Answers to Review Questions

Chapter 1

1. BCPL or Algol
2. True
3. 1980s
4. False. C++'s compact size makes it an excellent programming language for smaller computers.
5. The hard disk
6. A modem
7. b. Input. By moving the mouse, you give cursor-direction commands to the computer.
8. NumLock
9. UNIX

10. When you turn off the computer, the contents of RAM are destroyed.
11. True
12. 524,288 bytes (512 times 1,024)
13. *Mo*dulate, *dem*odulate

Chapter 2

1. A set of detailed instructions that tells the computer what to do.
2. Buy one or write it yourself.
3. False
4. The program produces the output.
5. A program editor
6. The .CPP extension
7. You must first plan the program by deciding which steps you will take to produce the final program.
8. To get the errors out of your program
9. So your programs work with various compilers and computer equipment
10. False. You must compile a program before linking it. Most compilers link the program automatically.

Chapter 3

1. Two comment markers (`//`)
2. A holding place for data that can be changed
3. A value that cannot be changed
4. The `+`, `-`, `*`, and `/` operators

5. The `=` assignment operator.
6. False. There are floating-point, double floating-point, short integers, long integers, and many more variable data types.
7. `cout`
8. `city` must be a variable name because it is not enclosed in quotation marks.
9. All C++ commands must be in lowercase.

Chapter 4

1. `my_name` and `sales_89`
2. Characters: `‘X’` and `‘0’`

   Strings: `“2.0”` and `“X”`

   Integers: `0` and `-708`

   Floating-point literals: `-12.0` and `65.4`
3. Seven variables are declared: three integers, three characters, and one floating-point variable.
4. A null zero, also called a binary zero or an ASCII zero.
5. True
6. 1
7. It is stored as a series of ASCII values, representing the characters and blanks in the string, ending in an ASCII `0`.
8. It is stored as a single ASCII `0`.
9. The constant value called `age` cannot be changed.

Chapter 5

1. `char my_name[] “This is C++”;`
2. The string is 11 characters long.

3. It consumes 12 bytes.
4. All string literals end with a binary zero.
5. Two character arrays are declared, each with 25 elements.
6. False. The keyword `char` must precede the variable name.
7. True. The binary zero terminates the string.
8. False. The characters do not represent a string because there is no terminating zero.

Chapter 6

1. False. You can define only constants with the `#define` preprocessor directive.
2. The `#include` directive
3. The `#define` directive
4. True
5. The preprocessor changes your source code before the compiler reads the source code.
6. The `const` keyword
7. Use angled brackets when the `include` files reside in the compiler's `include` subdirectory. Use quotation marks when the `include` file resides in the same subdirectory as the source program.
8. Defined literals are easier to change because you have to change only the line with `#define` and not several other lines in the program.
9. iostream.h
10. False. You cannot define constants enclosed in quotation marks (as `"MESSAGE"` is in the `cout` statement).
11. `Amount is 4`

Chapter 7

1. `cout` sends output to the screen, and `cin` gets input from the keyboard.
2. The prompt tells the user what is expected.
3. The user enters four values.
4. `cin` assigns values to variables when the user types them, whereas the programmer must assign data when using the assignment operator (`=`).
5. True. When printing strings, you do not need `%s`.
6. Arrays
7. `The backslash “\” character is special`
8. The following value prints, with one leading space: `123.456`

Chapter 8

1. a. 5
   b. 6
   c. 5
2. a. 2
   b. 7
3. a. `a = (3+3) / (4+4);`
   b. `x = (a-b)*( (a-c) * (a-c));`
   c. `f = (a*a)/(b*b*b);`
   d. `d = ((8 - x*x)/(x - 9))-((4*2 - 1)/(x*x*x));`
4. The area of a circle:

```
#include stdio.h>
const float PI = 3.14159;
main()
```

```
{
   printf(“%f”, (PI*(4*4));
   return;
}
```

