the state sales tax is 6.5 percent and the county sales tax is 2 percent. Display the purchase price, state tax, county tax, and total tax amounts on the screen.

2.4. **Restaurant Bill**

Write a program that computes the tax and tip on a restaurant bill for a patron with a $44.50 meal charge. The tax should be 6.75 percent of the meal cost. The tip should be 15 percent of the total after adding the tax. Display the meal cost, tax amount, tip amount, and total bill on the screen.

VideoNote
Solving the Restaurant Bill Problem

2.5. **Miles per Gallon**

A car holds 16 gallons of gasoline and can travel 312 miles before refueling. Write a program that calculates the number of miles per gallon the car gets. Display the result on the screen.

2.6. **Distance per Tank of Gas**

A car with a 20 gallon gas tank averages 23.5 miles per gallon when driven in town and 28.9 miles per gallon when driven on the highway. Write a program that calculates and displays the distance the car can travel on one tank of gas when driven in town and when driven on the highway.

2.7. **Number of Acres**

One acre of land is equivalent to 43,450 square feet. Write a program that calculates and displays the number of acres in a tract of land whose size is 869 × 360 feet.

2.8. **Land Calculation**

In the United States, land is often measured in square feet. In many other countries, it is measured in square meters. One acre of land is equivalent to 43,560 square feet. A square meter is equivalent to 10.7639 square feet. Write a program that computes and displays the number of square feet and the number of square meters in $\frac{1}{2}$ acre of land.

Hint: Because a square meter is larger than a square foot, there will be fewer square meters in $\frac{1}{2}$ acre than there are square feet.

2.9. **Circuit Board Price**

An electronics company makes circuit boards that cost $14.95 apiece to produce. Write a program to determine how much the company should sell them for if it wants to make a 35 percent profit. Display the result on the screen.

2.10. **Personal Information**

Write a program that displays the following information, each on a separate line:

Your name
Your address, with city, state, and zip code
Your telephone number
Your college major

Use only a single cout statement to display all of this information.

2.11. **Triangle Pattern**

Write a program that displays the following pattern on the screen:

```
   *
  ***
 *****
*******
```

2.12. **Diamond Pattern**

Write a program that displays the following pattern on the screen:

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

2.13. **Pay Period Gross Pay**

A particular employee earns $39,000 annually. Write a program that determines and displays what the amount of his gross pay will be for each pay period if he is paid twice a month (24 pay checks per year) and if he is paid bi-weekly (26 checks per year).

2.14. **Basketball Player Height**

The star player of a high school basketball team is 74 inches tall. Write a program to compute and display the height in feet/inches form.

Hint: Try using the modulus and integer divide operations.

2.15. Stock Loss

Kathryn bought 750 shares of stock at a price of $35.00 per share. A year later she sold them for just $31.15 per share. Write a program that calculates and displays the following:

- The total amount paid for the stock.
- The total amount received from selling the stock.
- The total amount of money she lost.

2.16. Energy Drink Consumption

A soft drink company recently surveyed 16,500 of its customers and found that approximately 15 percent of those surveyed purchase one or more energy drinks per week. Of those customers who purchase energy drinks, approximately 52 percent of them purchase citrus flavored energy drinks. Write a program that displays the following:

- The approximate number of customers in the survey who purchase one or more energy drinks per week.
- The approximate number of customers in the survey who purchase citrus flavored energy drinks.

2.17. Past Ocean Levels

The Earth's ocean levels have risen an average of 1.8 millimeters per year over the past century. Write a program that computes and displays the number of centimeters and number of inches the oceans rose during this time. One millimeter is equivalent to 0.1 centimeters. One centimeter is equivalent to 0.3937 inches.

2.18. Future Ocean Levels

During the past decade ocean levels have been rising faster than in the past, an average of approximately 3.1 millimeters per year. Write a program that computes how much ocean levels are expected to rise during the next 15 years if they continue rising at this rate. Display the answer in both centimeters and inches.

2.19. Annual High Temperatures

The average July high temperature is 85 degrees Fahrenheit in New York City, 88 degrees in Denver, and 106 degrees in Pheonix. Write a program that calculates and reports what the new average high July temperature would be for each of these cities if temperatures to rise by 2 percent.

2.20. How Much Paint

A particular brand of paint covers 340 square feet per gallon. Write a program to determine and report approximately how many gallons of paint will be needed to paint two coats on a wooden fence that is 6 feet high and 100 feet long.

Soft Skills

Often programmers work in teams with other programmers to develop a piece of software. It is important that the team members be able to communicate clearly with one another.

30. Suppose you and a fellow student have been assigned to develop together the pizza cost program described in Problem 25. You have developed a pseudocode algorithm for the program and emailed it to your partner, but he does not understand how it works. Write a paragraph that you might email back clearly explaining how the algorithm works, what steps must be done, why they must be done in a particular order, and why the calculations you have specified in the pseudocode are the correct ones to use. Write your answer using full English sentences with correct spelling and grammar.

Programming Challenges - CHAPTER 3

3,1. Miles per Gallon

Write a program that calculates a car's gas mileage. The program should ask the user to enter the number of gallons of gas the car can hold and the number of miles it can be driven on a full tank. It should then calculate and display the number of miles per gallon the car gets.

3,2. Stadium Seating

VideoNote
Solving the Stadium Seating Problem

There are three seating categories at a stadium. For a softball game, Class A seats cost $15, Class B seats cost $12, and Class C seats cost $9. Write a program that asks how many tickets for each class of seats were sold, then displays the amount of income generated from ticket sales. Format your dollar amount in a fixed-point notation with two decimal points and make sure the decimal point is always displayed.

3, 3. Housing Costs

Write a program that asks the user to enter their *monthly* costs for each of the following housing-related expenses:

- rent or mortgage payment
- utilities
- phones
- cable

The program should then display the total monthly cost of these expenses and the total annual cost of these expenses.

3, 4. How Much Insurance?

Many financial experts advise property owners to insure their homes or buildings for at least 80 percent of the amount it would cost to replace the structure. Write a program that asks the user to enter the replacement cost of a building and then displays the minimum amount of insurance that should be purchased for the property.

7, **5. Batting Average**

Write a program to find a baseball player's batting average. The program should ask the user to enter the number of times the player was at bat and the number of hits earned. It should then display the batting average to four decimal places.

7, **6. Test Average**

Write a program that asks for five test scores. The program should calculate the average test score and display it. The number displayed should be formatted in fixed-point notation, with one decimal point of precision.

7, **7. Average Rainfall**

Write a program that calculates the average monthly rainfall for three months. The program should ask the user to enter the name of each month, such as June or July, and the amount of rain (in inches) that fell that month. The program should display a message similar to the following:

```
The average monthly rainfall for June, July, and August was 6.72 inches.
```

7, **8. Male and Female Percentages**

Write a program that asks the user for the number of males and the number of females registered in a class. The program should compute and report what percentage of the students are males and what percentage are females. Display the output with two decimal points. If you remembered to convert the decimal result of each calculation to percent form when you displayed it, the two values should add up to 100.00 percent.

7, **9. Vacation Days**

Write a program that prompts the users to enter the number of days they plan to spend on their next vacation. Then compute and report how long that is in hours, in minutes, and in seconds.

7, **10. Box Office**

A movie theater only keeps 80 percent of the revenue earned from ticket sales. The other 20 percent goes to the distibutor. Write a program that calculates a theater's gross and net box office revenue for a night. The program should ask for the name of the movie, and how many adult and child tickets were sold. (The price of an adult ticket is $10 and a child's ticket is $6.) It should display a report similar to the following:

```
Movie Name:                     "Wheels of Fury"
Adult Tickets Sold:             382
Child Tickets Sold:             127
Gross Box Office Revenue:       $ 4582.00
Amount Paid to Distributor:    -$  916.40
Net Box Office Revenue:         $ 3665.60
```

7, 11. How Many Widgets?

The Yukon Widget Company manufactures widgets that weigh 12.5 pounds each. Write a program that calculates how many widgets are stacked on a pallet, based on the total weight of the pallet. The program should ask the user how much the pallet weighs by itself and with the widgets stacked on it. It should then calculate and display the number of widgets stacked on the pallet.

7, 12. How many Calories?

A bag of cookies holds 30 cookies. The calorie information on the bag claims that there are 10 "servings" in the bag and that a serving equals 240 calories. Write a program that asks the user to input how many cookies they actually ate and then reports how many total calories were consumed.

7, 13. Ingredients Adjuster

A cookie recipe calls for the following ingredients:

- 1.5 cups of sugar
- 1 cup of butter
- 2.75 cups of flour

The recipe produces 48 cookies with these amounts of the ingredients. Write a program that asks the user how many cookies he or she wants to make and then displays the number of cups of each ingredient needed for the specified number of cookies.

7, 14. Celsius to Fahrenheit

Write a program that converts Celsius temperatures to Fahrenheit temperatures. The formula is

$$F = \frac{9}{5}C + 32$$

where F is the Fahrenheit temperature and C is the Celsius temperature. The program should prompt the user to input a Celsius temperature and should display the corresponding Farenheit temperature.

7, 15. Currency

Write a program that will convert U.S. dollar amounts to Japanese yen and to euros, storing the conversion factors in the constant variables YEN_PER_DOLLAR and EUROS_PER_DOLLAR. To get the most up-to-date exchange rates, search the Internet using the term "currency exchange rate" or "currency converter." If you cannot find the most recent exchange rates, use the following:

1 Dollar = 120.005 Yen
1 Dollar = .881 Euros

3, **16. Monthly Sales Tax**

A retail company must file a monthly sales tax report listing the sales for the month and the amount of sales tax collected. Write a program that asks for the month, the year, and the total amount collected at the cash register (that is, sales plus sales tax). Assume the state sales tax is 4 percent and the county sales tax is 2 percent.

If the total amount collected is known and the total sales tax is 6 percent, the amount of product sales may be calculated as

$$S = \frac{T}{1.06}$$

where S is the product sales and T is the total income (product sales plus sales tax).

The program should display a report similar to the following:

```
Month: August 2016
--------------------
Total Collected:    $ 26572.89
Sales:              $ 25068.76
County Sales Tax:   $   501.38
State Sales Tax:    $  1002.75
Total Sales Tax:    $  1504.13
```

3. **17. Property Tax**

Madison County collects property taxes on the assessed value of property, which is 60 percent of its actual value. For example, if a house is valued at $158,000, its assessed value is $94,800. This is the amount the homeowner pays tax on. At last year's tax rate of $2.64 for each $100 of assessed value, the annual property tax for this house would be $2502.72. Write a program that asks the user to input the actual value of a piece of property and the current tax rate for each $100 of assessed value. The program should then calculate and report how much annual property tax the homeowner will be charged for this property.

