2

Introduction to C++ Programming

What's in a name?
that which we call a rose
By any other name
would smell as sweet.
—William Shakespeare

When faced with a decision, I always ask, "What would be the most fun?"
—Peggy Walker

"Take some more tea," the March Hare said to Alice, very earnestly. "I've had nothing yet," Alice replied in an offended tone: "so I can't take more." "You mean you can't take less," said the Hatter: "it's very easy to take more than nothing."
—Lewis Carroll

High thoughts must have high language.

Aristophanes
—Aristophanes

OBJECTIVES

In this chapter you'll learn:

- To write simple computer programs in C++.
- To write simple input and output statements.
- To use fundamental types.
- Basic computer memory concepts.
- To use arithmetic operators.
- The precedence of arithmetic operators.
- To write simple decision-making statements.

Assignment Checklist

Name: _____ Date: _____

Section: _____

Exercises	Assigned: Circle assignments	Date Due
Prelab Activities		
Matching	YES NO	
Fill in the Blank	14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25	
Short Answer	26, 27, 28, 29, 30, 31, 32	
Programming Output	33, 34, 35, 36, 37, 38, 39	
Correct the Code	40, 41, 42, 43, 44, 45	
Lab Exercises		
Exercise 1 — Sum, Average, Product, Smallest and Largest	YES NO	
Follow-Up Questions and Activities	1, 2	
Exercise 2 — Multiples	YES NO	
Follow-Up Questions and Activities	1, 2, 3, 4	
Exercise 3 — Separating Digits	YES NO	
Follow-Up Questions and Activities	1, 2, 3	
Debugging	YES NO	
Labs Provided by Instructor		
1.		
2.		
3.		
Postlab Activities		
Coding Exercises	1, 2, 3, 4, 5, 6, 7	
Programming Challenges	1, 2, 3	

Prelab Activities

Matching

Name: _____ **Date:** _____

Section: _____

After reading Chapter 2 of *C++ How to Program, Seventh Edition*, answer the given questions. These questions are intended to test and reinforce your understanding of key concepts and may be done either before the lab or during the lab.

For each term in the column on the left, write the corresponding letter for the description that best matches it from the column on the right.

Term	Description
___ 1. Integer division	a) Holds whole number values.
___ 2. Stream extraction operator	b) Outputs a newline and "flushes the output buffer."
___ 3. `return`	c) Appears at the end of every statement.
___ 4. Modulus operator	d) An operation that truncates any fractional part of its result.
___ 5. A variable of type `int`	e) Instruction that is performed before the program is compiled.
___ 6. Comments	f) Prevents a program from compiling.
___ 7. Stream insertion operator	g) An operation that yields the remainder after integer division.
___ 8. Preprocessor directive	h) `>>`.
___ 9. `std::endl` stream manipulator	i) One of several means to exit a function.
___ 10. Semicolon	j) `<<`.
___ 11. Conditions in `if` statements	k) `'\n'`.
___ 12. Newline escape sequence	l) Text that documents programs and improves their readability.
___ 13. Syntax error	m) Commonly formed by using equality operators and relational operators.

Prelab Activities

Name:

Fill in the Blank

Name: _____ **Date:** _____

Section: _____

Fill in the blanks in each of the following statements:

14. Both _____ and _____ are ignored by the C++ compiler.

15. A backslash is combined with the next character to form a(n) _____.

16. A(n) _____ is a location in the computer's memory where a value can be stored for use by a program.

17. Output and input in C++ are accomplished with _____ of characters.

18. Single-line comments begin with _____.

19. _____ is the first function executed in a C++ program.

20. All variables in a C++ program must be _____ before they are used.

21. _____ represents the standard input stream.

22. _____ represents the standard output stream.

23. The _____ statement allows a program to make a decision.

24. C++ evaluates arithmetic expressions in a precise sequence determined by the rules of _____ and associativity.

25. _____ operators and _____ operators are commonly used in `if` conditions.

Prelab Activities

Name:

Short Answer

Name: _____ **Date:** _____

Section: _____

In the space provided, answer each of the given questions. Your answers should be as concise as possible; aim for two or three sentences.

