Answers to Review Questions
Chapter 5

1. By indenting the statements, you make them stand out from the surrounding code. This helps you to identify at a glance the statements that are conditionally executed by a loop.

2. A pretest loop tests its condition before each iteration. A posttest loop tests its condition after each iteration. A posttest loop will always execute at least once.

3. Because they are only executed when a condition is true.

4. The `while` loop is a pretest loop and the `do-while` loop is a posttest loop.

5. The `while` loop.

6. The `do-while` loop.

7. The `for` loop.

8. A counter variable is used to control or keep track of the number of times a loop iterates. In a loop, the counter is usually incremented or decremented. If the counter variable is not properly initialized, it will not hold the correct number.

9. An accumulator is used to keep a running total of numbers. In a loop, a value is usually added to the current value of the accumulator. If it is not properly initialized, it will not contain the correct total.

10. Because the `for` loop has an update expression that normally changes the contents of a counter. If you have code inside that loop that also changes the counter, the loop will not be able to properly control the value of the counter.

11. increment, decrement
12. prefix
13. postfix
14. body
15. iteration
16. pretest
17. posttest
18. infinite or endless
19. counter
20. running total
21. accumulator
22. sentinel
23. `do-while`
24. `while` and `for`

25. `for`

26. initialization, test, update

27. nested

28. `break`

29. `continue`

30.
```
int product = 0, num;
while (product < 100)
{
     cin >> num;
     product = num * 10;
}
```

31.
```
do
{
     double num1, num2;
     char again;
     cout << "Enter two numbers: ";
     cin >> num1 >> num2;
     cout << "Their sum is " << (num1 + num2) << endl;
     cout << "Do you wish to do this again? (Y/N) ";
     cin >> again;
} while (again == 'Y' || again == 'y');
```

32.
```
for (int x = 0; x <= 1000; x += 10)
     cout << x;
```

33.
```
double total = 0.0, num;
for (int count = 0; count < 10; count++)
{
     cout << "Enter a number: ";
     cin >> num;
     total += num;
}
```

34.
```
for (int row = 0; row < 10; row++)
{
     for (int col = 0; col < 15; col++)
          cout << ‘#’;
     cout << endl;
}
```

35.
```
int x;
do
{
```

```
          cout << "Enter a number: ";
          cin >> x;
     } while (x > 0);
```

36.
```
char sure = ‘x’;
while (sure != ‘Y’ && sure != ‘N’)
{
     cout << “Are you sure you want quit?
     cin >> sure;
}
```

37.
```
for (int count = 0; count < 50; count++)
     cout << "count is " << count << endl;
```

38.
```
int x = 50;
while (x > 0)
{
     cout << x << “ seconds to go.\n”;
     x--;
}
```

39. false
40. true
41. false
42. true
43. false
44. false
45. false
46. true
47. false
48. true
49. true
50. false
51. true
52. false
53. false
54. false
55. true

56. The statement `result = ++(num1 + num2);` is invalid.
57. The `while` loop tests the variable again before any values are stored in it.
    The `while` loop is missing its opening and closing braces.
58. The `while` statement should not end with a semicolon.
    It could also be argued that `bigNum` should be defined a `long`.
    `count` should be initialized to 1.
59. The variable `total` is not initialized to 0.

The statement that calculates the average performs integer division. It should use a type cast to cast either `total` or `numCount` to a `double`.

The variable `count` is incremented in the `for` loop's update expression and again within the `for` loop.

60. The expression tested by the `do-while` loop should be `choice == 1` instead of `choice = 1`.

61. The variable `total` is not initialized to 0.

The `while` loop does not change the value of `count`, so it iterates an infinite number of times.