1. Review: **Arithmetic Expression** represents a single value

   **3 Forms:**
   a. constant
   b. variable
   c. combination of variables and constants joined by arithmetic operators

2. **Operator** specifies an operation to be carried out - an Action

   **Operand** is acted upon

   Example: \(2 + 3\) \(\ +\) is the operator; \(2\) and \(3\) are the operands.

3. \(\text{double fahrenheit, centigrade} = 1.;\)

   \(\text{fahrenheit} = 1.8 * \text{centigrade} + 32.0;\)

   **Multiplication First:**
   \[
   (1.8 \times 1.) + 32. = 1.8 + 32. = 33.8
   \]

   **Addition First:**
   \[
   1.8 \times (1. + 32.) = 1.8 \times 33. = 59.4
   \]

   **Precedence levels:**
   \[
   \begin{array}{cccc}
   + & - & \text{unary} \\
   * & / & \% \\
   + & - & \text{binary}
   \end{array}
   \]

   **Other examples:**

   \[
   \begin{array}{ll}
   \text{C++} & \text{Math. Notation} \\
   a. & \text{x} = \text{a} + \text{b} \times \text{c}; \quad \text{x} = \text{a} + \text{b.c} \\
   b. & \text{x} = \text{a} / \text{b} + \text{c}; \quad \text{x} = \text{a} + \text{c} / \text{b} \\
   c. & \text{x} = \text{a} - \text{b} / \text{c}; \quad \text{x} = \text{a} - \frac{\text{b}}{\text{c}} \quad \text{Binary minus}
   \quad \text{operates on 2 operands} \\
   d. & \text{x} = -\text{a} \times \text{b}; \quad \text{x} = (-\text{a}) \cdot \text{b} \quad \text{Unary minus}
   \quad \text{operates on 1 operand}
   \end{array}
   \]
4. Operators with same precedence use left to right associativity:

   \[
   \begin{align*}
   &\text{C++} & &\text{Math. Notation} \\
   a. & x = a / b \times c; & x = a / b \times c; \\
   b. & x = a \times b / c; & x = a \times b / c; \\
   c. & x = a \times b \times c; & x = a \times b \times c; \\
   d. & x = a + b + c + d; & x = a + b + c + d; \\
   e. & x = a / b \times c / d; & x = a / b \times c / d; \\
   \end{align*}
   \]

5. Associativity apply only when two operators share the same operand.

   \[
   X = 10 \times 2 + 15 / 3;
   \]

   * and / are done before the addition.

   Whether * or / is done first is left up to the implementors. They can decide which order is better for the particular system.
6. \[ \text{centigrade} = \frac{\text{fahrenheit} - 32}{1.8} \]

```cpp
double centigrade, fahrenheit = 33.8;
centigrade = fahrenheit - 32.0 / 1.8; // Precedence does not work
gives centigrade = fahrenheit - 32
1.8
centigrade = (fahrenheit - 32.0) / 1.8;
```

Use parentheses to override the normal precedence.

7. a. \[ x = \frac{a + b}{x - y}; \]

b. \[ x = \frac{(a + b)}{x - y}; \]

c. \[ x = \frac{a + b}{(x - y)}; \]

d. \[ x = \frac{(a + b)}{(x - y)}; \]

8. \[ x = \frac{a}{1 + b / (2 + c)}; \]
1. **Integer Expression**: expression consisting of only integer operands

   Integer expression is evaluated using **Integer Arithmetic** which always yields an **Integer result**.

   **Integer Division**: fractional digits of quotient are truncated; no rounding

   ```
   int a, b;
   a = 7 / 4; // a is set to 1 of 1.75
   b = 1 / 2; // b is set to 0 of 0.5
   ```

2. **Floating-Point Expression**: expression consisting of only floating-point operands

   Floating-point expression is evaluated using **Floating-point Arithmetic** which always yields a **Floating-point result**.

   Fractional digits are kept.

   ```
   double x, y;
   x = 7. / 4.; // x is set to 1.75
   y = 1. / 2.; // y is set to 0.5
   ```
3. **Mixed-Expression**: expression consisting of operands of different types

Ranking of data types:  
- long double  
  (higher rank)  
- double  
- float  
- int  
- short  
  (lower rank)

Operand of lower rank is automatically converted to higher rank.

```c
double x, y;
x = 2.5 + 1 / 2;  
versus  
y = 2.5 + 1.0 / 2;
```

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5 + 0</td>
<td>2.5 + 0.0</td>
</tr>
<tr>
<td>2.5 + 0.0</td>
<td>2.5 + 0.5</td>
</tr>
<tr>
<td>2.5</td>
<td>3.0</td>
</tr>
</tbody>
</table>

