```
    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 ≤ age < 65. Similarly, if control reaches the second `else`, then both conditions must be false so in fact age ≥ 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
```