3. **18. Senior Citizen Property Tax**

Madison County provides a $5000 homeowner exemption for senior citizens. For example, if their house is valued at $158,000 its assessed value would be $94,800, as explained above. However they would only pay tax on $89,800. At last year's tax rate of $2.64 for each $100 of assessed value, their property tax would be $2370.72. In addition to the tax break, senior citizens are allowed to pay their property tax in four equal payments. The quarterly payment due on this property would be $592.68. Write a program that asks the user to input the actual value of a piece of property and the current tax rate for each $100 of assessed value. The program should then calculate and report how much annual property tax a senior homeowner will be charged for this property and what their quarterly tax bill will be.

3, **19. Math Tutor**

Write a program that can be used as a math tutor for a young student. The program should display two random numbers between 1 and 9 to be added, such as

```
 2
+1
```

After the student has entered an answer and pressed the [Enter] key, the program should display the correct answer so the student can see if his or her answer is correct.

3, **20. Interest Earned**

Assuming there are no deposits other than the original investment, the balance in a savings account after one year may be calculated as

$$\text{Amount} = \text{Principal} * \left(1 + \frac{\text{Rate}}{T}\right)^{T}$$

- where Principal is the balance in the account
- Rate is the annual interest rate,
- *T* is the number of times the interest is compounded during a year (e.g., *T* is 4 if the interest is compounded quarterly).

Write a program that asks for the principal, the interest rate, and the number of times the interest is compounded. It should display a report similar to the following:

```
Interest Rate:           4.25%
Times Compounded:           12
Principal:            $ 1000.00
Interest:             $   43.33
Final balance:        $ 1043.33
```

3, **21. Monthly Payments**

The monthly payment on a loan may be calculated by the following formula:

$$\text{Payment} = \frac{\text{Rate}*(1+\text{Rate})^{N}}{(1+\text{Rate})^{N}-1} * \text{L}$$

- Rate is the monthly interest rate, which is the annual interest rate divided by 12. (A 12 percent annual interest would be 1 percent monthly interest.)
- *N* is the number of payments
- *L* is the amount of the loan.

Write a program that asks for these values and displays a report similar to the following:

```
Loan Amount:              $ 10000.00
Monthly Interest Rate:            1%
Number of Payments:               36
Monthly Payment:          $   332.14
Amount Paid Back:         $ 11957.15
Interest Paid:            $  1957.15
```

7c 22. Pizza Slices

Joe's Pizza Palace needs a program to calculate the number of slices a pizza of any size can be divided into. The program should perform the following steps:

A) Ask the user for the diameter of the pizza in inches.
B) Divide the diameter by 2 to get the radius.
C) Calculate the number of slices that may be taken from a pizza of that size if each slice has an area of 14.125 square inches.
D) Display a message telling the number of slices.

The number of square inches in the total pizza can be calculated with this formula:

$$Area = \pi r^2$$

where variable r is the radius of the pizza and π is the Greek letter PI. In your program make PI a named constant with the value 3.14. Display the number of slices as a whole number (i.e., with no decimals).

7d 23. How Many Pizzas?

Modify the program you wrote in Programming Challenge 22 so that it reports the number of pizzas you need to buy for a party if each person attending is expected to eat an average of four slices. The program should ask the user for the number of people who will be at the party and for the diameter of the pizzas to be ordered. It should then calculate and display the number of pizzas to purchase. Because it is impossible to buy a part of a pizza, the number of required pizzas should be displayed as a whole number.

7e 24. Angle Calculator

Write a program that asks the user for an angle, entered in radians. The program should then display the sine, cosine, and tangent of the angle. (Use the `sin`, `cos`, and `tan` library functions to determine these values.) The output should be displayed in fixed-point notation, rounded to four decimal places of precision.

7f 25. Stock Transaction Program

Last month Joe purchased 100 shares of stock for $45.50 per share and paid his stock broker a commission that amounted to 2 percent of the total amount he paid for the stock.

Two months later Joe sold the stock for $47.92 per share and paid his stock broker another commission that amounted to 2 percent of the total amount he received for the stock.

Write a program that displays the following information:

- The amount of money Joe paid for the stock.
- The amount of commission Joe paid his broker when he bought the stock.
- The amount that Joe sold the stock for.
- The amount of commission Joe paid his broker when he sold the stock.
- The amount of profit or loss that Joe had after selling his stock and paying both broker commissions.

B)
```
double num1, num2, quotient;

cout << "Enter a number: ";
cin  >> num1;
cout << "Enter another number: ";
cin  >> num2;

if (num2 == 0)
   cout << "Division by zero is not possible.\n";
   cout << "Please run the program again ";
   cout << "and enter a number besides zero.\n";
else
   quotient = num1 / num2;
   cout << "The quotient of " << num1 <<
   cout << " divided by " << num2 << " is ";
   cout << quotient << endl;
```

C)
```
int testScore;

cout << "Enter your test score and I will tell you\n";
cout << "the letter grade you earned: ";
cin  >> testScore;

if (testScore < 60)
     cout << "Your grade is F.\n";
else if (testScore < 70)
     cout << "Your grade is D.\n";
else if (testScore < 80)
     cout << "Your grade is C.\n";
else if (testScore < 90)
     cout << "Your grade is B.\n";
else
     cout << "That is not a valid score.\n";
else if (testScore <= 100)
     cout << "Your grade is A.\n";
```

D)
```
double testScore;
cout << "Enter your test score and I will tell you\n";
cout << "the letter grade you earned: ";
cin  >> testScore;

switch (testScore)
{ case (testScore < 60.0):
             cout << "Your grade is F.\n";
  case (testScore < 70.0):
             cout << "Your grade is D.\n";
  case (testScore < 80.0):
             cout << "Your grade is C.\n";
  case (testScore < 90.0):
             cout << "Your grade is B.\n";
  case (testScore <= 100.0):
             cout << "Your grade is A.\n";
  default:   cout << "That score isn't valid\n"; }
}
```

Soft Skills

Programmers need to be able to look at alternative approaches to solving a problem and at different ways of implementing a solution, weighing the pros and cons of each. Further, they need to be able to clearly articulate to others why they recommend, or have chosen, a particular solution. Come to class prepared to discuss the following:

36. Sometimes either a `switch` statement or an `if/else if` statement can be used to implement logic that requires branching to different blocks of program code. But the two are not interchangeable.
    - A) Under what circumstances would an `if/else if` statement be a more appropriate choice than a switch statement?
    - B) Under what circumstances would a `switch` statement be a more appropriate choice than an `if/else if` statement?
    - C) Under what circumstances would a set of nested `if/else` statements be more appropriate than either of the other two structures?

Try to come up with at least one example case for each of the three, where it is the best way to implement the desired branching logic.

Programming Challenges ~ CHAPTER 4

4.1. Minimum/Maximum

Write a program that asks the user to enter two numbers. The program should use the conditional operator to determine which number is the smaller and which is the larger.

4.2. Roman Numeral Converter

Write a program that asks the user to enter a number within the range of 1 through 10. Use a switch statement to display the Roman numeral version of that number.

Input Validation: Decide how the program should handle an input that is less than 1 or greater than 10.

4. 3. Magic Dates

The date June 10, 1960, is special because when we write it in the following format, the month times the day equals the year.

6/10/60

Write a program that asks the user to enter a month (in numeric form), a day, and a two-digit year. The program should then determine whether the month times the day is equal to the year. If so, it should display a message saying the date is magic. Otherwise, it should display a message saying the date is not magic.

Input Validation: Think about what legal values the program should accept for month and day.

4,4. Areas of Rectangles

The area of a rectangle is the rectangle's length times its width. Write a program that asks for the length and width of two rectangles. The program should then tell the user which rectangle has the greater area or if the areas are the same.

4. **5. Book Club Points**

An online book club awards points to its customers based on the number of books purchased each month. Points are awarded as follows:

Books Purchased	Points Earned
0	0
1	5
2	15
3	30
4 or more	50

Write a program that asks the user to enter the number of books purchased this month and then displays the number of points awarded.

4. **6. Change for a Dollar Game**

Create a change-counting game that asks the user to enter what coins to use to make exactly one dollar. The program should ask the user to enter the number of pennies, nickels, dimes, and quarters. If the total value of the coins entered is equal to one dollar, the program should congratulate the user for winning the game. Otherwise, the program should display a message indicating whether the amount entered was more or less than one dollar. Use constant variables to hold the coin values.

4. **7. Time Calculator**

VideoNote
Solving the Time Calculator Problem

Write a program that asks the user to enter a number of seconds.

- There are 86400 seconds in a day. If the number of seconds entered by the user is greater than or equal to 86400, the program should display the number of days in that many seconds.
- There are 3600 seconds in an hour. If the number of seconds entered by the user is less than 86400 but is greater than or equal to 3600, the program should display the number of hours in that many seconds.
- There are 60 seconds in a minute. If the number of seconds entered by the user is less than 3600 but is greater than or equal to 60, the program should display the number of minutes in that many seconds.

4. **8. Math Tutor Version 2**

This is a modification of the math tutor Programming Challenge problem in Chapter 3. Write a program that can be used as a math tutor for a young student. The program should display two random numbers between 10 and 50 that are to be added, such as:

```
  24
+ 12
  —
```

The program should then wait for the student to enter the answer. If the answer is correct, a message of congratulations should be printed. If the answer is incorrect, a message should be printed showing the correct answer.

4, 9. Software Sales

A software company sells a package that retails for $199. Quantity discounts are given according to the following table.

Quantity	Discount
10–19	20%
20–49	30%
50–99	40%
100 or more	50%

Write a program that asks for the number of units purchased and computes the total cost of the purchase.

Input Validation: Decide how the program should handle an input of less than 0.

4, 10. Bank Charges

A bank charges $15 per month plus the following check fees for a commercial checking account:

$0.10 each for fewer than 20 checks
$0.08 each for 20–39 checks
$0.06 each for 40–59 checks
$0.04 each for 60 or more checks

Write a program that asks for the number of checks written during the past month, then computes and displays the bank's fees for the month.

Input Validation: Decide how the program should handle an input of less than 0.

4, 11. Geometry Calculator

Write a program that displays the following menu:

```
Geometry Calculator

      1. Calculate the Area of a Circle
      2. Calculate the Area of a Rectangle
      3. Calculate the Area of a Triangle
      4. Quit

Enter your choice (1-4):
```

- If the user enters 1, the program should ask for the radius of the circle and then display its area. Use 3.14159 for π.
- If the user enters 2, the program should ask for the length and width of the rectangle, and then display the rectangle's area.
- If the user enters 3, the program should ask for the length of the triangle's base and its height, and then display its area.
- If the user enters 4, the program should end.

Input Validation: Decide how the program should handle an illegal input for the menu choice or a negative value for any of the other inputs.