**Promotion**: conversion from lower rank to higher rank

**Demotion**: conversion from higher rank to lower rank

4.  
```
10 int a, b;
20 double x, y;
30 b = 2;
40 x = b + 3;  // int expression; int result automatically promoted to double; x is set to 5.0, not 5
50 y = .03;
60 a = 3.75 + y; // double expression; double result automatically demoted to int; a is set to 3, not 3.78
```

5. **Type casts**

Syntax: `type (operand to be converted)`

<table>
<thead>
<tr>
<th>Example</th>
<th>Type Conversion</th>
</tr>
</thead>
</table>
| a. `x = 2.5 + 1 / 2;` | b. `y = 2.5 + 1.0 / 2;`  
  `x = 2.5 + double (1/2);`  
|                   |                 |
| c. `x = b + 3;`    | d. `a = 3.75 + y;`  
  `x = double (b + 3);`  
|                   | `a = int (3.75 + y);` |
6. **True Quotient**

```plaintext
90    int a;
100   double x;
110   a = 7 / 4;    // a is set to 1, not 1.75
120   a = double (7) / 4; // a is set to 1, not 1.75
130   x = double (7) / 4; // x is set to 1.75
```

To store the true quotient:
- Force floating-point arithmetic and
- Store quotient in floating-point variable
1. C has several assignment operators - Shorthand Notation

2. int a, b, c, d, e, f, x, y;

   a += 100; a = a + 100;
   b -= 23; b = b - 23;
   c *= 3; c = c * 3;
   d /= 5; d = d / 5;
   e %= 2; e = e % 2;

3. var op= exp       var = var op (exp)
   f *= 5 + x - y;     f = f * (5 + x - y);

4. Both forms work, but

   assignment operators are more compact;
   more commonly used, and
   may produce more efficient machine language code.
Code an arithmetic assignment statement for each of the following formulas. Do not use redundant parentheses. Assume the following declaration statement:

```c
int a, b, c, d, x, y;
```

1. \[ x = \frac{a}{b + c} \]
2. \[ y = (a \cdot c + b) \cdot c \]
3. \[ x = \frac{a \cdot (b + c)}{d} \]
4. \[ y = \frac{a \cdot b}{c \cdot d} \]
5. \[ x = \frac{a}{b} \cdot \frac{c}{d} \]
6. \[ y = (a + b) \cdot (c + d) \]
7. \[ x = a + \frac{b}{c} \]
8. \[ y = \frac{a + b}{c} \]
9. \[ x = \frac{a}{5 + \frac{b}{c + d}} \]
10. \[ y = -(a \cdot b) \]
Code an arithmetic assignment statement for each of the following formulas. Do not use redundant parentheses. Assume all variables are of type int.

1. \[ x = \frac{a}{b + c} \]
   
   \[ x = a / (b + c); \]

2. \[ y = (a\cdot c + b)\cdot c \]
   
   \[ y = (a * c + b) * c; \]

3. \[ x = \frac{a \cdot (b + c)}{d} \]
   
   \[ x = a * (b + c) / d; \]

4. \[ y = \frac{a \cdot b}{c \cdot d} \]
   
   \[ y = a * b / (c * d); \]

5. \[ x = \frac{a}{b} \frac{c}{d} \]
   
   \[ x = a / b / (c * d); \]

6. \[ y = (a + b) \cdot (c + d) \]
   
   \[ y = (a + b) * (c + d); \]

7. \[ x = a + \frac{b}{c} \]
   
   \[ x = a + b / c; \]

8. \[ y = \frac{a + b}{c} \]
   
   \[ y = (a + b) / c; \]

9. \[ x = \frac{a}{5 + \frac{b}{c + d}} \]
   
   \[ x = a / (5 + b / (c + d)); \]

10. \[ y = -(a \cdot b) \]
    
    \[ y = - (a * b); \]
Give the current values of the four indicated variables after each of the following assignment statements has been executed. Assume the following declaration statements:

```plaintext
int a, b;
double x, y;
```

<table>
<thead>
<tr>
<th></th>
<th>Statement</th>
<th>a</th>
<th>b</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>x = 10.0 + 5.0;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><code>a = 2 * 3;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>y = x + 1.0 * 2.0;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><code>b = 3;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>a += 1;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>b = (a + b + 4) / 5;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><code>x = y - x;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><code>b = b % 3 + a;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><code>a = b * b / 2;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><code>x = 2 + a - b;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td><code>a = 10.5 + x;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td><code>a = x / (y - 15.0);</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td><code>y = (a - 1) * (b - 6) / b;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
14  b++;
15  b = -(b + a / b - a);
16  a = a + b - 2 * a / b;
17  a = (b / (3 * b + 2) - 1);
18  x = y * y + x / 5.0;
19  y = (x - 20) * 2 - ((y + 1) / 2);
20  y = 19 - (15 + a) / 4.0;
21  x = 19 - (15 + a) / 4;
22  a = 8.50 / 2 * 5;
23  b = 8.50 / (2 * 5);
24  a %= 8;
25  a /= 2;
26  b = x - 3 * a;
27  x = double (b) / 4;
28  y = double (b / 4);
Give the current values of the four indicated variables after each of the following assignment statements has been executed. Assume the following declaration statements:

```c
int a, b;
double x, y;
```

