3.18 Write a single C++ statement that prints "too many" if the variable `count` exceeds 100, using
a. an `if` statement;
b. the conditional expression operator.

Problems

3.1 Modify the program in Example 3.1 on page 36 so that it prints a response only if `n` is divisible by `d`.

3.2 Modify the program in Example 3.5 on page 39 so that it prints the minimum of four input integers.

3.3 Modify the program in Example 3.5 on page 39 so that it prints the median of three input integers.

3.4 Modify the program in Example 3.6 on page 39 so that it has the same effect without using a statement block.

3.5 Predict the output from the program in Example 3.7 on page 40 after removing the declaration on the fifth line of the program. Then run that program to check your prediction.

3.6 Write and run a program that reads the user's age and then prints "You are a child." if the age < 18, "You are an adult." if 18 ≤ age < 65, and "You are a senior citizen." if age ≥ 65.

3.7 Write and run a program that reads two integers and then uses the conditional expression operator to print either "multiple" or "not" according to whether one of the integers is a multiple of the other.

3.8 Write and run a program that simulates a simple calculator. It reads two integers and a character. If the character is a `+`, the sum is printed; if it is a `-`, the difference is printed; if it is a `*`, the product is printed; if it is a `/`, the quotient is printed; and if it is a `%`, the remainder is printed. Use a `switch` statement.

3.9 Write and run a program that plays the game of "Rock, paper, scissors." In this game, two players simultaneously say (or display a hand symbol representing) either "rock," "paper," or "scissors." The winner is the one whose choice dominates the other. The rules are: paper dominates (wraps) rock, rock dominates (breaks) scissors, and scissors dominate (cut) paper. Use enumerated types for the choices and for the results.

3.10 Modify the solution to Problem 3.9 by using a `switch` statement.

3.11 Modify the solution to Problem 3.10 by using conditional expressions where appropriate.

3.12 Write and test a program that solves quadratic equations. A *quadratic equation* is an equation of the form $ax^2 + bx + c = 0$, where a, b, and c are given coefficients and x is the unknown. The coefficients are real number inputs, so they should be declared of type `float` or `double`. Since quadratic equations typically have two solutions, use `x1` and `x2` for the solutions to be output. These should be declared of type `double` to avoid inaccuracies from round-off error. (See Example 2.15 on page 28.)

3.13 Write and run a program that reads a six-digit integer and prints the sum of its six digits. Use the *quotient operator* `/` and the *remainder operator* `%` to extract the digits from the integer. For example, if `n` is the integer 876,543, then `n/1000%10` is its thousands digit 6.

3.14 Correct Example 3.17 on page 47 so that it produces the correct response for all inputs.

Answers to Review Questions

3.1
```
if (count > 100) cout << "Too many";
```

```
if (n > 2)
{ if (n < 6) cout << "OK";
}
else cout << "NG";
```

The braces are needed here to override the "`else` matching" rule. This `else` is intended to match the first `if`. The second statement should be written

```
if (n > 2)
  if (n < 6) cout << "OK";
  else cout << "NG";
```

Here the braces are not needed because the `else` is intended to be matched with the second `if`.

3.15 A "fall through" in a `switch` statement is a case that does not include a `break` statement, thereby causing control to continue right on to the next case statement.

3.16 This expression evaluates to –1 if `x < y`, it evaluates to 0 if `x == y`, and it evaluates to 1 if `x > y`.

3.17 `absx = ( x>0 ? x : -x );`

3.18 *a.* `if (count > 100) cout << "too many";`

b. `cout << (count > 100 ? "too many" : "" );`

Solutions to Problems

3.1 This version of Example 3.1 on page 36 prints a response only when n is divisible by d:

```
int main()
{ int n, d;
  cout << "Enter two positive integers: ";
  cin >> n >> d;
  if (n%d == 0) cout << n << " is divisible by " << d << endl;
}
```

```
Enter two positive integers: 56 7
56 is divisible by 7
```

3.2 This version of Example 3.5 on page 39 prints the minimum of four input integers:

```
int main()
{ int n1, n2, n3, n4;
  cout << "Enter four integers: ";
  cin >> n1 >> n2 >> n3 >> n4;
  int min=n1;                 // now min <= n1
  if (n2 < min) min = n2;     // now min <= n1, n2
  if (n3 < min) min = n3;     // now min <= n1, n2, n3
  if (n4 < min) min = n4;     // now min <= n1, n2, n3, n4
  cout << "Their minimum is " << min << endl;
}
```

```
Enter four integers: 44 88 22 66
Their minimum is 22
```