4, **12. Color Mixer**

The colors red, blue, and yellow are known as the primary colors because they cannot be made by mixing other colors. When you mix two primary colors, you get a secondary color, as shown here:

When you mix red and blue, you get purple.
When you mix red and yellow, you get orange.
When you mix blue and yellow, you get green.

Write a program that prompts the user to enter the names of two primary colors to mix. If the user enters anything other than "red," "blue," or "yellow," the program should display an error message. Otherwise, the program should display the name of the secondary color that results.

4, **13. Running the Race**

Write a program that asks for the names of three runners and the time it took each of them to finish a race. The program should display who came in first, second, and third place. Think about how many test cases are needed to verify that your problem works correctly. (That is, how many different finish orders are possible?)

Input Validation: Only allow the program to accept positive numbers for the times.

4, **14. Personal Best**

Write a program that asks for the name of a pole vaulter and the dates and vault heights (in meters) of the athlete's three best vaults. It should then report in height order (best first), the date on which each vault was made, and its height.

4, **15. February Days**

The month of February normally has 28 days. But if it is a *leap year*, February has 29 days. Write a program that asks the user to enter a year. The program should then display the number of days in February that year. Use the following criteria to identify leap years:

1. Determine whether the year is divisible by 100. If it is, then it is a leap year if and if only it is also divisible by 400. For example, 2000 is a leap year but 2100 is not.
2. If the year is not divisible by 100, then it is a leap year if and if only it is divisible by 4. For example, 2008 is a leap year but 2009 is not.

Here is a sample run of the program:

```
Enter a year: 2020[Enter]
In 2020 February has 29 days.
```

4, **16. Body Mass Index**

Write a program that calculates and displays a person's body mass index (BMI). The BMI is often used to determine whether a person with a sedentary lifestyle is overweight or underweight for his or her height. A person's BMI is calculated with the following formula:

$$\text{BMI} = \text{weight} \times 703/\text{height}^2$$

where weight is measured in pounds and height is measured in inches.

The program should display a message indicating whether the person has optimal weight, is underweight, or is overweight. A sedentary person's weight is considered to be optimal if his or her BMI is between 18.5 and 25. If the BMI is less than 18.5, the person is considered to be underweight. If the BMI value is greater than 25, the person is considered to be overweight.

4. 17. Fat Gram Calculator

Write a program that asks for the number of calories and fat grams in a food. The program should display the percentage of calories that come from fat. If the calories from fat are less than 30 percent of the total calories of the food, it should also display a message indicating the food is low in fat.

One gram of fat has 9 calories, so

```
Calories from fat = fat grams * 9
```

The percentage of calories from fat can be calculated as

```
Calories from fat ÷ total calories
```

Input Validation: The program should make sure that the number of calories is greater than 0, the number of fat grams is 0 or more, and the number of calories from fat is not greater than the total number of calories.

4. 18. The Speed of Sound

The speed of sound varies depending on the medium through which it travels. In general, sound travels fastest in rigid media, such as steel, slower in liquid media, such as water, and slowest of all in gases, such as air. The following table shows the approximate speed of sound, measured in feet per second, in air, water, and steel.

Medium	Speed (feet per sec)
Air	1,100
Water	4,900
Steel	16,400

Write a program that displays a menu allowing the user to select air, water, or steel. After the user has made a selection, the number of feet a sound wave will travel in the selected medium should be entered. The program will then display the amount of time it will take. (Round the answer to four decimal places.)

Input Validation: Decide how the program should handle an illegal input for the menu choice or a negative value for the distance.

4. 19. The Speed of Sound in Gases

When traveling through a gas, the speed of sound depends primarily on the density of the medium. The less dense the medium, the faster the speed will be. The following table shows the approximate speed of sound at 0 degrees Celsius, measured in meters per second, when traveling through carbon dioxide, air, helium, and hydrogen.

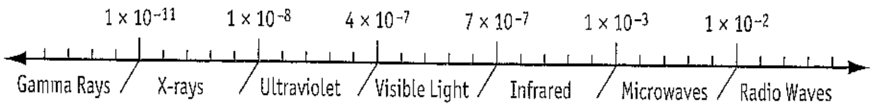
Medium	Speed (meters per sec)
Carbon dioxide	258.0
Air	331.5
Helium	972.0
Hydrogen	1270.0

Write a program that displays a menu allowing the user to select one of these four gases. After a valid selection has been made, the program should ask the user to enter the number of seconds (0 to 30) it took for the sound to travel in this medium from its source to the location at which it was detected. The program should then report how far away (in meters) the source of the sound was from the detection location.

Input Validation: If the use enters an invalid menu choice the program should display an error message instead of prompting for the number of seconds.

4. **20. Spectral Analysis**

If a scientist knows the wavelength of an electromagnetic wave, she can determine what type of radiation it is. Write a program that asks for the wavelength in meters of an electromagnetic wave and then displays what that wave is according to the following chart. (For example, a wave with a wavelength of 1E-10 meters would be an X-ray.)



4. **21. Freezing and Boiling Points**

The following table lists the freezing and boiling points of several substances. Write a program that asks the user to enter a temperature, and then shows all the substances that will freeze at that temperature and all that will boil at that temperature. For example, if the user enters –20, the program should report that water will freeze and oxygen will boil at that temperature.

Substance	Freezing Point (°F)	Boiling Point (°F)
Ethyl alcohol	–173	172
Mercury	–38	676
Oxygen	–362	–306
Water	32	212

4. **22. Mobile Service Provider**

A mobile phone service has three different subscription packages for its customers:

Package A:	For \$39.99 per month, 450 minutes are provided. Additional usage costs \$0.45 per minute.
Package B:	For \$59.99 per month, 900 minutes are provided. Additional usage costs \$0.40 per minute.
Package C:	For \$69.99 per month, unlimited minutes are provided.

Write a program that calculates a customer's monthly bill. It should input customer name, which package the customer has purchased, and how many minutes were used. It should then create a bill that includes the input information and the total amount due. It should also display much money Package A customers would save if they purchased package B or C, and how much money package B customers would save if they purchased package C. If there would be no savings, no message should be printed. Wherever possible, use named constants instead of numbers.

Soft Skills

Programmers need to be able to analyze what is wrong with a faulty algorithm and be able to explain the problem to others.

46. Write a clear problem description for a simple program and create a pseudocode solution for it. The pseudocode should incorporate the logic, including all the calculations, needed in the program, but should purposely contain a subtle logic error. Then pair up with another student in the class who has done the same thing and swap your work. Each of you should trace the logic to find the error in the pseudocode you are given, then clearly explain to your partner what the problem is, why the "code" will not work as written, and what should be done to correct it.

As an alternative, your instructor may wish to provide you with a problem description and an incorrect pseudocode solution. Again, the goal is not only for you to find the error, but also to clearly explain what the problem is, why the "code" will not work as written, and what should be done to correct it.

Programming Challenges – CHAPTER 5

5. 1. Characters for the ASCII Codes

Write a program that uses a loop to display the characters for each ASCII code 32 through 127. Display 16 characters on each line with one space between characters.

5. 2. Sum of Numbers

Write a program that asks the user for a positive integer value and that uses a loop to validate the input. The program should then use a second loop to compute the sum of all the integers from 1 up to the number entered. For example, if the user enters 50, the loop will find the sum of 1, 2, 3, 4, ... 50.

5. 3. Distance Traveled

The distance a vehicle travels can be calculated as follows:

```
distance = speed * time
```

For example, if a train travels 40 miles per hour for 3 hours, the distance traveled is 120 miles.

Write a program that asks the user for the speed of a vehicle (in miles per hour) and how many hours it has traveled. It should then use a loop to display the total distance traveled at the end of each hour of that time period. Here is an example of the output:

```
What is the speed of the vehicle in mph? 40
How many hours has it traveled? 3
Hour         Miles Traveled
-------------------------------
 1                40
 2                80
 3               120
```

5. **4. Celsius to Fahrenheit Table**

In one of the Chapter 3 Programming Challenges you were asked to write a program that converts a Celsius temperature to Fahrenheit. Modify that program so it uses a loop to display a table of the Celsius temperatures from 0 to 30 and their Fahrenheit equivalents.

$F = 9/5C + 32$

5, **5. Speed Conversion Chart**

Write a program that displays a table of speeds in kilometers per hour with their values converted to miles per hour. The table should display the speeds from 40 kilometers per hour through 120 kilometers per hour, in increments of 10 kilometers per hour. (In other words, it should display 40 kph, 50 kph, 60 kph and so forth, up through 120 kph.)

$MPH = KPH * 0.6214$

5, **6. Ocean Levels**

VideoNote
Solving the Ocean Levels Problem

Assuming the level of the Earth's oceans continues rising at about 3.1 millimeters per year, write a program that displays a table showing the total number of millimeters the oceans will have risen each year for the next 25 years.

5, **7. Circle Areas**

The formula to compute the area of a circle is

$\text{area} = \text{PI} * \text{radius}^2$

so if a circle's radius doubles (i.e., is multiplied by 2), the circle's area will be four times as large as before. Write a program that creates a table showing the radius and area for a circle whose radius begins with 1 and continues doubling until it is 8. Use 3.14 for PI.

5, **8. Pennies for Pay**

Write a program that calculates how much a person earns in a month if the salary is one penny the first day, two pennies the second day, four pennies the third day, and so on with the daily pay doubling each day the employee works. The program should ask the user for the number of days the employee worked during the month, validate that it is between 1 and 31, and then display a table showing how much the salary was for each day worked, as well as the total pay earned for the month. The output should be displayed in dollars with two decimal points, not in pennies.

5, **9. Weight Loss**

If moderately active persons cut their calorie intake by 500 calories a day, they can typically lose about 4 pounds a month. Write a program that has the users enter their starting weight and then creates and displays a table showing what their expected weight will be at the end of each month for the next 6 months if they stay on this diet.

5. **10. Calories Burned**

Running on a particular treadmill, you burn 3.9 calories per minute. Write a program that uses a loop to display the number of calories burned after 5, 10, 15, 20, 25, and 30 minutes.

5. **11. Membership Fees Increase**

A country club, which currently charges $3,000 per year for membership, has announced it will increase its membership fee by 3 percent each year for the next five years. Write a program that uses a loop to display the projected rates for the next five years.

5. **12. Random Number Guessing Game**

Write a program that generates a random number between 1 and 100 and asks the user to guess what the number is. If the user's guess is higher than the random number, the program should display "Too high. Try again." If the user's guess is lower than the random number, the program should display "Too low. Try again." The program should use a loop that repeats until the user correctly guesses the random number. Then the program should display "Congratulations. You figured out my number."

5. **13. Random Number Guessing Game Enhancement**

Enhance the program that you wrote for Programming Challenge 12 so it keeps a count of the number of guesses the user makes. When the user correctly guesses the random number, the program should display the number of guesses along with the message of congratulations.

