Answers to Review Questions
Chapter 7

1. The size declarator is used in a definition of an array to indicate the number of elements the array will have. A subscript is used to access a specific element in an array.

2. The array has 10 elements.
The subscript of the first element is 0.
The subscript of the last element is 9.
Using four-byte integers, this array uses 40 bytes of memory.

3. Because, with the array alone the function has no way of determining the number of elements it has.

4. 2
   14
   8

5. By providing an initialization list. The array is sized to hold the number of values in the list.

6. 3
   0

7. Because an array name without brackets and a subscript represents the array's beginning memory address. The statement shown attempts to assign the address of array2 to array1, which is not permitted.

8. No.


10. The array's beginning memory address.

11. By using the same subscript value for each array.

12. Eight elements
    Yes, it is null terminated.

13. Eight rows
    Ten columns
    Eighty elements
    \texttt{sales[7][9] = 123.45;}

14. The second size declarator, which is for the number of columns.

15. • You do not have to declare the number of elements that a \texttt{vector} will have.
    • If you add a value to a \texttt{vector} that is already full, the \texttt{vector} will automatically increase its size to accommodate the new value.
    • A \texttt{vector} can report the number of elements it contains.

16. size declarator

17. integer, 0

18. subscript

19. 0

20. size declarator, subscript

21. bounds

22. initialization

23. 0
24. initialization list
25. null terminator
26. subscript
27. =
28. an address
29. address, or name
30. multi-dimensional
31. rows, columns
32. two
33. braces
34. column
35. Standard Template Library (or STL)
36. sequence and associative
37. sequence
38. vector
39. push_back
40. size
41. pop_back
42. clear

43. for (int i = 0; i < 20; i++)
    cout << names[i] << endl;

44. for (int i = 0; i < 100; i++)
    numberArray2[i] = numberArray1[i];

45. const int SIZE = 10;
    int id[SIZE];   // To hold ID numbers
    double weeklyPay[SIZE];  // To hold weekly pay
    // Display each employee's gross weekly pay.
    for (int i = 0; i < SIZE; i++)
    {
        cout << "The pay for employee "
             << id[i] << " is $" << fixed
             << showpoint << setprecision(2)
             << weeklyPay[i] << endl;
    }

46. const int ROWS = 30, COLS = 10;
    int grades[ROWS][COLS];

47. const int SIZE = 12;
    char countries[SIZE];   // To hold the country names
    long populations[SIZE]; // To hold populations
    // Display each country's name and population.
    for (int i = 0; i < 12; i++)
cout << "The population of " << countries[1] << " is " << populations[i] << endl;

48. No, the correct sum will not be printed for numberArray2 because the accumulator (total) is not set back to zero before the second loop executes.

49. const int NUM_NAMES = 3, SIZE = 10;
    char myNames[NUM_NAMES][SIZE] = {"Jason",
        "Lee",
        "Smith" };

50. numberArray[0][0] = 145;
    numberArray[8][10] = 18;

51. int row, col;     // Loop counters
    float total = 0.0;   // Accumulator
    // Sum the values in the array.
    for (row = 0; row < 10; row++)
    {
        for (col = 0; col < 20; col++)
            total += values[row][col];
    }

52. int row, col,    // Loop counters
    total;            // Accumulator
    // Display the sum of each row.
    for (row = 0; row < 29; row++)
    {
        // Set the accumulator.
        total = 0;
        // Sum a row.
        for (col = 0; col < 5; col++)
            total += days[row][col];
        // Display the row's total.
        cout << "The total for row " << row << " is " << total << endl;
    }
    // Display the sum of each column.
    for (col = 0; col < 29; col++)
    {
        // Set the accumulator.
        total = 0;
        // Sum a column.
        for (row = 0; row < 5; row++)
            total += days[row][col];
        // Display the column's total.
cout << "The total for column " 
<< col << " is " << total << endl;
}

53. false
54. true
55. true
56. false
57. true
58. false
59. false
60. true
61. false
62. true
63. true
64. false
65. false
66. false
67. true
68. false
69. true
70. false
71. true
72. true
73. true
74. true
75. true
76. false
77. false
78. true
79. false
80. true
81. true
82. false
83. true
84. The size declarator cannot be a variable.
85. The size declarator cannot be negative.
86. The loop will write data past the end of the array.
87. The initialization list must be enclosed in braces.
88. The string “George Washington” includes a null terminator, which makes it 18 characters in length.
89. Two of the initialization values are left out.
90. For the array to be implicitly sized there must be an initialization list.
91. A null terminator must be specified in the initialization list.
92. The assignment operator cannot be used to assign the contents of one array to another, in a single statement.
93. The parameter should be declared as `int nums[]`.
94. The parameter must specify the number of columns, not the number of rows.