<table>
<thead>
<tr>
<th>Assignment</th>
<th>a</th>
<th>b</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( x = 10.0 + 5.0; )</td>
<td>?</td>
<td>?</td>
<td>15.0</td>
<td>?</td>
</tr>
<tr>
<td>2 ( a = 2 * 3; )</td>
<td>6</td>
<td>?</td>
<td>15.0</td>
<td>?</td>
</tr>
<tr>
<td>3 ( y = x + 1.0 * 2.0; )</td>
<td>6</td>
<td>?</td>
<td>15.0</td>
<td>17.0</td>
</tr>
<tr>
<td>4 ( b = 3; )</td>
<td>6</td>
<td>3</td>
<td>15.0</td>
<td>17.0</td>
</tr>
<tr>
<td>5 ( a + 1; )</td>
<td>7</td>
<td>3</td>
<td>15.0</td>
<td>17.0</td>
</tr>
<tr>
<td>6 ( b = (a * b + 4) / 5; )</td>
<td>7</td>
<td>5</td>
<td>15.0</td>
<td>17.0</td>
</tr>
<tr>
<td>7 ( x = y - x; )</td>
<td>7</td>
<td>5</td>
<td>2.0</td>
<td>17.0</td>
</tr>
<tr>
<td>8 ( b = b % 3 + a; )</td>
<td>7</td>
<td>9</td>
<td>2.0</td>
<td>17.0</td>
</tr>
<tr>
<td>9 ( a = b * b / 2; )</td>
<td>40</td>
<td>9</td>
<td>2.0</td>
<td>17.0</td>
</tr>
<tr>
<td>10 ( x = 2 + a - b; )</td>
<td>40</td>
<td>9</td>
<td>33.0</td>
<td>17.0</td>
</tr>
<tr>
<td>11 ( a = 10.5 + x; )</td>
<td>43</td>
<td>9</td>
<td>33.0</td>
<td>17.0</td>
</tr>
<tr>
<td>12 ( a = x / (y - 15.0); )</td>
<td>16</td>
<td>9</td>
<td>33.0</td>
<td>17.0</td>
</tr>
<tr>
<td>13 ( y = (a - 1) * (b - 6) / b; )</td>
<td>16</td>
<td>9</td>
<td>33.0</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>b++;</td>
<td>16</td>
<td>10</td>
<td>33.0</td>
</tr>
<tr>
<td>15</td>
<td>b = -(b + a / b - a);</td>
<td>16</td>
<td>5</td>
<td>33.0</td>
</tr>
<tr>
<td>16</td>
<td>a = a + b - 2 * a / b;</td>
<td>15</td>
<td>5</td>
<td>33.0</td>
</tr>
<tr>
<td>17</td>
<td>a = (b / (3 * b + 2) - 1);</td>
<td>-1</td>
<td>5</td>
<td>33.0</td>
</tr>
<tr>
<td>18</td>
<td>x = y * y + x / 5.0;</td>
<td>-1</td>
<td>5</td>
<td>31.6</td>
</tr>
<tr>
<td>19</td>
<td>y = (x - 20) * 2 - ((y + 1) / 2);</td>
<td>-1</td>
<td>5</td>
<td>31.6</td>
</tr>
<tr>
<td>20</td>
<td>y = 19 - (15 + a) / 4.0;</td>
<td>-1</td>
<td>5</td>
<td>31.6</td>
</tr>
<tr>
<td>21</td>
<td>x = 19 - (15 + a) / 4;</td>
<td>-1</td>
<td>5</td>
<td>16.0</td>
</tr>
<tr>
<td>22</td>
<td>a = 8.50 / 2 * 5;</td>
<td>21</td>
<td>5</td>
<td>16.0</td>
</tr>
<tr>
<td>23</td>
<td>b = 8.50 / (2 * 5);</td>
<td>21</td>
<td>0</td>
<td>16.0</td>
</tr>
<tr>
<td>24</td>
<td>a %= 8;</td>
<td>5</td>
<td>0</td>
<td>16.0</td>
</tr>
<tr>
<td>25</td>
<td>a / = 2;</td>
<td>2</td>
<td>0</td>
<td>16.0</td>
</tr>
<tr>
<td>26</td>
<td>b = x - 3 * a;</td>
<td>2</td>
<td>10</td>
<td>16.0</td>
</tr>
<tr>
<td>27</td>
<td>x = double (b) / 4;</td>
<td>2</td>
<td>10</td>
<td>2.5</td>
</tr>
<tr>
<td>28</td>
<td>y = double (b / 4);</td>
<td>2</td>
<td>10</td>
<td>2.5</td>
</tr>
</tbody>
</table>