26. What is the difference between stream insertion and stream extraction? What is each used for?

27. What is a syntax error? Give an example.

28. What is a logic error? Give an example.

29. What are operator precedence and associativity? How do they affect program execution?

Prelab Activities

Name:

Short Answer

30. What are redundant parentheses? When might a programmer use them?

31. Write an example of a preprocessor directive.

32. What is a variable? How are they used in computer programs?

Prelab Activities

Name:

Programming Output

Name: _____ **Date:** _____

Section: _____

For each of the given program segments, read the code and write the output in the space provided below each program. [*Note:* Do not execute these programs on a computer.]

33. What is the output of the following program?

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6     int x;
7     int y;
8
9     x = 30;
10    y = 2;
11    cout << x * y + 9 / 3 << endl;
12 } // end main
```

Your answer:

34. What is output by the following line of code?

```
1  cout << ( 8 * 4 * 2 + 6 ) / 2 + 4;
```

Your answer:

Prelab Activities

Name:

Programming Output

For *Programming Output Exercises 35* and *36*, use the program in Fig. L 2.1.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6     int input;
7
8     cout << "Please enter an integer: ";
9     cin >> input;
10
11    if ( input != 7 )
12       cout << "Hello" << endl;
13
14    if ( input == 7 )
15       cout << "Goodbye" << endl;
16 } // end main
```

Fig. L 2.1 | Program used for *Programming Output Exercises 35* and *36*.

35. What is output by the program in Fig. L 2.1? Assume that the user enters `5` for `input`.

Your answer:

36. What is output by the program in Fig. L 2.1? Assume that the user enters `7` for `input`.

Your answer:

Prelab Activities

Name:

Programming Output

For *Programming Output Exercises 37* and *38*, use the program in Fig. L 2.2.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6     int input;
7
8     cout << "Please enter an integer: ";
9     cin >> input;
10
11    if ( input >= 0 )
12       cout << "Hello" << endl;
13
14    cout << "Goodbye" << endl;
15 } // end main
```

Fig. L 2.2 | Program used for *Programming Output Exercises 37* and *38*.

37. What is output by the program in Fig. L 2.2? Assume the user enters `2` for `input`.

Your answer:

38. What is output by the program in Fig. L 2.2? Assume the user enters `-2` for `input`.

Your answer:

Prelab Activities

Name:

Programming Output

39. What is output by the following program?

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6     int x = 3;
7     int y = 9;
8     int z = 77;
9
10    if ( x == ( y / 3 ) )
11       cout << "H";
12
13    if ( z != 77 )
14       cout << "q";
15
16    if ( z == 77 )
17       cout << "e";
18
19    if ( z * y + x < 0 )
20       cout << "g";
21
22    if ( y == ( x * x ) )
23       cout << "ll";
24
25    cout << "o!" << endl;
26 } // end main
```

Your answer:

Prelab Activities

Name:

Correct the Code

Name: _____ **Date:** _____

Section: _____

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic or compilation error, circle the error in the code and write the corrected code in the space provided after each problem. If the code does not contain an error, write "no error." For code segments, assume the code appears in `main` and that `using` directives are provided. [*Note*: It is possible that a program segment may contain multiple errors.]

40. The following program should print an integer to the screen:

```
1  #include <iostream>;
2  using namespace std
3
4  int main()
5  {
6     int x = 30;
7     int y = 2;
8
9     cout << x * y + 9 / 3 << endl;
10 } // end main
```

Your answer:

Prelab Activities

Name:

Correct the Code

41. The following code should declare an integer variable and assign it the value 6.

```
1  int 1stPlace
2  1stPlace = 6;
```

Your answer:

42. The following code should determine whether variable x is less than or equal to 9.

```
1  int x = 9;
2
3  if ( x < = 9 )
4     cout << "Less than or equal to.";
```

Your answer:

Prelab Activities

Name:

Correct the Code

43. The following code should determine whether q is equal to 10.

```
1  int q = 10;
2
3  cout << "q is: " << q << endl;
4
5  if ( q = 10 )
6     cout << "q is equal to 10";
```