5. Assignment and `printf()` statements:

```
r = 100%4;
cout << r;
```

Chapter 9

1. The `==` operator
2. a. True
   b. True
   c. True
   d. True
3. True
4. The `if` statement determines what code executes when the relational test is true. The `if-else` statement determines what happens for both the True and the False relational test.
5. No
6. a. False
   b. False
   c. False

Chapter 10

1. The `&&`, `||`, and `!` operators are the three logical operators.
2. a. False
   b. False

   c. True
   d. True
3. a. True
   b. True
   c. True
4. `g is 25 and f got changed to 8`
5. a. True
   b. True
   c. False
   d. True
6. Yes

Chapter 11

1. The `if-else` statement
2. The conditional operator is the only C++ operator with three arguments.
3.

```
if (a == b)
   { ans = c + 2; }
else
   { ans = c + 3; }
```

4. True
5. The increment and decrement operators compile into single assembly instructions.
6. A comma operator (`,`), which forces a left-to-right execution of the statements on either side
7. The output cannot be determined reliably. Do not pass an increment operator as an argument.

8.
   ```
   The size of name is 20
   ```
9. a. True
   b. True
   c. False
   d. False

Chapter 12

1. The `while` loop tests for a true condition at the beginning of the loop. The `do-while` tests for the condition at the end of the loop.
2. A counter variable increments by one. A total variable increments by the addition to the total you are performing.
3. The `++` operator
4. If the body of the loop is a single statement, the braces are not required. However, braces are *always* recommended.
5. There are no braces. The second `cout` always executes, regardless of the result of the `while` loop's relational test.
6. The stdlib.h header file
7. One time
8. By returning a value inside the `exit()` function's parentheses
9.
   ```
   This is the outer loop

   This is the outer loop

   This is the outer loop

   This is the outer loop
   ```

Chapter 13

1. A loop is a sequence of one or more instructions executed repeatedly.
2. False
3. A nested loop is a loop within a loop.
4. Because the expressions might be initialized elsewhere, such as before the loop or in the body of the loop
5. The inner loop

6.
```
10
7

4

1
```

---

7. True
8. The body of the `for` loop stops repeating.
9. False, due to the semicolon after the first `for` loop
10. There is no output. The value of `start` is already less than `end` when the loop begins; therefore, the `for` loop's test is immediately False.

Chapter 14

1. Timing loops force a program to pause.
2. Because some computers are faster than others.
3. If the `continue` and `break` statements were unconditional, there would be little use for them.
4. Because of the unconditional `continue` statement, there is no output.
5. `*****`

6. A single variable rarely can hold a large enough value for the timer's count.

Chapter 15

1. The program does not execute sequentially, as it would without `goto`.
2. The `switch` statement
3. A `break` statement
4. False because you should place the `case` most likely to execute at the beginning of the `case` options.
5.
```
switch (num)
{ case (1) : { cout << "Alpha";
               break; }
  case (2) : { cout << "Beta";
               break; }
  case (3) : { cout << "Gamma";
               break; }
  default :  { cout << "Other";
               break; }
}
```
6.
```
do
  { cout << "What is your first name? ";
    cin >> name;
  } while ((name[0] < 'A') || (name[0] > 'Z'));
```

Chapter 16

1. True
2. `main()`

3. Several smaller functions are better because each function can perform a single task.
4. Function names always end with a pair of parentheses.
5. By putting separating comments between functions.
6. The function `sq_25()` cannot be nested in `calc_it()`.
7. A function definition (a prototype).
8. True

Chapter 17

1. True
2. Local variables are passed as arguments.
3. False
4. The variable data types
5. Static
6. You should never pass global variables—they do not need to be passed.
7. Two arguments (the string `“The rain has fallen %d inches”`, and the variable, `rainf`)

Chapter 18

1. Arrays
2. Nonarray variables are always passed by value, unless you override the default with `&` before each variable name.
3. True
4. No
5. Yes

6. The data types of variables `x`, `y`, and `z` are not declared in the receiving parameter list.
7. c

Chapter 19

1. By putting the return type to the left of the function name.
2. One
3. To prototype built-in functions.
4. `int`
5. False
6. Prototypes ensure that the correct number of parameters is being passed.
7. Global variables are already known across functions.
8. The return type is `float`. Three parameters are passed: a character, an integer, and a floating-point variable.