5. **14. The Greatest and Least of These**

Write a program with a loop that lets the user enter a series of integers, followed by -99 to signal the end of the series. After all the numbers have been entered, the program should display the largest and smallest numbers entered.

5. **15. Student Line-Up**

A teacher has asked all her students to line up single file according to their first name. For example, in one class Amy will be at the front of the line and Yolanda will be at the end. Write a program that prompts the user to enter a number between 1 and 20 for the number of students in the class and then loops to read in that many names. Once all the names have been read in, it reports which student would be at the front of the line and which one would be at the end of the line. You may assume that no two students have the same name.

5. 16. Rate of Inflation

The annual rate of inflation is the rate at which money loses its value. For example, if the annual rate of inflation is 3.0 percent, then in one year it will cost $1,030 to buy the goods that could have been purchased for $1,000 today. Put another way, a year from now $1,000 will only buy 1/1.03 * $1,000, or $970.87, worth of goods. Two years from now, $1,000 will only buy only 1/1.03 of $970.87, or $942.59 worth of goods. Write a program that allows the user to enter an annual rate of inflation between 1 percent and 10 percent, and which then displays a table showing how much $1,000 today will be worth each year for the next 10 years.

5. 17. Population

Write a program that will predict the size of a population of organisms. The program should ask the user for the starting number of organisms, their average daily population increase (as a percentage of current population), and the number of days they will multiply. A loop should display the size of the population for each day.

> *Input Validation: The program should not accept a number less than 2 for the starting size of the population, a negative number for average daily population increase, or a number less than 1 for the number of days they will multiply.*

5. 18. Math Tutor Version 3

This program started in Chapter 3 and was modified in Chapter 4. Starting with the version described in Chapter 4, modify the program again so that it displays a menu allowing the user to select an addition, subtraction, or multiplication problem. The final selection on the menu should let the user quit the program. After the user has finished the math problem, the program should display the menu again. This process must repeat until the user chooses to quit the program. If the user selects an item not on the menu, the program should print an error message and then display the menu again.

5. 19. Hotel Suites Occupancy

Write a program that calculates the occupancy rate of the 120 suites (20 per floor) located on the top six floors of a 15-story luxury hotel. These are floors 10–12 and 14–16 because, like many hotels, there is no 13th floor. Solve the problem by using a *single* loop that iterates once for each floor between 10 and 16 and, on each iteration, asks the user to input the number of suites occupied on that floor. Use a nested loop to validate that the value entered is between 0 and 20. After all the iterations, the program should display how many suites the hotel has, how many of them are occupied, and what percentage of them are occupied.

5, 20. Rectangle Display

Write a program that asks the user for two positive integers between 2 and 10 to use for the length and width of a rectangle. If the numbers are different, the larger of the two numbers should be used for the length and the smaller for the width. The program should then display a rectangle of this size on the screen using the character 'X'. For example, if the user enters either 2 5 or 5 2, the program should display the following:

```
XXXXX
XXXXX
```

5, 21. Diamond Display

Write a program that uses nested loops to display the diamond pattern shown below.

```
   +
  +++
 +++++
+++++++
 +++++
  +++
   +
```

5, 22. Triangle Display

Write a program that uses nested loops to display the triangle pattern shown below.

```
+
+++
+++++
+++++++
+++++
+++
+
```

5, 23. Arrowhead Display

Write a program that uses nested loops to display the arrowhead pattern shown below.

```
      +
      +++
      +++++
+++++++++++++
      +++++
      +++
      +
```

5, 24. Sales Bar Chart

Write a program that asks the user to enter today's sales rounded to the nearest $100 for each of three stores. The program should then produce a bar graph displaying each store's sales. Create each bar in the graph by displaying a row of asterisks. Each asterisk should represent $100 of sales.

Here is an example of the program's output. User input is shown in bold.

```
Enter today's sales for store 1: 1000[Enter]
Enter today's sales for store 2: 1200[Enter]
Enter today's sales for store 3: 900[Enter]

    DAILY SALES
  (each * = $100)
Store 1: **********
Store 2: ************
Store 3: *********
```

5,c 25. Savings Account Balance

Write a program that calculates the balance of a savings account at the end of a three-month period. It should ask the user for the starting balance and the annual interest rate. A loop should then iterate once for every month in the period, performing the following steps:

A) Ask the user for the total amount deposited into the account during that month and add it to the balance. Do not accept negative numbers.

B) Ask the user for the total amount withdrawn from the account during that month and subtract it from the balance. Do not accept negative numbers or numbers greater than the balance after the deposits for the month have been added in.

C) Calculate the interest for that month. The monthly interest rate is the annual interest rate divided by 12. Multiply the monthly interest rate by the average of that month's starting and ending balance to get the interest amount for the month. This amount should be added to the balance.

After the last iteration, the program should display a nicely formatted report that includes the following information:

- Starting balance at the beginning of the three-month period
- Total deposits made during the three months
- Total withdrawals made during the three months
- Total interest posted to the account during the three months
- Final balance

5,f 26. Using Files—Total and Average Rainfall

Write a program that reads in from a file a starting month name, an ending month name, and then the monthly rainfall for each month during that period. As it does this, it should sum the rainfall amounts and then report the total rainfall and average rainfall for the period. For example, the output might look like this:

During the months of March–June, the total rainfall was 7.32 inches and the average monthly rainfall was 1.83 inches.

Data for the program can be found in the `Rainfall.txt` file located in the Chapter 5 programs folder on the book's companion website.

Hint: After reading in the month names, you will need to read in rain amounts until the EOF is reached and count how many pieces of rain data you read in.

5 27. Using Files—Population Bar Chart

Write a program that produces a bar chart showing the population growth of Prairieville, a small town in the Midwest, at 20-year intervals during the past 100 years. The program should read in the population figures (rounded to the nearest 1,000 people) for 1910, 1930, 1950, 1970, 1990, and 2010 from a file. For each year it should display the date and a bar consisting of one asterisk for each 1,000 people. The data can be found in the `People.txt` file located in the Chapter 5 programs folder on the book's companion website.

Here is an example of how the chart might begin:

```
PRAIRIEVILLE POPULATION GROWTH
(each * represents 1000 people)

1910  **
1930  ****
1950  *****
```

5 28. Using Files—Student Line Up

Modify the Student Line-Up program described in Programming Challenge 15 so that it gets the names from a data file. Names should be read in until there is no more data to read. Data to test your program can be found in the `LineUp.txt` file located in the Chapter 5 programs folder on the book's companion website.

5 29. Using Files—Savings Account Balance Modification

Modify the Savings Account Balance program described in Programming Challenge 25 so that it writes the report to a file. After the program runs, print the file to hand in to your instructor.

Programming Challenges - CHAPTER 6

6.1. Markup

VideoNote Solving the Markup Problem

Write a program that asks the user to enter an item's wholesale cost and its markup percentage. It should then display the item's retail price. For example:

- If an item's wholesale cost is \$5.00 and its markup percentage is 100 percent, then the item's retail price is \$10.00.
- If an item's wholesale cost is \$5.00 and its markup percentage is 50 percent, then the item's retail price is \$7.50.

The program should have a function named `calculateRetail` that receives the wholesale cost and the markup percentage as arguments and returns the retail price of the item.

6.2. Celsius Temperature Table

The formula for converting a temperature from Fahrenheit to Celsius is

$$C = \frac{5}{9}(F - 32)$$

where F is the Fahrenheit temperature and C is the Celsius temperature. Write a function named `celsius` that accepts a Fahrenheit temperature as an argument and returns the temperature converted to Celsius. Demonstrate the function by calling it in a loop that displays a table of the Fahrenheit temperatures 0 through 20 and their Celsius equivalents.

6.3. Falling Distance

The following formula can be used to determine the distance an object falls due to gravity in a specific time period:

$$d = \frac{1}{2} gt^2$$

The variables in the formula are as follows:

- d is the distance in meters,
- g is 9.8,
- and t is the time in seconds that the object has been falling.

Write a function named `fallingDistance` that accepts an object's falling time (in seconds) as an argument. The function should return the distance, in meters, that the object has fallen during that time interval. Write a program that demonstrates the function by calling it in a loop that passes the values 1 through 10 as arguments and displays the return value.

6.4. Kinetic Energy

In physics, an object that is in motion is said to have kinetic energy. The following formula can be used to determine a moving object's kinetic energy:

$$KE = \frac{1}{2} mv^2$$

The variables in the formula are as follows:

- KE is the kinetic energy in joules,
- m is the object's mass in kilograms,
- and v is the object's velocity in meters per second.

Write a function named `kineticEnergy` that accepts an object's mass (in kilograms) and velocity (in meters per second) as arguments. The function should return the amount of kinetic energy that the object has. Demonstrate the function by calling it in a program that asks the user to enter values for mass and velocity.

6, 5. Winning Division

Write a program that determines which of a company's four divisions (Northeast, Southeast, Northwest, and Southwest) had the greatest sales for a quarter. It should include the following two functions, which are called by `main`.

- `double getSales()` is passed the name of a division. It asks the user for a division's quarterly sales figure, validates that the input is not less than 0, then returns it. It should be called once for each division.
- `void findHighest()` is passed the four sales totals. It determines which is the largest and prints the name of the high grossing division, along with its sales figure.

6. 6. Shipping Charges

The Fast Freight Shipping Company charges the following rates:

Weight of Package (in kilograms)	Rate per 500 Miles Shipped
2 kg or less	$3.10
Over 2 kg but not more than 6 kg	$4.20
Over 6 kg but not more than 10 kg	$5.30
over 10 kg	$6.40

Write a program that asks for the weight of a package and the distance it is to be shipped. This information should be passed to a `calculateCharge` function that computes and returns the shipping charge to be displayed. The main function should loop to handle multiple packages until a weight of 0 is entered.

6,7. String Compare

You know that the `==` operator can be used to test if two `string` objects are equal. However, you will recall that they are not considered equal, even when they hold the exact same letters, if the cases of any letters are different. So, for example, if `name1 = "Jack"` and `name2 = "JACK"`, they are not considered the same. Write a program that asks the user to enter two names and stores them in `string` objects. It should then report whether or not, ignoring case, they are the same.

To help the program accomplish its task, it should use two functions in addition to `main`, `upperCaseIt()` and `sameString()`. Here are their function headers.

```
string upperCaseIt(string s)
Boolean sameString (string s1, string s2)
```

The `sameString` function, which receives the two strings to be compared, will need to call `upperCaseIt` for each of them before testing if they are the same. The `upperCaseIt` function should use a loop so that it can call the `toupper` function for every character in the string it receives before returning it to the `sameString` function.