Your answer:

44. The following code segment should determine whether an integer variable's value is greater than zero and display an appropriate message.

```
1  int x = 9;
2
3  if ( x > 0 );
4     cout << "Greater than zero";
```

Your answer:

Prelab Activities

Name:

Correct the Code

45. The following program should print 302 to the screen:

```
1  #include <iostream>
2  using namespace std;
3
4  int ma in()
5  {
6     int x = 30;
7     int y = 2;
8
9     cout << y << x << endl;
10 } // end main
```

Your answer:

Lab Exercises

Lab Exercise 1 — Sum, Average, Product, Smallest and Largest

Name: _____ **Date:** _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 2.3)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 2 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using `cout` to output text and variables.
- Using `cin` to input data from the user.
- Using `if` statements to make decisions based on the truth or falsity of a condition.
- Using the arithmetic operators to perform calculations.
- Using relational operators to compare values.

The follow-up questions and activities also will give you practice:

- Comparing < to <=.
- Modifying existing code to perform the same task in a different manner.

Description of the Problem

Write a program that inputs three integers from the keyboard, and prints the sum, average, product, smallest and largest of these numbers. The screen dialogue should appear as follows: [*Note:* `13`, `27` and `14` are input by the user.]

Sample Output

```
Input three different integers: 13 27 14
Sum is 54
Average is 18
Product is 4914
Smallest is 13
Largest is 27
```

Lab Exercises

Name:

Lab Exercise 1 — Sum, Average, Product, Smallest and Largest

Template

```
// Lab 1: numbercompare.cpp
#include <iostream> // allows program to perform input and output
using namespace std;

int main()
{
   int number1; // first integer read from user
   int number2; // second integer read from user
   int number3; // third integer read from user
   int smallest; // smallest integer read from user
   int largest; // largest integer read from user

   cout << "Input three different integers: "; // prompt
   /* Write a statement to read in values for number1, number2 and
      number3 using a single input statement */

   largest = number1; // assume first integer is largest

   /* Write a statement to determine if number2 is greater than
      largest. If so assign number2 to largest */

   /* Write a statement to determine if number3 is greater than
      largest. If so assign number3 to largest */

   smallest = number1; // assume first integer is smallest

   /* Write a statement to determine if number2 is less than
      smallest. If so assign number2 to smallest */

   /* Write a statement to determine if number3 is less than
      smallest. If so assign number3 to smallest */

   /* Write an output statement that prints the sum, average,
      product, largest and smallest */
} // end main
```

Fig. L 2.3 | `numbercompare.cpp`.

Problem-Solving Tips

1. Prompt the user to input three integer values. You will use a single input statement to read all three values.
2. Sometimes it is useful to make an assumption to help solve or simplify a problem. For example, we assume `number1` is the largest of the three values and assign it to `largest`. You will use `if` statements to determine whether `number2` or `number3` are larger.
3. Using an `if` statement, compare `largest` to `number2`. If the content of `number2` is larger, then store the variable's value in `largest`.
4. Using an `if` statement, compare `largest` to `number3`. If the content of `number3` is larger, then store the variable's value in `largest`. At this point you are guaranteed to have the largest value stored in `largest`.
5. Perform similar steps to those in Steps 2–4 to determine the smallest value.
6. Write a statement that outputs the sum, average, product (i.e., multiplication), largest and smallest values.
7. Be sure to test your program with all possible combinations of values.

Lab Exercises

Name:

Lab Exercise 1 — Sum, Average, Product, Smallest and Largest

8. If you have any questions as you proceed, ask your lab instructor for assistance.

Follow-Up Questions and Activities

1. Modify your solution to use three separate input statements rather than one. Write a separate prompt for each `cin`.

2. Does it matter whether < or <= is used when making comparisons to determine the smallest integer? Which did you use and why?

Lab Exercises

Name:

Lab Exercise 2 — Multiples

Name: _____ **Date:** _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 2.4)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 2 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using `cout` to output text and values.
- Using `cin` to input data from the user.
- Using `if` statements to make decisions based on the truth or falsity of a condition.
- Using the modulus operator (`%`) to determine the remainder of an integer division operation.