Chapter 20

1. In the function prototypes.
2. Overloaded functions
3. Overloaded functions
4. False. You can specify multiple default arguments.
5. `void my_fun(float x, int i=7, char ch='A');`
6. False. Overloaded functions must differ in their argument lists, not only in their return values.

Chapter 21

1. For portability between different computers
2. False. The standard output can be redirected to any device through the operating system.
3. `getch()` assumes `stdin` for the input device.
4. `get`
5. `>` and `<`
6. `getche()`
7. False. The input from `get` goes to a buffer as you type it.
8. Enter
9. True

Chapter 22

1. The character-testing functions do not change the character passed to them.
2. `gets()` and `fgets()`
3. `floor()` rounds down and `ceil()` rounds up.
4. The function returns `0` (false) because `islower('s')` returns a `1` (true) and `isalpha(1)` is `0`.
5. `PeterParker`
6. `8 9`
7. True
8. `Prog` with a null zero at the end.
9. True

Chapter 23

1. False
2. The array subscripts differentiate array elements.
3. C does not initialize arrays for you.
4. `0`
5. Yes. All arrays are passed by address because an array name is nothing more than an address to that array.
6. C++ initializes all types of global variables (and every other static variable in your program) to zero or null zero.

Chapter 24

1. False
2. From the low numbers floating to the top of the array like bubbles.
3. Ascending order
4. The name of an array is an address to the starting element of that array.
5. a. `Eagles`
    b. `Rams`
    c. `les`
    d. `E`
    e. `E`
    f. The statement prints the character string, `s`.
    g. The third letter of “Eagles” (`g`) prints.

Chapter 25

1. `int scores[5][6];`
2. `char initials[4][10][20]`
3. The first subscript represents rows and the last represents columns.
4. 30 elements
5. a. 2
   b. 1
   c. 91
   d. 8
6. Nested `for` loops step through multidimensional tables very easily.
7. a. 78
   b. 100
   c. 90

Chapter 26

1. a. Integer pointer
   b. Character pointer
   c. Floating-point pointer
2. “Address of “
3. The `*` operator
4. `pt_sal = &salary;`
5. False
6. Yes
7. a. 2313.54

   b. 2313.54
   c. invalid
   d. invalid

8. b

Chapter 27

1. Array names are pointer constants, not pointer variables.
2. 8
3. *a*, *c*, and *d* are equivalent. Parentheses are needed around `iptr+4` and `iptr+1` to make *b* and *e* valid.
4. You have to move only pointers, not entire strings.
5. a and d

Chapter 28

1. Structures hold groups of more than one value, each of which can be a different data type.
2. Members
3. At declaration time and at runtime
4. Structures pass by copy.
5. False. Memory is reserved only when structure variables are declared.
6. Globally
7. Locally
8. 4

Chapter 29

1. True
2. Arrays are easier to manage.
3. a. `inventory[32].price = 12.33;`
   b. `inventory[11].part_no[0] = 'X';`
   c. `inventory[96] = inventory[62];`
4. a. `item` is not a structure variable.
   b. `inventory` is an array and must have a subscript.
   c. `inventory` is an array and must have a subscript.

Chapter 30

1. Write, append, and read.
2. Disks hold more data than memory.
3. You can access sequential files only in the same order that they were originally written.
4. An error condition occurs.
5. The old file is overwritten.
6. The file is created.
7. C++ returns an end-of-file condition.

Chapter 31

1. Records are stored in files and structures are stored in memory.
2. False
3. The file pointer continually updates to point to the next byte to read.

4. `read()` and `write()`
5. The `open()` function cannot be called without a filename.

Chapter 32

1. Data members and member functions
2. No
3. No
4. Private
5. Declare it with the `public` keyword.