6. **8. Lowest Score Drop**

- Write a program that calculates the average of a group of test scores, where the lowest score in the group is dropped. It should use the following functions:
- `void getScore()` should ask the user for a test score, store it in a reference parameter variable, and validate that it is not lower than 0 or higher than 100. This function should be called by `main` once for each of the five scores to be entered.
- `void calcAverage()` should calculate and display the average of the four highest scores. This function should be called just once by `main` and should be passed the five scores.
- `int findLowest()` should find and return the lowest of the five scores passed to it. It should be called by `calcAverage`, which uses the function to determine which one of the five scores to drop.

6. **9. Star Search**

A particular talent competition has five judges, each of whom awards a score between 0 and 10 to each performer. Fractional scores, such as 8.3, are allowed. A performer's final score is determined by dropping the highest and lowest score received, then averaging the three remaining scores. Write a program that uses these rules to calculate and display a contestant's score. It should include the following functions:

- `void getJudgeData()` should ask the user for a judge's score, store it in a reference parameter variable, and validate it. This function should be called by `main` once for each of the five judges.
- `double calcScore()` should calculate and return the average of the three scores that remain after dropping the highest and lowest scores the performer received. This function should be called just once by `main` and should be passed the five scores.

Two additional functions, described below, should be called by `calcScore`, which uses the returned information to determine which of the scores to drop.

- `int findLowest()` should find and return the lowest of the five scores passed to it.
- `int findHighest()` should find and return the highest of the five scores passed to it.

6. **10. `isPrime` Function**

A prime number is an integer greater than 1 that is evenly divisible by only 1 and itself. For example, the number 5 is prime because it can only be evenly divided by 1 and 5. The number 6, however, is not prime because it can be divided by 1, 2, 3, and 6.

Write a Boolean function named `isPrime`, which takes an integer as an argument and returns `true` if the argument is a prime number, and `false` otherwise. Demonstrate the function in a complete program.

TIP: Recall that the `%` operator divides one number by another and returns the remainder of the division. In an expression such as `num1 % num2`, the `%` operator will return 0 if `num1` is evenly divisible by `num2`.

6. **11. Present Value**

Suppose you want to deposit a certain amount of money into a savings account and then leave it alone to draw interest for the next 10 years. At the end of 10 years you would like to have $10,000 in the account. How much do you need to deposit today to make that happen? To find out you can use the following formula, which is known as the *present value formula:*

$$P = \frac{F}{(1 + r)^n}$$

The terms in the formula are as follows:

- P is the **present value**, or the amount that you need to deposit today.
- F is the **future value** that you want in the account (in this case, $10,000).
- r is the **annual interest rate** (expressed in decimal form, such as .042).
- n is the **number of years** that you plan to let the money sit in the account.

Write a program with a function named `presentValue` that performs this calculation. The function should accept the future value, annual interest rate, and number of years as arguments. It should return the present value, which is the amount that you need to deposit today. Demonstrate the function in a program that lets the user experiment with different values for the formula's terms.

6. **12. Future Value**

Suppose you have a certain amount of money in a savings account that earns compound monthly interest, and you want to calculate the amount that you will have after a specific number of months. The formula, which is known as the *future value formula*, is:

$$F = P \times (1 + i)^t$$

The variables in the formula are as follows:

- F is the **future value** of the account after the specified time period.
- P is the **present value** of the account.
- i is the **monthly interest rate**.
- t is the **number of months**.

Write a program that prompts the user to enter the account's present value, monthly interest rate, and number of months that the money will be left in the account. The program should pass these values to a function named `futureValue` that computes and returns the future value of the account after the specified number of months. The program should display the account's future value.

6. 13. Stock Profit

The profit from the sale of a stock can be calculated as follows:

$$\text{Profit} = ((NS \times SP) - SC) - ((NS \times PP) + PC)$$

- where *NS* is the number of shares,
- *SP* is the sale price per share,
- *SC* is the sale commission paid,
- *PP* is the purchase price per share,
- and *PC* is the purchase commission paid.

If the calculation yields a positive value, then the sale of the stock resulted in a profit. If the calculation yields a negative number, then the sale resulted in a loss.

Write a function that accepts as arguments the number of shares, the purchase price per share, the purchase commission paid, the sale price per share, and the sale commission paid. The function should return the profit (or loss) from the sale of stock.

Demonstrate the function in a program that asks the user to enter the necessary data and displays the amount of the profit or loss.

6. 14. Multiple Stock Sales

Use the function that you wrote for Programming Challenge 13 (Stock Profit) in a program that calculates the total profit or loss from the sale of multiple stocks. The program should ask the user for the number of stock sales, and the necessary data for each stock sale. It should accumulate the profit or loss for each stock sale and then display the total.

6. 15. Order Status

The Middletown Wholesale Copper Wire Company sells spools of copper wiring for \$100 each and ships them for \$10 apiece. Write a program that displays the status of an order. It should use two functions. The first function asks for the following data and stores the input values in reference parameters.

- The number of spools ordered.
- The number of spools in stock.
- Any special shipping and handling charges (above the regular \$10 rate).

The second function receives as arguments any values needed to compute and display the following information:

- The number of ordered spools ready to ship from current stock.
- The number of ordered spools on backorder (if the number ordered is greater than what is in stock).
- Total selling price of the portion ready to ship (the number of spools ready to ship times \$100).
- Total shipping and handling charges on the portion ready to ship.
- Total of the order ready to ship.

The shipping and handling parameter in the second function should have the default argument 10.00.

6. 16. Overloaded Hospital

Write a program that computes and displays the charges for a patient's hospital stay. First, the program should ask if the patient was admitted as an inpatient or an outpatient. If the patient was an inpatient, the following data should be entered:

- The number of days spent in the hospital
- The daily rate
- Charges for hospital services (lab tests, etc.)
- Hospital medication charges

If the patient was an outpatient, the following data should be entered:

- Charges for hospital services (lab tests, etc.)
- Hospital medication charges

Use a single, separate function to validate that no input is less than zero. If it is, it should be reentered before being returned.

Once the required data has been input and validated, the program should use two overloaded functions to calculate the total charges. One of the functions should accept arguments for the inpatient data, while the other function accepts arguments for outpatient data. Both functions should return the total charges.

6. 17. Population

In a population, the birth rate is the percentage increase of the population due to births, and the death rate is the percentage decrease of the population due to deaths. Write a program that asks for the following:

- The starting size of a population (minimum 2)
- The annual birth rate
- The annual death rate
- The number of years to display (minimum 1)

The program should then display the starting population and the projected population at the end of each year. It should use a function that calculates and returns the projected new size of the population after a year. The formula is

$$N = P(1 + B)(1 - D)$$

where

- N is the new population size,
- P is the previous population size,
- B is the birth rate,
- and D is the death rate.

Annual birth rate and death rate are the typical number of births and deaths in a year per 1,000 people, expressed as a decimal. So, for example, if there are normally about 32 births and 26 deaths per 1,000 people in a given population, the birth rate would be .032 and the death rate would be .026.

6. 18. Transient Population

Modify Programming Challenge 17 to also consider the effect on population caused by people moving into or out of a geographic area. Given as input a starting population size, the annual birth rate, the annual death rate, the number of individuals that typically move into the area each year, and the number of individuals that typically leave the area each year, the program should project what the population will be `numYears` from now. You can either prompt the user to input a value for `numYears`, or you can set it within the program.

6. 19. Using Files—Hospital Report

Modify Programming Challenge 16, Overloaded Hospital, to write the report it creates to a file. Print the contents of the file to hand in to your instructor.

Group Project

6. 20. Using Files—Travel Expenses

This program should be designed and written by a team of students. Here are some suggestions:

- One student should design function `main`, which will call the other functions in the program. The rest of the functions should be designed by other team members.
- Analyze the program requirements so each student is given about the same workload.
- Decide on the function names, parameters, and return types in advance.
- Use stubs and drivers to test and debug the program.
- The program can be implemented either as a multifile program, or all the functions can be cut and pasted into the main file.

Here is the assignment. Write a program that calculates and displays the total travel expenses of a businessperson on a trip. The program should have functions that ask for and return the following:

- The total number of days spent on the trip
- The time of departure on the first day of the trip and the time of arrival back home on the last day of the trip
- The amount of any round-trip airfare
- The amount of any car rentals
- Miles driven, if a private vehicle was used. Vehicle allowance is $0.58 per mile.
- Parking fees. (The company allows up to $12 per day. Anything in excess of this must be paid by the employee.)
- Taxi fees. (The company allows up to $40 per day for each day a taxi was used. Anything in excess of this must be paid by the employee.)
- Conference or seminar registration fees
- Hotel expenses. (The company allows up to $90 per night for lodging. Anything in excess of this amount must be paid by the employee.)

- The cost of each meal eaten. On the first day of the trip, breakfast is allowed as an expense if the time of departure is before 7 a.m. Lunch is allowed if the time of departure is before noon. Dinner is allowed if the time of departure is before 6 p.m. On the last day of the trip, breakfast is allowed if the time of arrival is after 8 a.m. Lunch is allowed if the time of arrival is after 1 p.m. Dinner is allowed if the time of arrival is after 7 p.m. The program should only ask for the costs of allowable meals. (The company allows up to $18 for breakfast, $12 for lunch, and $20 for dinner. Anything in excess of this must be paid by the employee.)

The program should perform the necessary calculations to determine the total amount spent by the business traveler in each category (mileage charges, parking, hotel, meals, etc.) as well as the maximum amount allowed in each category. It should then create a nicely formatted expense report that includes the amount spent and the amount allowed in each category, as well as the total amount spent and total amount allowed for the entire trip. This report should be written to a file.

Input Validation: The program should not accept negative numbers for any dollar amount or for miles driven in a private vehicle. It should also ensure that the number of days is at least 1 and that the time of departure and the time of arrival are valid.

55. Look at the following description of a problem domain:

    The bank offers the following types of accounts to its customers: savings accounts, checking accounts, and money market accounts. Customers are allowed to deposit money into an account (thereby increasing its balance), withdraw money from an account (thereby decreasing its balance), and earn interest on the account. Each account has an interest rate.

    Assume that you are writing an application that will calculate the amount of interest earned for a bank account.

    A) Identify the potential classes in this problem domain.
    B) Refine the list to include only the necessary class or classes for this problem.
    C) Identify the responsibilities of the class or classes.

Soft Skills

Working in a team can often help individuals better understand new ideas related to programming. Others can explain things that you do not understand. Also, you will find that by explaining something to someone else, you actually understand it better.

56. Write down one question you have about the object-oriented programming material from Chapter 7. For example, you could mention something you want explained about how classes are designed and created, about how objects are related to classes, or about how overloaded constructors work. Then form a group with three to four other students. Each person in the group should participate in answering the questions posed by the other members of the group.