3.3 This program finds the median of three input integers:

```
int main()
{ int n1, n2, n3;
  cout << "Enter three integers: ";
  cin >> n1 >> n2 >> n3;
  cout << "Their median is ";
  if (n1 < n2)
    if (n2 < n3) cout << n2;        // n1 <  n2 <  n3
```

```
    else if (n1 < n3) cout << n3;   // n1 <  n3 <= n2
    else cout << n1;                // n3 <= n1 <  n2
  else if (n1 < n3) cout << n1;     // n2 <= n1 <  n3
  else if (n2 < n3) cout << n2;     // n2 <  n3 <= n1
  else cout << n3;                  // n3 <= n2 <= n1
}
```

```
Enter three integers: 44 88 22
Their median is 44
```

3.4 This program has the same effect as the one in Example 3.6 on page 39:

```
int main()
{ int x, y;
  cout << "Enter two integers: ";
  cin >> x >> y;
  if (x > y) cout << y << " <= " << x << endl;
  else cout << x << " <= " << y << endl;
}
```

```
Enter two integers: 66 44
44 <= 6
```

3.5 Modification of the program in Example 3.7 on page 40:

```
int main()
{ int n=44;
  cout << "n = " << n << endl;
  { cout << "Enter an integer: ";
    cin >> n;
    cout << "n = " << n << endl;
  }
  { cout << "n = " << n << endl;
  }
  { int n;
    cout << "n = " << n << endl;
  }
  cout << "n = " << n << endl;
}
```

```
n = 44
Enter an integer: 77
n = 77
n = 77
n = 4251897
n = 77
```

3.6 Here we used the `else if` construct because the three outcomes depend upon `age` being in one of three disjoint intervals:

```
int main()
{ int age;
  cout << "Enter your age: ";
  cin >> age;
  if (age < 18) cout << "You are a child.\n";
  else if (age < 65) cout << "You are an adult.\n";
  else cout << "you are a senior citizen.\n";
}
```

```
Enter your age: 44
You are an adult.
```

If control reaches the second condition `(age < 65)`, then the first condition must be false so in fact $18 \le age < 65$. Similarly, if control reaches the second `else`, then both conditions must be false so in fact $age \ge 65$.

3.7 An integer `m` is a multiple of an integer `n` if the remainder from the integer division of `m` by `n` is 0. So the compound condition `m % n == 0 || n % m == 0` tests whether either is a multiple of the other:

```
int main()
{ int m, n;
  cin >> m >> n;
  cout << (m % n == 0 || n % m == 0 ? "multiple" : "not") << endl;
}
```

```
30 4
not
```

```
30 5
multiple
```

The value of the conditional expression will be either `"multiple"` or `"not"`, according to whether the compound condition is true. So sending the complete conditional expression to the output stream produces the desired result.

3.8 The character representing the operation should be the control variable for the `switch` statement:

```
int main()
{ int x, y;
  char op;
  cout << "Enter two integers: ";
  cin >> x >> y;
  cout << "Enter an operator: ";
  cin >> op;
  switch (op)
  { case '+': cout << x + y << endl;  break;
    case '-': cout << x - y << endl;  break;
    case '*': cout << x * y << endl;  break;
    case '/': cout << x / y << endl;  break;
    case '%': cout << x % y << endl;  break;
  }
}
```

```
Enter two integers: 30 13
Enter an operator: %
4
```

In each of the five cases, we simply print the value of the corresponding arithmetic operation and then break.

3.9 First define the two `enum` types `Choice` and `Result`. Then declare variables `choice1`, `choice2`, and `result` of these types, and use an integer `n` to get the required input and assign it to them:

```
enum Choice {ROCK, PAPER, SCISSORS};
enum Winner {PLAYER1, PLAYER2, TIE};
int main()
{ int n;
  Choice choice1, choice2;
  Winner winner;
  cout << "Choose rock (0), paper (1), or scissors (2):" << endl;
  cout << "Player #1: ";
  cin >> n;
  choice1 = Choice(n);
```

```
  cout << "Player #2: ";
  cin >> n;
  choice2 = Choice(n);
  if (choice1 == choice2) winner = TIE;
  else if (choice1 == ROCK)
    if (choice2 == PAPER) winner = PLAYER2;
    else winner = PLAYER1;
  else if (choice1 == PAPER)
    if (choice2 == SCISSORS) winner = PLAYER2;
    else winner = PLAYER1;
  else // (choice1 == SCISSORS)
    if (choice2 == ROCK) winner = PLAYER2;
    else winner = PLAYER1;
  if (winner == TIE) cout << "\tYou tied.\n";
  else if (winner == PLAYER1) cout << "\tPlayer #1 wins." <<endl;
  else cout << "\tPlayer #2 wins." << endl;
}
```

```
Choose rock (0), paper (1), or scissors (2):
Player #1: 1
Player #2: 1
    You tied.
```

```
Choose rock (0), paper (1), or scissors (2):
Player #1: 2
Player #2: 1
    Player #1 wins.
```

```
Choose rock (0), paper (1), or scissors (2):
Player #1: 2
Player #2: 0
        Player #2 wins.
```

Through a series of nested `if` statements, we are able to cover all the possibilities.

