1. C++ data type review: indicate if each the following matched with the correct type:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Type</th>
<th>Right/Wrong (correction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>‘a’</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>char</td>
<td></td>
</tr>
<tr>
<td>“5.0”</td>
<td>char</td>
<td></td>
</tr>
</tbody>
</table>

2. Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Relational operators: to perform comparison of variables, constants, or expressions in C/C++

<table>
<thead>
<tr>
<th>Operators(s)</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td></td>
<td></td>
</tr>
<tr>
<td>!=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Conditional Statements: if/else

What will each implementation print if ‘grade’ stores 95?

Implementation 1:
```cpp
if (grade >= 90) {
    cout << "A range" << endl;
}
else if (grade >= 80) {
    cout << "B range" << endl;
}
else if (grade >= 70) {
    cout << "C range" << endl;
}
else {
    cout << "Below C range!" << endl;
}
```

Implementation 2:
```cpp
if (grade >= 90) {
    cout << "A range" << endl;
}
else if (grade >= 80) {
    cout << "B range" << endl;
}
else if (grade >= 70) {
    cout << "C range" << endl;
}
else {
    cout << "Below C range!" << endl;
}
```
What did you notice about if and else?

if:

else:

5. Logical Operators: to create compound conditions

<table>
<thead>
<tr>
<th>Operators(s)</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND</td>
<td>x &amp;&amp; y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>Logical NOT</td>
<td>!x</td>
</tr>
</tbody>
</table>

Quick check: Which of the following is NOT a condition to check if the integer x is in the range [-1 to 5]?

A. x >= -1 && x <= 5
B. -1 <= x <= 5
C. !( x < -1 || x > 5)
D. x > -2 && x < 6

6. Common mistakes

a. Using assignment operator (=) rather than equality check operator (==)
   Correct the following code:
   ```cpp
   int x;
   cin >> x;
   if (x = 0)
     cout << "x is 0" << endl;
   ```
   Tip: When comparing with a constant, many companies recommend flipping the order to:
   ```cpp
   if (0 == x) { /*some code*/ }
   ```
   This way, the code won't compile if you accidentally write:
   ```cpp
   if (0 = x) { /*some code*/ }
   ```

b. Using multiple if statements rather than if ... else
   Correct the following code:
   ```cpp
   int x, y;
   cin >> x >> y; //takes two inputs, and store them into x and y, respectively
   if (x != y)
     x = 5;
   if (x == y)
     y = 7;
   ```

c. Wrong formulated conditions.
   Correct the following code:
   ```cpp
   if (0 <= x <= 9) { /*some code*/ }
   ```
   ```cpp
   if (x == 0 || 1) { /*some code*/ }
   ```
7. Loops
   a. for loop: used when you DO know the number of times to iterate BEFORE the loop starts
      Ex: print out all multiples of 7 from 0 to 100, inclusive

   b. while loop: used when you DON’T know how many times to iterate before the loop starts
      Ex: let user guess my secret number until they are correct
          int guess;
          int secret_num = /* some code */;
          cin >> guess;
          // complete the rest….

   Tip: Use while loop whenever you see/use “until”, until x == while not x
   For example: keep guessing until correct == keep guessing while not correct

   c. do-while loop: often used to run/play again. Loop body is executed at least once
      Ex: ask the user whether they want to run the program again, 1=yes, 0-no

   d. nested loop: The inner loop executes completely for each single iteration of the outer loop
      Ex: Trace through the execution of the following code and show what will be printed.

         for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 3; j++) {
               cout << i << “ “ << j << endl;
            }  
         }