Programming Challenges ~ CHAPTER 7

7, 1. Date

Design a class called `Date` that has integer data members to store month, day, and year. The class should have a three-parameter default constructor that allows the date to be set at the time a new `Date` object is created. If the user creates a `Date` object without passing any arguments, or if any of the values passed are invalid, the default values of 1, 1, 2001 (i.e., January 1, 2001) should be used. The class should have member functions to print the date in the following formats:

```
3/15/16
March 15, 2016
15 March 2016
```

Demonstrate the class by writing a program that uses it. Be sure your program only accepts reasonable values for month and day. The month should be between 1 and 12. The day should be between 1 and the number of days in the selected month.

7, **2. Report Heading**

Design a class called `Heading` that has data members to hold the company name and the report name. A two-parameter default constructor should allow these to be specified at the time a new `Heading` object is created. If the user creates a `Heading` object without passing any arguments, "`ABC Industries`" should be used as a default value for the company name and "`Report`" should be used as a default for the report name. The class should have member functions to print a heading in either one-line format, as shown here:

```
Pet Pals Payroll Report
```

or in four-line "boxed" format, as shown here:

```
********************************************************
                    Pet Pals
                 Payroll Report
********************************************************
```

Try to figure out a way to center the headings on the screen, based on their lengths. Demonstrate the class by writing a simple program that uses it.

7, **3. Widget Factory**

Design a class for a widget manufacturing plant. Assuming that 10 widgets may be produced each hour, the class object will calculate how many days it will take to produce any number of widgets. (The plant operates two 8-hour shifts per day.) Write a program that asks the user for the number of widgets that have been ordered and then displays the number of days it will take to produce them. Think about what values your program should accept for the number of widgets ordered.

VideoNote
Solving the Car Class Problem

7, **4. Car Class**

Write a class named `Car` that has the following member variables:

- **`year`**. An `int` that holds the car's model year.
- **`make`**. A `string` object that holds the make of the car.
- **`speed`**. An `int` that holds the car's current speed.

In addition, the class should have the following member functions.

- **`Constructor`**. The constructor should accept the car's year and make as arguments and assign these values to the object's `year` and `make` member variables. The constructor should initialize the `speed` member variable to 0.
- **`Accessors`**. Appropriate accessor functions should be created to allow values to be retrieved from an object's `year`, `make`, and `speed` member variables.
- **`accelerate`**. The `accelerate` function should add 5 to the `speed` member variable each time it is called.
- **`brake`**. The `brake` function should subtract 5 from the `speed` member variable each time it is called.

Demonstrate the class in a program that creates a `Car` object and then calls the `accelerate` function five times. After each call to the `accelerate` function, get the current speed of the car and display it. Then, call the `brake` function five times. After each call to the `brake` function, get the current speed of the car and display it.

7, 5. Population

In a population, the birth rate and death rate are calculated as follows:

Birth Rate = Number of Births ÷ Population
Death Rate = Number of Deaths ÷ Population

For example, in a population of 100,000 that has 8,000 births and 6,000 deaths per year,

Birth Rate = 8,000 ÷ 100,000 = 0.08
Death Rate = 6,000 ÷ 100,000 = 0.06

Design a `Population` class that stores a current population, annual number of births, and annual number of deaths for some geographic area. The class should allow these three values to be set in either of two ways: by passing arguments to a three-parameter constructor when a new `Population` object is created or by calling the `setPopulation`, `setBirths`, and `setDeaths` class member functions. In either case, if a population figure less than 2 is passed to the class, use a default value of 2. If a birth or death figure less than 0 is passed in, use a default value of 0. The class should also have `getBirthRate` and `getDeathRate` functions that compute and return the birth and death rates. Write a short program that uses the `Population` class and illustrates its capabilities.

7, 6. Gratuity Calculator

Design a `Tips` class that calculates the gratuity on a restaurant meal. Its only class member variable, `taxRate`, should be set by a one-parameter constructor to whatever rate is passed to it when a `Tips` object is created. If no argument is passed, a default tax rate of .065 should be used. The class should have just one public function, `computeTip`. This function needs to accept two arguments, the total bill amount and the tip rate. It should use this information to compute what the cost of the meal was before the tax was added. It should then apply the tip rate to just the meal cost portion of the bill to compute and return the tip amount. Demonstrate the class by creating a program that creates a single `Tips` object, then loops multiple times to allow the program user to retrieve the correct tip amount using various bill totals and desired tip rates.

7, 7. Inventory Class

Design an `Inventory` class that can hold information for an item in a retail store's inventory. The class should have the following private member variables.

Variable Name	Description
`itemNumber`	An `int` that holds the item's number.
`quantity`	An `int` that holds the quantity of the item on hand.
`cost`	A `double` that holds the wholesale per-unit cost of the item

The class should have the following public member functions.

Member Function	Description
default constructor	Sets all the member variables to 0.
constructor #2	Accepts an item's number, quantity, and cost as arguments. Calls other class functions to copy these values into the appropriate member variables. Then calls the `setTotalCost` function.

Member Function	Description
`setItemNumber`	Accepts an `int` argument and copies it into the `itemNumber` member variable.
`setQuantity`	Accepts an `int` argument and copies it into the `quantity` member variable.
`setCost`	Accepts a `double` argument and copies it into the `cost` member variable.
`getItemNumber`	Returns the value in `itemNumber`.
`getQuantity`	Returns the value in `quantity`.
`getCost`	Returns the value in `cost`.
`getTotalCost`	Computes and returns the `totalCost`.

Demonstrate the class by writing a simple program that uses it. This program should validate the user inputs to ensure that negative values are not accepted for item number, quantity, or cost.

7 **8. Movie Data**

Write a program that uses a structure named `MovieData` to store the following information about a movie:

Title
Director
Year Released
Running time (in minutes)

Include a constructor that allows all four of these member data values to be specified at the time a `MovieData` variable is created. The program should create two `MovieData` variables and pass each one in turn to a function that displays the information about the movie in a clearly formatted manner. Pass the `MovieData` variables to the display function by value.

7 **9. Movie Profit**

Modify the Movie Data program written for Programming Challenge 8 to include two more members that hold the movie's production costs and first-year revenues. The constructor should be modified so that all six member values can be specified when a `MovieData` variable is created. Modify the function that displays the movie data to display the title, director, release year, running time, and first year's profit or loss. Also, improve the program by having the `MovieData` variables passed to the display function as constant references.

7 **10. Corporate Sales Data**

Write a program that uses a structure named `CorpData` to store the following information on a company division:

Division name (such as East, West, North, or South)
First quarter sales
Second quarter sales
Third quarter sales
Fourth quarter sales

Include a constructor that allows the division name and four quarterly sales amounts to be specified at the time a `CorpData` variable is created.

The program should create four `CorpData` variables, each representing one of the following corporate divisions: East, West, North, and South. These variables should be passed one at a time, as constant references, to a function that computes the division's annual sales total and quarterly average, and displays these along with the division name.

7, 11. Monthly Budget Screen Form

A student has established the following monthly budget:

Housing	500.00
Utilities	150.00
Household expenses	65.00
Transportation	50.00
Food	250.00
Medical	30.00
Insurance	100.00
Entertainment	150.00
Clothing	75.00
Miscellaneous	50.00

Write a modular program that declares a `MonthlyBudget` structure with member variables to hold each of these expense categories. The program should create two `MonthlyBudget` structure variables. The first will hold the budget figures given above. The second will hold the user-enter amounts actually spent during the past month. Using Program 7-19 as a model, the program should create a screen form that displays each category name and its budgeted amount, then positions the cursor next to it for the user to enter the amount actually spent in that category. Once the user data has all been entered, the program should compute and display the amount over or under budget the student's expenditures were in each category, as well as the amount over or under budget for the entire month.

7, 12. Ups and Downs

Write a program that displays the word UP on the bottom line of the screen a couple of inches to the left of center and displays the word DOWN on the top line of the screen a couple of inches to the right of center. Moving about once a second, move the word UP up a line and the word DOWN down a line until UP disappears at the top of the screen and DOWN disappears at the bottom of the screen.

7, 13. Wrapping Ups and Downs

Modify the program you wrote for Programming Challenge 12, so that after disappearing off of the screen, the word UP reappears at the bottom of the screen and the word DOWN reappears at the top of the screen. Have these words each traverse the screen three times before the program terminates.

7, 14. Left and Right

Modify the program you wrote for Programming Challenge 12 to display the words LEFT (starting at the right-hand side of the screen a row or two down from the middle) and RIGHT (starting at the left-hand side of the screen a row or two up from the middle). Moving about six moves per second, move LEFT to the left and RIGHT to the right until both words disappear off the screen.

7, 15. Moving Inchworm

Write a program that displays an inchworm on the left-hand side of the screen, facing right. Then slowly move him across the screen, until he disappears off the right-hand side. You may wish to do this in a loop so that after disappearing to the right, the worm appears again on the left. The diagram below shows how he may look at various points on the screen.

```
        \/              \/              \/              \/              \/
        00          0   00          000 00          0   00              00
~000000000      ~0000 0000      ~000   000      ~0000 0000      ~000000000
```

7, 16. Coin Toss Simulator

Write a class named `Coin`. The `Coin` class should have the following member variable:

- A `string` named `sideUp`. The `sideUp` member variable will hold either "heads" or "tails" indicating the side of the coin that is facing up.

The `Coin` class should have the following member functions:

- A default constructor that randomly determines the side of the coin that is facing up ("heads" or "tails") and initializes the `sideUp` member variable accordingly.
- A `void` member function named `toss` that simulates the tossing of the coin. When the `toss` member function is called, it randomly determines the side of the coin that is facing up ("heads" or "tails") and sets the `sideUp` member variable accordingly.
- A member function named `getSideUp` that returns the value of the `sideUp` member variable.

Write a program that demonstrates the `Coin` class. The program should create an instance of the class and display the side that is initially facing up. Then, use a loop to toss the coin 20 times. Each time the coin is tossed, display the side that is facing up. The program should keep count of the number of times heads is facing up and the number of times tails is facing up, and display those values after the loop finishes.

7, 17. Tossing Coins for a Dollar

Create a game program using the `Coin` class from Programming Challenge 16. The program should have three instances of the `Coin` class: one representing a quarter, one representing a dime, and one representing a nickel.

When the game begins, your starting balance is $0. During each round of the game, the program will toss each of the simulated coins. When a tossed coin lands heads-up, the value of the coin is added to your balance. For example, if the quarter lands heads-up, 25 cents is added to your balance. Nothing is added to your balance for coins that land tails-up. The game is over when your balance reaches one dollar or more. If your balance is *exactly* one dollar, you win the game. If your balance exceeds one dollar, you lose.