The follow-up questions and activities also will give you practice:

- Understanding the modulus operator.
- Recognizing common mistakes with the `if` statement.
- Adapting a program to solve a similar problem.

Description of the Problem

Write a program that reads in two integers and determines and prints whether the first is a multiple of the second. [*Hint*: Use the modulus operator.]

Sample Output

```
Enter two integers: 22 8
22 is not a multiple of 8
```

Lab Exercises

Name:

Lab Exercise 2 — Multiples

Template

```
1  // Lab 2: multiples.cpp
2  #include <iostream> // allows program to perform input and output
3  using namespace std;
4
5  int main()
6  {
7     /* Write variables declarations here */
8
9     cout << "Enter two integers: "; // prompt
10    /* Write an input statement to read data into variables here */
11
12    // using modulus operator
13    if ( /* Write a condition that tests whether number1 is a multiple of
14            number2 */)
15       cout << number1 << " is a multiple of " << number2 << endl;
16
17    if ( /* Write a condition that tests whether number1 is not a multiple
18            of number2 */ )
19       cout << number1 << " is not a multiple of " << number2 << endl;
20 } // end main
```

Fig. L 2.4 | `multiples.cpp`.

Problem-Solving Tips

1. The input data consists of two integers, so you will need two `int` variables to store the input values.
2. Use `cin` to read the user input into the `int` variables.
3. Use an `if` statement to determine whether the first number input is a multiple of the second number input. Use the modulus operator, `%`. If one number divides into another evenly, the modulus operation results in `0`. If the result is `0`, display a message indicating that the first number is a multiple of the second number.
4. Use an `if` statement to determine whether the first number input is not a multiple of the second number input. If one number does not divide into another evenly, the modulus operation results in a non-zero value. If non-zero, display a message indicating that the first number is not a multiple of the second.
5. Be sure to follow the spacing and indentation conventions mentioned in the text.
6. If you have any questions as you proceed, ask your lab instructor for assistance.

Lab Exercises

Name:

Lab Exercise 2 — Multiples

Follow-Up Questions and Activities

1. Can the modulus operator be used with non-integer operands? Can it be used with negative numbers? What is output by each expression in Fig. L 2.5? If there is an error, explain why.

Expression	Output
`cout << 73 % 22;`	_____
`cout << 0 % 100;`	_____
`cout << 100 % 0;`	_____
`cout << -3 % 3;`	_____
`cout << 9 % 4.5;`	_____
`cout << 16 % 2;`	_____

Fig. L 2.5 | Determine the output of the `cout` statements in the third column.

2. Place a semicolon at the end of the `if` statement in your solution that corresponds to the `if` statement in lines 16–18 in the template. What happens? Explain.

3. Rewrite the output statement in your solution that corresponds to the output statement in line 15 in the template. This statement should now look as follows:

```
cout << number1;
cout << " is a multiple of ";
cout << number2 << endl;
```

Rerun the program and observe the differences. Why is the output different?

Lab Exercises

Name:

Lab Exercise 2 — Multiples

4. Modify the program to determine whether a number entered is even or odd. [*Note:* Now, the user needs to enter only one number.]

Lab Exercises

Name:

Lab Exercise 3 — Separating Digits

Name: _____ **Date:** _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 2.6)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working C++ program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with C++ code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 2 of *C++ How To Program, Seventh Edition*. In this lab, you will practice:

- Using the modulus operator (`%`) to determine the remainder of a division operation.
- Integer division, which differs from floating-point division because integer division truncates the decimal portion of the result.

The follow-up questions and activities also will give you practice:

- Using the division and modulus operators.
- Examining what happens during program execution when the user enters invalid input.
- Adapting a program to solve a similar problem.

Problem Description

Write a program that inputs a five-digit number, separates the number into its individual digits and prints the digits separated from one another by three spaces each. [*Hint:* Use integer division and the modulus operator.] For example, if the user inputs **`42339`**, the program should print what is shown in the sample output.

