EXERCISES (Iteration)

from **Object-Oriented ProgrammingUsing C++ [**Fourth Edition] By ***Joyce Farrell***

1. Write a C++ program that allows the user to enter an integer that represents the number of times a message will display. Display a greeting as many times as requested. Save the file as **MultipleGreetings.cpp**.

2. a. Write a program that allows the user to enter two integer values. Display every whole number that falls between these values. Save the file as **InBetween.cpp**.

   b. Modify the InBetween program so that the user must reenter the second value so it is guaranteed to be higher than the first entered value. Save the file as **InBetween2.cpp**.

3. Write a program that asks a user to enter an integer between 1 and 10. Continue to prompt the user while the value entered does not fall within this range. When the user is successful, display a congratulatory message. Save the file as **OneToTen.cpp**.

4. Write a program that asks a user to enter an integer between 1 and 10. Continue to prompt the user while the value entered does not fall within this range. When the user is successful, display a congratulatory message as many times as the value of the successful number the user entered. Save the file as **OneToTenMessage.cpp**.

5. Write an application that prints all even numbers from 2 to 100 inclusive. Save the file as **EvenNums.cpp**.

6. Write an application that asks a user to type A, B, C, or Q to quit. When the user types Q, the program ends. When the user types A, B, or C, the program displays the message "Good job!" and then asks for another input. When the user types anything else, issue an error message and then ask for another input. Save the file as **ABCInput.cpp**.

7. Write an application that displays every integer value from 1 to 20 along with its squared value. Save the file as **TableOfSquares.cpp**.

8. Write an application that sums the integers from 1 to 50 (that is, 1 + 2 + 3 . . . + 50) and displays the result. Save the file as **Sum50.cpp**.

9. Write an application that shows the sum of 1 to **n** for every **n** from 1 to 50. That is, the program prints 1 (the sum of 1 alone), 3 (the sum of 1 and 2), 6 (the sum of 1, 2, and 3), 10 (the sum of 1, 2, 3, and 4), and so on. Save the file as **EverySum.cpp**.

10. Write an application that displays every perfect number from 1 through 1000. A perfect number is one that equals the sum of all the numbers that divide evenly into it. For example, 6 is perfect because 1, 2, and 3 divide evenly into it, and their sum is 6; however, 12 is not a perfect number because 1, 2, 3, 4, and 6 divide evenly into it, and

their sum is greater than 12. Save the file as **Perfect.cpp**.

11. Write an application that calculates the amount of money earned on an investment, based on an 8 percent annual return. Prompt the user to enter an investment amount and the number of years for the investment. Do not allow the user to enter a number of years less than 1 or more than 30. Display the total amount (balance) for each year of the investment. Save the file as **Investment.cpp**.

12. Write an application that creates a quiz that contains at least five questions about a hobby, book, astronomy, or any other personal interest. Each question can be multiple choice (for which valid responses are a, b, c, or d), or true/false (for which valid responses are t and f). If the user responds to a question with an invalid character, display an error message and prompt the user again. If the user answers the question with a valid and correct response, display an appropriate message. If the user responds to a question with a valid but incorrect response, display an appropriate message as well as the correct answer. At the end of the quiz, display the number of correct and incorrect answers. Save the file as **Quiz.cpp**.

13. Write a program that generates a random number from 1 through 100 and allows the user to guess the number. After each guess, display a message that informs the user if the guess was too high, too low, or correct. Allow the user to play until the number is guessed correctly, and then display a count of the number of guesses that were needed. Save the program as **LoopingGuessingGame.cpp**.

14. Create a structure named Purchase. Each Purchase contains an invoice number, amount of sale, and amount of sales tax. Create a `main()` method that declares a Purchase object and prompts the user for purchase details. When you prompt for an invoice number, do not let the user proceed until a number between 1000 and 8000 has been entered. When you prompt for a sale amount, do not proceed until the user has entered a non-negative value. Compute the sales tax as 5 percent of the purchase price. After a valid Purchase object has been created, display the object's invoice number, sale amount, and sales tax. Save the file as **CreatePurchase.cpp**.[**ADVANCED**]

15. Create an Investment structure. Include fields for the term of the Investment in years, the beginning dollar amount of the Investment, and the final value of the Investment. Write a `main()` function in which you declare an Investment object. Prompt the user for the term and the initial Investment amount for this object. Display the value of the Investment after each year of the term, using a simple compound interest rate of 8 percent. At the end of the program, display all the final field values. Save the file as **Investucpp**. [**ADVANCED**]