7, 18. Fishing Game Simulation

Write a program that simulates a fishing game. In this game, a six-sided die is rolled to determine what the user has caught. Each possible item is worth a certain number of fishing points. The points will remain hidden until the user is finished fishing, and then a message is displayed congratulating the user depending on the number of fishing points gained.

Here are some suggestions for the game's design:

- Each round of the game is performed as an iteration of a loop that repeats as long as the player wants to fish for more items.
- At the end of each round, the program will ask the user whether or not he or she wants to continue fishing.
- The program simulates the rolling of a six-sided die
- Each item that can be caught is represented by a number generated from the die—for example, 1 for "a huge fish", 2 for "an old shoe", 3 for "a little fish", and so on.
- Each item the user catches is worth a different number of points.
- You, the program designer, get to decide what fish or object each number will represent and how many points is associated with each "catch".
- The loop keeps a running total of the user's fishing points.
- When the loop is exited, the total number of fishing points is displayed, along with a message that varies depending on the number of points earned.

Group Project

7, 19. Patient Fees

This program should be designed and written by a team of students. Here are some suggestions:

- One or more students may work on a single class.
- The requirements of the program should be analyzed so each student is given about the same workload.
- The names, parameters, and return types of each function and class member function should be decided in advance.
- The program will be best implemented as a multifile program.

Write a program that computes a patient's bill for a hospital stay. The different components of the program are

- The `PatientAccount` class will keep a total of the patient's charges. It will also keep track of the number of days spent in the hospital. The group must decide on the hospital's daily rate.
- The `Surgery` class will have stored within it the charges for at least five types of surgery. It can update the `charges` variable of the `PatientAccount` class.
- The `Pharmacy` class will have stored within it the price of at least five types of medication. It can update the `charges` variable of the `PatientAccount` class.
- The `main` program.

The student who designs the main program will design a menu that allows the user to enter a type of surgery, enter one or more types of medication, and check the patient out of the hospital. When the patient checks out, the total charges should be displayed.

Soft Skills

Diagrams are an important means of clarifying many programming concepts. You have seen them used throughout this book to illustrate such things as how the flow of control works for various programming constructs, how a program is broken into modules and those modules related, how data is stored in memory, and how data is organized. Once you have covered Chapter 7, your teacher may wish to assign Question 40, which uses nested structures to organize a program's data.

40. Here is a set of declarations that define how the data for a set of poker hands is organized. Create a neat diagram that illustrates this organization. Figure 7-8 in Chapter 7 might give you an idea of how to begin.

```
struct CardStruct
{ int  face;
  char suit;      // 's', 'h', 'd', or 'c'
};

struct PlayerStruct
{ int playerNum;
  CardStruct card[5];
}

PlayerStruct player[4];
```

Programming Challenges – CHAPTER 8

Programming Challenges 1–10 allow you to practice working with arrays without using classes or structures. Most of the problems beginning with Programming Challenge 11 use arrays with classes or structures.

8.1. Perfect Scores

Write a modular program that accepts up to 20 integer test scores in the range of 0 to 100 from the user and stores them in an array. Then main should report how many perfect scores were entered (i.e., scores of 100), using a value-returning countPerfect function to help it.

8.2. Larger Than *n*

Create a program with a function that accepts three arguments: an integer array, an integer size that indicates how many elements are in the array, and an integer n. The function should display all of the numbers in the array that are greater than the number n.

8.3. Roman Numeral Converter

Write a program that displays the Roman numeral equivalent of any decimal number between 1 and 20 that the user enters. The Roman numerals should be stored in an array of strings, and the decimal number that the user enters should be used to locate the array element holding the Roman numeral equivalent. The program should have a loop that allows the user to continue entering numbers until an end sentinel of 0 is entered.

8.4. Chips and Salsa

VideoNote
Solving the Chips and Salsa Problem

Write a program that lets a maker of chips and salsa keep track of their sales for five different types of salsa they produce: mild, medium, sweet, hot, and zesty. It should use two parallel five-element arrays: an array of strings that holds the five salsa names and an array of integers that holds the number of jars sold during the past month for each salsa type. The salsa names should be stored using an initialization list at the time the name array is created. The program should prompt the user to enter the number of jars sold for each type. Once this sales data has been entered, the program should produce a report that displays sales for each salsa type, total sales, and the names of the highest selling and lowest selling products.

4, 5. Monkey Business

A local zoo wants to keep track of how many pounds of food each of its three monkeys eats each day during a typical week. Write a program that stores this information in a two-dimensional 3 × 7 array, where each row represents a different monkey and each column represents a different day of the week. The program should first have the user input the data for each monkey. Then it should create a report that includes the following information:

- Average amount of food eaten per day by the whole family of monkeys
- The least amount of food eaten during the week by any one monkey
- The greatest amount of food eaten during the week by any one monkey

4, 6. Rain or Shine

An amateur meteorologist wants to keep track of weather conditions during the past year's three-month summer season and has designated each day as either rainy ('R'), cloudy ('C'), or sunny ('S'). Write a modular program that stores this information in a 3 × 30 array of characters, where the row indicates the month (0 = June, 1 = July, 2 = August) and the column indicates the day of the month. Note that data is not being collected for the 31st of any month. The program should begin by calling a function to read the weather data in from a file. Then it should create a report that displays for each month and for the whole three-month period, how many days were rainy, how many were cloudy, and how many were sunny. To help it do this, it should use a value-returning function that is passed the array, the number of the month to examine, and the character to look for ('R', 'C', or 'S'). This function should return the number of days the indicated month had the requested weather. Data for the program can be found in the `RainOrShine.dat` file located in the Chapter 8 programs folder on this book's companion website.

4,7. Lottery

Write a program that simulates a lottery. The program should have an array of five integers named `winningDigits`, with a randomly generated number in the range of 0 through 9 for each element in the array. The program should ask the user to enter five digits and should store them in a second integer array named `player`. The program must compare the corresponding elements in the two arrays and count how many digits match. For example, the following shows the `winningDigits` array and the `Player` array with sample numbers stored in each. There are two matching digits, elements 2 and 4.

`WinningDigits`	7	4	9	1	3
`player`	4	2	9	7	3

Once the user has entered a set of numbers, the program should display the winning digits and the player's digits and tell how many digits matched.

9. **8. Rainfall Statistics**

Write a modular program that analyzes a year's worth of rainfall data. In addition to `main`, the program should have a `getData` function that accepts the total rainfall for each of 12 months from the user and stores it in a `double` array. It should also have four value-returning functions that compute and return to `main` the `totalRainfall`, `averageRainfall`, `driestMonth`, and `wettestMonth`. These last two functions return the *number* of the month with the lowest and highest rainfall amounts, not the amount of rain that fell those months. Notice that this month number can be used to obtain the amount of rain that fell those months. This information should be used either by `main` or by a `displayReport` function called by `main` to print a summary rainfall report similar to the following:

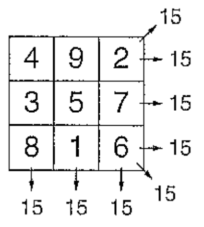
```
        2015 Rain Report for Neversnows County

    Total rainfall: 23.19 inches
    Average monthly rainfall: 1.93 inches
    The least rain fell in January with 0.24 inches.
    The most rain fell in April with 4.29 inches.
```

9. **9. Lo Shu Magic Square**

The Lo Shu Magic Square is a grid with three rows and three columns that has the following properties:

- The grid contains the numbers 1 through 9 exactly.
- The sum of each row, each column, and each diagonal all add up to the same number. This is shown in Figure 8-18.

Figure 8-18



Write a program that simulates a magic square using a two-dimensional 3 × 3 array. It should have a Boolean function `isMagicSquare` that accepts the array as an argument and returns `true` if it determines it is a Lo Shu Magic Square and `false` if it is not. Test the program with one array, such as the one shown in Figure 8-18, that is a magic square and one that is not.

9. **10. Baseball Champions**

This challenge uses two files located in the Chapter 8 programs folder on the book's companion website.

- `Teams.txt`—This file contains an alphabetical list of a number of Major League baseball teams that have won the World Series at least once.
- `WorldSeriesWinners.txt`—This file contains a chronological list of World Series' winning teams from 1950 through 2014. The first line in the file is the name of the team that won in 1950, and the last line is the name of the team that won in 2014. (Note that the World Series was not played in 1994.)

Write a program that reads the contents of each of these files into an array or vector. It should then display the contents of the Teams.txt file on the screen and prompt the user to enter the name of one of the teams. When the user enters a team name, the program should display the number of times that team has won the World Series in the time period from 1950 through 2014.

8, 11. Chips and Salsa Version 2

Revise Programming Challenge 4 to use an array of `Product` objects instead of two parallel arrays. The `Product` class will need member variables to hold a product name and a quantity.

8, 12. `Stats` Class and Rainfall Statistics

Create a `Stats` class whose member data includes an array capable of storing 30 `double` data values, and whose member functions include `total`, `average`, `lowest`, and `highest` functions for returning information about the data to the client program. These are general versions of the same functions you created for Programming Challenge 8, but now they belong to the `Stats` class, not the application program. In addition to these functions, the `Stats` class should have a Boolean `storeValue` function that accepts a `double` value from the client program and stores it in the array. It is the job of this function to keep track of how many values are currently in the array, so it will know where to put the next value it receives and will know how many values there are to process when it is carrying out its other functions. It is also the job of this function to make sure that no more than 30 values are accepted. If the `storeValue` function is able to successfully store the value sent to it, it should return `true` to the client program. However, if the client program tries to store a thirty-first value, the function should *not* store the value and should return `false` to the client program.

The client program should create and use a `Stats` object to carry out the same rainfall analysis requested by Programming Challenge 8. Notice that the `Stats` object does no I/O. All input and output is done by the client program.

4, 13. `Stats` Class and Track Statistics

Write a client program that uses the `Stats` class you created for Programming Challenge 12 to store and analyze "best" 100-yard dash times for each of the 15 runners on a track team. All I/O should be done by the client program. In addition to `main`, it should have two other functions: a `getData` function to accept input from the user and send it to the `Stats` object and a `createReport` function that creates and displays a report similar to the one shown here,

```
              Tulsa Tigers Track Team

Average 100 yard-dash time: 11.16 seconds
Slowest runner: Jack        13.09 seconds
Fastest runner: Will        10.82 seconds
```

4b, 14. Character Converter Class

Create a `CharConverter` class that performs various operations on strings. It should have the following two public member functions to start with. Your instructor may ask you to add more functions to the class.

- The `uppercase` member function accepts a string and returns a copy of it with all lowercase letters converted to uppercase. If a character is already uppercase or is not a letter, it should be left alone.
- The `properWords` member function accepts a string of words separated by spaces and returns a copy of it with the first letter of each word converted to uppercase.