Sample Output

```
4   2   3   3   9
```

Lab Exercises

Name:

Lab Exercise 3 — Separating Digits

Template

```
1  // Lab 3: digits.cpp
2  #include <iostream> // allows program to perform input and output
3  using namespace std;
4
5  int main()
6  {
7     int number; // integer read from user
8
9     cout << "Enter a five-digit integer: "; // prompt
10    cin >> number; // read integer from user
11
12    /* Write a statement to print the left-most digit of the
13       5-digit number */
14    /* Write a statement that changes number from 5-digits
15       to 4-digits */
16    /* Write a statement to print the left-most digit of the
17       4-digit number */
18    /* Write a statement that changes number from 4-digits
19       to 3-digits */
20    /* Write a statement to print the left-most digit of the
21       3-digit number */
22    /* Write a statement that changes number from 3-digits
23       to 2-digits */
24    /* Write a statement to print the left-most digit of the
25       2-digit number */
26    /* Write a statement that changes number from 2-digits
27       to 1-digit */
28    cout << number << endl;
29 } // end main
```

Fig. L 2.6 | `digits.cpp`.

Problem-Solving Tips

1. The input data consists of one integer, so you will use an `int` variable (`number`) to represent it. Note that the description indicates that one five-digit number is to be input—not five separate digits.
2. You will use a series of statements to "break down" the number into its individual digits using modulus (`%`) and division (`/`) calculations.
3. After the number has been input using `cin`, divide the number by `10000` to get the leftmost digit. Why does this work? In C++, dividing an integer by an integer results in an integer. For example, `42339 / 10000` evaluates to `4` because `10000` divides evenly into `42339` four times. The remainder `2339` is truncated.
4. Change the number to a 4-digit number using the modulus operator. The number modulus `10000` evaluates to the integer remainder—in this case, the right-most four digits. For example, `42339 % 10000` results in `2339`. Assign the result of this modulus operation to the variable that stores the five-digit number input.
5. Repeat this pattern of division and modulus reducing the divisor by a factor of `10` each time (i.e., `1000`, `100`, `10`). After the number is changed to a four-digit number, divide/modulus by `1000`. After the number is changed to a three-digit number, divide/modulus by `100`. And so on.
6. Be sure to follow the spacing and indentation conventions mentioned in the text.
7. If you have any questions as you proceed, ask your lab instructor for assistance.

Lab Exercises

Name:

Lab Exercise 3 — Separating Digits

Follow-Up Questions and Activities

1. What are the results of the following expressions?

   `24 / 5 =` _____

   `18 % 3 =` _____

   `13 % 9 =` _____

   `13 / 2 % 2 =` _____

2. What happens when the user inputs a number which has fewer than five digits? Why? What is the output when `1763` is entered?

3. The program you completed in this lab exercise inputs a number with multiple digits and separates the digits. Write the inverse program, a program which asks the user for three one-digit numbers and combines them into a single three-digit number. [*Hint:* Use multiplication and addition to form the three-digit number.]

Lab Exercises

Name:

Debugging

Name: _____ **Date:** _____

Section: _____

The program in this section does not compile and run properly. Fix all the compilation errors so that the program will compile successfully. Once the program compiles, compare the output with the sample output, and eliminate any logic errors that may exist. The sample output demonstrates what the program's output should be once the program's code has been corrected. `debugging02.cpp` (Fig. L 2.7) is available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`.

Sample Output

```
Enter two integers to compare: 5 2
5 != 2
5 > 2
5 >= 2
```

```
Enter two integers to compare: 2 7
2 != 7
2 < 7
2 <= 7
```

```
Enter two integers to compare: 4 4
4 == 4
4 <= 4
4 >= 4
```

Broken Code

```
1  // Debugging
2  include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int number1;
8     int number2;
9
10    cout << "Enter two integers to compare: ";
11    cout >> number1 >> number2;
12
13    if ( number1 == number2 )
14       cout << number1 << ' == ' << number2 << endl;
15
```

Fig. L 2.7 | debugging02.cpp. (Part 1 of 2.)

