1. An expression using the greater-than, less-than, greater-than-or-equal-to, less-than-or-equal-to, or not-equal operator is called a(n) \_**relational**\_ expression.
2. The value of a relational expression is 0 if the expression is \_\_**false**\_\_ or 1 if the expression is \_**true**\_.
3. The if statement regards an expression with the value 0 as \_\_**false**\_\_ and an expression with a nonzero value as \_\_**true**\_\_\_.
4. For an if statement to conditionally execute a group of statements, the statements must be enclosed in a set of \_\_\_\_**{}**\_\_\_\_\_.
5. In an if/else statement, the if part executes its statement(s) if the expression is \_\_\_**true**\_\_, and the else part executes its statement(s) if the expression is \_\_\_**false**\_\_.
6. The trailing else in an if/else if statement has a similar purpose as the \_\_**default**\_\_ section of a switch statement.
7. If the sub-expression on the left of the && logical operator is \_**false**\_, the right sub-expression is not checked.
8. If the sub-expression on the left of the || logical operator is \_**true**\_, the right sub-expression is not checked.
9. The \_\_\_**!**\_\_\_\_ logical operator has higher precedence than the other logical operators.
10. Logical operators have \_\_**lower**\_\_ precedence than relational operators.
11. The \_\_\_**&&**\_\_ logical operator works best when testing a number to determine if it is within a range.
12. The \_\_\_\_**||**\_\_\_\_\_ logical operator works best when testing a number to determine if it is outside a range.
13. A variable with \_\_\_\_\_**block (or local)**\_\_\_\_ scope is only visible when the program is executing in the block containing the variable’s definition.
14. The expression that is tested by a switch statement must have a(n) \_\_\_\_**integer**\_\_\_\_ value.
15. A program will “fall through” to the following case section if it is missing the \_\_\_\_\_**break**\_\_\_\_ statement.
16. What value will be stored in the variable t after each of the following statements execute?
    1. t = (12 > 1); \_\_\_\_**true; 1**\_\_\_
    2. t = (2 < 0); \_\_**false; 0**\_\_\_
    3. t = (5 == (3 \* 2)); \_\_\_**false; 0**\_\_\_
    4. t = (5 == 5); \_\_\_**true; 1**\_\_\_\_
17. write an if statement that assigns 100 to x when y is equal to 0.

**if (y == 0)**

**x = 100;**

1. write an if/else statement that assigns 0 to x when y is equal to 10. Otherwise it should assign 1 to x.

**if (y == 10)**

**{**

**X = 0;**

**}**

**Else**

**{**

**x = 1;**

**}**

1. write an if/else statement that prints “Excellent” when score is 90 or higher, “Good” when score is between 80 and 89, and “Try Harder” when score is less than 80.

**If (score >= 90)**  
 **cout << “Excellent”;**

**Else if (sore >= 80)**

**Cout << “Good”;**

**Else**

**Cout << “Try Harder”;**

1. write an if statement that sets the variable hours to 10 when the flag variable minimum is set to true.

**If (flag == 0)**

1. convert the following conditional expression into an if/else statement.  
   q = (x < y) ? (a + b) : (x \* 2);

**if (x < y)**

**q = a + b;**

**else**

**q = x \* 2;**

1. convert the following if/else statement into a switch statement:  
   if (choice == 1)  
   {  
   cout << fixed << showpoint << setprecision (2);  
   }  
   else if ((choice == 2) || (choice == 3))  
   {  
   cout << fixed << showpoint << setprecision (4);  
   }  
   else if (choice == 4)  
   {  
   cout << fixed << showpoint << setprecision (6);  
   }  
   else  
   {  
   cout << fixed << showpoint << setprecision (8);  
   }

**switch (choice)**

**{**

**case 1:**

**cout << fixed << showpoint << setprecision (2);**

**break;**

**case 2:**

**case 3:**

**cout << fixed << showpoint << setprecision(4);**

**break;**

**case 4:**

**cout << fixed << showpoint << setprecision(6);**

**break;**

**default:**

**cout << fixed << showpoint << setprecision(8);**

**}**

1. Assume the variables x = 5, y = 6, and z = 8. Indicate if each of the following conditions is true or false:
   1. (x == 5) || (y > 3) **= T**
   2. (7 <= x) && (z > 4) = **F**
   3. (2 != y) && (z != 4) = **T**
2. Assume the variables x = 5, y = 6, and z = 8. Indicate if each of the following conditions is true or false.
   1. (x >= 0) || (x <= y) =  **T**
   2. (z – y) > y = **F**
   3. ! ((z – y) > x) = **T**

**1-20 FILL IN THE BLANKS**

1. To \_\_\_**++1**\_\_\_\_\_ a value means to increase it by one.
2. To \_\_\_\_**- -1**\_\_\_\_\_ a value means to decrease it by one.
3. When the increment or decrement operator is placed before the operand (or to the operand’s left), the operator is being used in \_\_\_\_**prefix**\_\_\_\_ mode.
4. When the increment or decrement operator is placed after the operand (or to the operand’s right), the operator is being used in \_\_\_\_**postfix**\_\_\_\_ mode.
5. The statement or block that is repeated is known as the \_\_\_\_**body**\_\_\_\_ of the loop.
6. Each repetition of a loop is known as a(n) \_**iteration**\_\_\_\_\_.
7. A loop that evaluates its test expression before each repetition is a(n) \_\_\_**pretest**\_\_\_ loop.
8. A loop that evaluates its test expression after each repetition is a(n) \_\_\_**posttest**\_\_\_\_ loop.
9. A loop that does not have a way of stopping is a(n) \_\_**infinite or endless**\_\_\_\_\_ loop.
10. A(n) \_**counter**\_\_\_ is a variable that “counts” the number of times a loop repeats.
11. A(n) \_\_**running total**\_\_\_ is a sum of numbers that accumulates with each iteration of a loop.
12. A(n) \_**accumulator**\_\_ is a variable that is initialized to some starting value, usually zero, and then has numbers added to it in each iteration of a loop.
13. A(n) \_**sentinel**\_\_\_ is a special value that marks the end of a series of values.
14. The \_\_\_**for**\_\_\_ loop is ideal for situations that require a counter.
15. The \_\_**do while**\_\_\_ loop always iterates at least once.
16. The \_\_\_**while**\_\_\_ and \_\_**for**\_\_\_ loops will not iterate at all if their test expressions are false to start with.
17. Inside the for loop’s parentheses, the first expression is the \_**initialization**\_\_ , the second expression is the \_\_**test**\_\_ , and the third expression is the \_**update**\_\_\_ .
18. A loop that is inside another is called a(n) \_\_\_**nested**\_\_\_\_\_\_ loop.
19. The \_**break**\_\_ statement causes a loop to terminate immediately.
20. The \_\_**continues**\_\_\_ statement causes a loop to skip the remaining statements in the current iteration.

Programming challenges will be in the main.cpp file.