3.10 Using a `switch` statement:

```
enum Winner {PLAYER1, PLAYER2, TIE};
int main()
{ int choice1, choice2;
  Winner winner;
  cout << "Choose rock (0), paper (1), or scissors (2):" << endl;
  cout << "Player #1: ";
  cin >> choice1;
  cout << "Player #2: ";
  cin >> choice2;
  switch (choice2 - choice1)
  { case  0:
      winner = TIE;
      break;
    case -1:
    case  2:
      winner = PLAYER1;
      break;
    case -2:
    case  1:
      winner = PLAYER2;
  }
```

```
  if (winner == TIE) cout << "\tYou tied.\n";
  else if (winner == PLAYER1) cout << "\tPlayer #1 wins." << endl;
  else cout << "\tPlayer #2 wins." << endl;
}
```

3.11 Using a `switch` statement and conditional expressions:

```
enum Winner {PLAYER1, PLAYER2, TIE};
int main()
{ int choice1, choice2;
  cout << "Choose rock (0), paper (1), or scissors (2):" << endl;
  cout << "Player #1: ";
  cin >> choice1;
  cout << "Player #2: ";
  cin >> choice2;
  int n = (choice1 - choice2 + 3) % 3;
  Winner winner = ( n==0 ? TIE : (n==1?PLAYER1:PLAYER2) );
  if (winner == TIE) cout << "\tYou tied.\n";
  else if (winner == PLAYER1) cout << "\tPlayer #1 wins." << endl;
  else cout << "\tPlayer #2 wins." << endl;
}
```

3.12 The solution(s) to the quadratic equation is given by the *quadratic formula*:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

But this will not apply if a is zero, so that condition must be checked separately. The formula also fails to work (for real numbers) if the expression under the square root is negative. That expression $b^2 + 4ac$ is called the *discriminant* of the quadratic. We define that as the separate variable d and check its sign.

```
#include <iostream>
#include <cmath>  // defines the sqrt() function
int main()
{ // solves the equation a*x*x + b*x + c == 0:
  float a, b, c;
  cout << "Enter coefficients of quadratic equation: ";
  cin >> a >> b >> c;
  if (a == 0)
  { cout << "This is not a quadratic equation: a == 0\n";
    return 0;
  }
  cout << "The equation is: " << a << "x^2 + " << b
       << "x + " << c << " = 0\n";
  double d, x1, x2;
  d = b*b - 4*a*c;  // the discriminant
  if (d < 0)
  { cout << "This equation has no real solutions: d < 0\n";
    return 0;
  }
  x1 = (-b + sqrt(d))/(2*a);
  x2 = (-b - sqrt(d))/(2*a);
  cout << "The solutions are: " << x1 << ", " << x2 << endl;
}
```

```
Enter coefficients of quadratic equation: 2 1 -6
The equation is: 2x^2 + 1x + -6 = 0
The solutions are: 1.5, -2
```

```
Enter coefficients of quadratic equation: 1 4 5
The equation is: 1x^2 + 4x + 5 = 0
This equation has no real solutions: d < 0
```

```
Enter coefficients of quadratic equation: 0 4 5
This is not a quadratic equation: a == 0
```

Note how we use the `return` statement inside the selection statements to terminate the program if either `a` is zero or `d` is negative. The alternative would have been to use an `else` clause in each `if` statement.

3.13 This program prints the sum of the digits of the given integer:

```
int main()
{ int n, sum;
  cout << "Enter a six-digit integer: ";
  cin >> n;
  sum = n%10 + n/10%10 + n/100%10 + n/1000%10 + n/10000%10
        + n/100000;
  cout << "The sum of the digits of " << n << " is " << sum <<endl;
}
```

```
Enter a six-digit integer: 876543
The sum of the digits of 876543 is 33
```

3.14 A corrected version of Example 3.17 on page 47:

```
int main()
{ // reports the user's grade for a given test score:
  int score;
  cout << "Enter your test score: ";
  cin >> score;
  if (score > 100 || score < 0)
    cout << "Error: that score is out of range.\n";
  else
    switch (score/10)
    { case 10:
      case  9: cout << "Your grade is an A.\n";  break;
      case  8: cout << "Your grade is a B.\n";   break;
      case  7: cout << "Your grade is a C.\n";   break;
      case  6: cout << "Your grade is a D.\n";   break;
      default: cout << "Your grade is an F.\n";  break;
    }
  cout << "Goodbye." << endl;
}
```

```
Enter your test score: 103
Error: that score is out of range.
Goodbye.
```

```
Enter your test score: 93
Your grade is an A.
Goodbye.
```

```
Enter your test score: -3
Error: that score is out of range.
Goodbye.
```