Write a simple program that uses the class. It should prompt the user to input a string. Then it should call the `properWords` function and display the resulting string. Finally, it should call the `uppercase` function and display this resulting string. The program should loop to allow additional strings to be converted and displayed until the user chooses to quit.

4b/ 15. Driver's License Exam

The State Department of Motor Vehicles (DMV) has asked you to write a program that grades the written portion of the driver's license exam, which has 20 multiple-choice questions. Here are the correct answers:

1.B	5.C	9.C	13.D	17.C
2.D	6.A	10.D	14.A	18.B
3.A	7.B	11.B	15.D	19.D
4.A	8.A	12.C	16.C	20.A

To do this, you should create a `TestGrader` class. The class will have an `answers` array of 20 characters, which holds the correct test answers. It will have two public member functions that enable user programs to interact with the class: `setKey` and `grade`. The `setKey` function receives a 20-character string holding the correct answers and copies this information into its `answers` array. The `grade` function receives a 20-character array holding the test taker's answers and compares each of their answers to the correct one. An applicant must correctly answer 15 or more of the 20 questions to pass the exam. After "grading" the exam, the grade function should create and return to the user a string that includes the following information:

- A message indicating whether the applicant passed or failed the exam
- The number of right answers and the number of wrong answers
- A list of the question numbers for all incorrectly answered questions

The client program that creates and uses a `TestGrader` object should first make a single call to `setKey`, passing it a string containing the 20 correct answers. Once this is done, it should allow a test taker's 20 answers to be entered, making sure only answers of A–D are accepted, and store them in a 20-character array. Then it should call the `grade` function to grade the exam and should display the string the function returns. The program should loop to allow additional tests to be entered and graded until the user indicates a desire to quit.

5, **16. Array of Payroll Objects**

Design a PayRoll class that has data members for an employee's hourly pay rate and number of hours worked. Write a program with an array of seven PayRoll objects. The program should read the number of hours each employee worked and their hourly pay rate from a file and call class functions to store this information in the appropriate objects. It should then call a class function, once for each object, to return the employee's gross pay, so this information can be displayed. Sample data to test this program can be found in the payroll.dat file located in the Chapter 8 programs folder on this book's companion website.

6, **17. Drink Machine Simulator**

Create a class that simulates and manages a soft drink machine. Information on each drink type should be stored in a structure that has data members to hold the drink name, the drink price, and the number of drinks of that type currently in the machine.

The class should have an array of five of these structures, initialized with the following data.

Drink Name	Cost	Number in Machine
Cola	1.00	20
Root beer	1.00	20
Orange soda	1.00	20
Grape soda	1.00	20
Bottled water	1.50	20

The class should have two public member functions, displayChoices (which displays a menu of drink names and prices) and buyDrink (which handles a sale). The class should also have at least two private member functions, inputMoney, which is called by buyDrink to accept, validate, and return (to buyDrink) the amount of money input, and dailyReport, which is called by the destructor to report how many of each drink type remain in the machine at the end of the day and how much money was collected. You may want to use additional functions to make the program more modular.

The client program that uses the class should have a main processing loop that calls the displayChoices class member function and allows the patron to either pick a drink or quit the program. If the patron selects a drink, the buyDrink class member function is called to handle the actual sale. This function should be passed the patron's drink choice. Here is what the buyDrink function should do:

- Call the inputMoney function, passing it the patron's drink choice.
- If the patron no longer wishes to make the purchase, return all input money.
- If the machine is out of the requested soda, display an appropriate "sold out" message and return all input money.
- If the machine has the soda and enough money was entered, complete the sale by updating the quantity on hand and money collected information, calculating any change due to be returned to the patron, and delivering the soda. This last action can be simulated by printing an appropriate "here is your beverage" message.

18. Bin Manager Class

Design and write an object-oriented program for managing inventory bins in a warehouse. To do this you will use two classes: `InvBin` and `BinManager`. The `InvBin` class holds information about a single bin. The `BinManager` class will own and manage an array of `InvBin` objects. Here is a skeleton of what the `InvBin` and `BinManager` class declarations should look like:

```
class InvBin
{
   private:
      string description;                 // Item name
      int qty;                            // Quantity of items
                                          // in this bin


   public:
      InvBin (string d = "empty", int q = 0) // 2-parameter constructor
      {   description = d;  qty = q; }    // with default values

      // It will also have the following public member functions. They
      // will be used by the BinManager class, not the client program.
      void setDescription(string d)
      string getDescription()
      void setQty(int q)
      int getQty( )
};

class BinManager
{
   private:
      InvBin bin[30];                      // Array of InvBin objects
      int numBins;                         // Number of bins
                                           // currently in use


   public:
      BinManager()                         // Default constructor
      {   numBins = 0; }

      BinManager(int size, string d[], int q[]) // 3-parameter constructor
      {   // Receives number of bins in use and parallel arrays of item names
          // and quantities. Uses this info. to store values in the elements
          // of the bin array. Remember, these elements are InvBin objects.
      }

      // The class will also have the following public member functions:
      string getDescription(int index)     // Returns name of one item
      int getQuantity(int index)           // Returns qty of one item
      bool addParts(int binIndex, int q)   // These return true if the
      bool removeParts(int binIndex, int q) // action was done and false
                                           // if it could not be done—
                                           // see validation information
};
```

Client Program

Once you have created these two classes, write a menu-driven client program that uses a `BinManager` object to manage its warehouse bins. It should initialize it to use nine of the bins, holding the following item descriptions and quantities. The bin index where the item will be stored is also shown here.

1. regular pliers 25
2. n. nose pliers 5
3. screwdriver 25
4. p. head screw driver 6
5. wrench-large 7
6. wrench-small 18
7. drill 51
8. cordless drill 16
9. hand saw 12

The modular client program should have functions to display a menu, get and validate the user's choice, and carry out the necessary activities to handle that choice. This includes adding items to a bin, removing items from a bin, and displaying a report of all bins. Think about what calls the `displayReport` client function will need to make to the `BinManager` object to create this report. When the user chooses the "Quit" option from the menu, the program should call its `displayReport` function one last time to display the final bin information. All I/O should be done in the client class. The `BinManager` class only accepts information, keeps the array of `InvBin` objects up to date, and returns information to the client program.

Input Validation: The `BinManager` class functions should not accept numbers less than 1 for the number of parts being added or removed from a bin. They should also not allow the user to remove more items from a bin than it currently holds.

Group Projects

[illegible]

19. Tic-Tac-Toe Game

Write a modular program that allows two players to play a game of tic-tac-toe. Use a two-dimensional `char` array with three rows and three columns as the game board. Each element of the array should be initialized with an asterisk (*). The program should display the initial board configuration and then start a loop that does the following:

- Have player 1 select a board location for an X by entering a row and column number. Then redisplay the board with an X replacing the * in the chosen location.
- If there is no winner yet and the board is not yet full, have player 2 select a board location for an O by entering a row and column number. Then redisplay the board with an O replacing the * in the chosen location.

The loop should continue until a player has won or a tie has occurred, then display a message indicating who won, or reporting that a tie occurred.

- Player 1 wins when there are three Xs in a row, a column, or a diagonal on the game board.
- Player 2 wins when there are three Os in a row, a column, or a diagonal on the game board.
- A tie occurs when all of the locations on the board are full, but there is no winner.

Input Validation: Only allow legal moves to be entered. The row and column must be 1, 2, or 3. The selected board location must currently be empty (i.e., still have an asterisk in it).

8, 20. Theater Ticket Sales

Create a TicketManager class and a program that uses it to sell tickets for a single performance theater production. This project is intended to be designed and written by a team of two to four students. Here are some suggestions:

- One student might design and write the client program that uses the class, while other team members design and write the TicketManager class and all of its functions.
- Each student should be given about the same workload.
- The class design and the names, parameters, and return types of each function should be decided in advance.
- The project can be implemented as a multifile program, or all the functions can be cut and pasted into a single file.

Here are the specifications:

- The theater's auditorium has 15 rows, with 30 seats in each row. To represent the seats, the TicketManager class should have a two-dimensional array of SeatStructures. Each of these structures should have data members to keep track of the seat's price and whether or not it is available or already sold.
- The data for the program is to be read in from two files located in the Chapter 8 programs folder on this book's companion website. The first one, SeatPrices.dat, contains 15 values representing the price for each row. All seats in a given row are the same price, but different rows have different prices. The second file, SeatAvailability.dat, holds the seat availability information. It contains 450 characters (15 rows with 30 characters each), indicating which seats have been sold ('*') and which are available ('#'). Initially all seats are available. However, once the program runs and the file is updated, some of the seats will have been sold. The obvious function to read in the data from these files and set up the array is the constructor that runs when the TicketManager object is first created.
- The client program should be a menu-driven program that provides the user with a menu of box office options, accepts and validates user inputs, and calls appropriate class functions to carry out desired tasks. The menu should have options to display the seating chart, request tickets, print a sales report, and exit the program.
- When the user selects the *display seats* menu option, a TicketManager function should be called that creates and returns a string holding a chart, similar to the one shown here. It should indicate which seats are already sold (*) and which are still available for purchase (#). The client program should then display the string.

```
                              Seats
            123456789012345678901234567890
Row  1      ***###***###******############
Row  2      ####*************####*******##
Row  3      **###**********########****###
Row  4      **######**************##******
Row  5      ********#####*********########
Row  6      ##############************####
Row  7      #######************###########
Row  8      ************##****############
Row  9      #########*****############****
Row 10      #####*************############
Row 11      #**********#################**
Row 12      #############********########*
Row 13      ###***********#########**#####
Row 14      ##############################
Row 15      ##############################
```

- When the user selects the *request tickets* menu option, the program should prompt for the number of seats the patron wants, the desired row number, and the desired starting seat number. A `TicketManager` ticket request function should then be called and passed this information so that it can handle the ticket request. If any of the requested seats do not exist, or are not available, an appropriate message should be returned to be displayed by the client program. If the seats exist and are available, a string should be created and returned that lists the number of requested seats, the price per seat in the requested row, and the total price for the seats. Then the user program should ask if the patron wishes to purchase these seats.

- If the patron indicates they do want to buy the requested seats, a `TicketManager` purchase tickets module should be called to handle the actual sale. This module must be able to accept money, ensure that it is sufficient to continue with the sale, and if it is, mark the seat(s) as sold, and create and return a string that includes a ticket for each seat sold (with the correct row, seat number, and price on it).

- When the user selects the *sales report* menu option, a `TicketManager` report module should be called. This module must create and return a string holding a report that tells how many seats have been sold, how many are still available, and how much money has been collected so far for the sold seats. Think about how your team will either calculate or collect and store this information so that it will be available when it is needed for the report.

- When the day of ticket sales is over and the *quit* menu choice is selected, the program needs to be able to write the updated seat availability data back out to the file. The obvious place to do this is in the `TicketManager` destructor.