Lab Exercises

Name:

Debugging

```
16     if ( number1 <> number2 )
17        cout << number1 << " <> " << number2 << endl;
18
19     if ( number2 < number1 )
20        cout << number1 << " < " << number2 << endl;
21
22     if  number1 > number2 )
23        cout << number1 << " > " << number2 << endl;
24
25     if ( number1 < number2 )
26        cout << number1 << " <= " << number2 << endl;
27
28     if ( number1 >= number2 )
29        cout << number1 << " >= " << number2 << endl
30  } // end main
```

Fig. L 2.7 | `debugging02.cpp`. (Part 2 of 2.)

Postlab Activities

Coding Exercises

Name: _____ **Date:** _____

Section: _____

These coding exercises reinforce the lessons learned in the lab and provide additional programming experience outside the classroom and laboratory environment. They serve as a review after you have completed the Prelab Activities and Lab Exercises successfully.

For each of the following problems, write a program or a program segment that performs the specified action.

1. Write the preprocessor directive which includes the `iostream` file in a program. Also write an appropriate `using` directive.

2. Define a `main` function which declares three integer variables.

3. Write a single line of code that reads values into the three integer variables from *Coding Exercise 2*.

Postlab Activities

Name:

Coding Exercises

4. Write a line of code that prints all three integer values from *Coding Exercise 3* separated by hyphens, -.

5. Modify your solution to *Coding Exercise 4* to write a C++ program that determines which variable's value is the largest. Use variable `largest` to store the largest value.

Postlab Activities

Name:

Coding Exercises

6. Modify your solution to *Coding Exercise 5* to write a C++ program that determines which integer variable's value is the smallest. Use variable `smallest` to store the smallest value.

7. Modify your solution to *Coding Exercise 6* to test if any of the variable's values are equal and if so print that they are equal. For example, if two variables have the same value, `5`, print "`5 and 5 are equal`." You'll need to test all three possibilities.

Postlab Activities

Name:

Programming Challenges

Name: _____ **Date:** _____

Section: _____

The *Programming Challenges* are more involved than the *Coding Exercises* and may require a significant amount of time to complete. Write a C++ program for each of the problems in this section. The answers to these problems are available from the Companion Website for *C++ How to Program, Seventh Edition* at `www.pearsonhighered.com/deitel/`. Pseudocode, hints and/or sample outputs are provided to aid you in your programming.

1. Write a program that prints the numbers 1 to 4 on the same line with each pair of adjacent numbers separated by one space. Write the program using the following methods:
   a) Using one output statement with one stream-insertion operator.
   b) Using one output statement with four stream-insertion operators.
   c) Using four output statements.

Hints:

- Use comments to separate your program into three clearly marked sections, one for each part (i.e., a–c) of the problem.
- For Part a) the entire output should be contained within one string.
- Use either `endl` or `"\n"` after each part to separate their outputs.
- Your output should look like:

```
1 2 3 4
1 2 3 4
1 2 3 4
```

2. Write a program that asks the user to enter two integers, obtains the numbers from the user, then prints the larger number followed by the words "`is larger`" If the numbers are equal, print the message "`These numbers are equal`"

Hints:

- The user should input both integers at once, i.e., `cin >> x >> y;`
- A typical run of your program might look as follows:

```
Enter two integers: 5 3
5 is larger
```

```
Enter two integers: 6 6
These numbers are equal
```

Postlab Activities

Name:

Programming Challenges

3. Write a program that reads in five integers and determines and prints the largest and the smallest integers in the group. Use only the programming techniques you learned in Chapter 2 of *C++ How to Program, Seventh Edition.*

Hints:

- This program requires seven variables: five for user input and two to store the largest and the smallest, respectively.
- As soon as the user inputs the values, assign the `largest` and `smallest` variables the value of the first input. If instead, `largest` was initially assigned to zero, this would be a logic error because negative numbers could be input by the user.
- Ten separate `if` statements are required to compare each input to `largest` and `smallest`.
- A typical run of your program might look as follows:

```
Enter five integers: 2 4 -4 10 3
Largest integer: 10
Smallest integer: -4
```