**Operators Associativity and Precedence Assignment**

1. Use operator associativity, evaluate the following expressions and predict the output

**a. x = 34 + 12/4 – 56**

* First, we evaluate the division due to its higher precedence.
  + 12 / 4 = 3
* Now, we substitute the result into the expression:
  + x = 34 + 3 - 56
* Next, we perform addition and subtraction from left to right (since both have the same precedence and are evaluated left to right):
  + 34 + 3 = 37
  + 37 - 56 = -19

x = -19

**b. 12 + 3 - 4 / 2 < 3 + 1**

* First, we evaluate the division:
  + 4 / 2 = 2
* Now, we substitute the result into the expression:
  + 12 + 3 - 2 < 3 + 1
* Perform addition and subtraction from left to right:
  + 12 + 3 = 15
  + 15 - 2 = 13
  + 3 + 1 = 4
* Now we compare the two sides:
  + 13 < 4 is false (which is equivalent to 0 in C)  
    The expression evaluates to false (or 0).

**c. (2 + (3 + 2) ) \* 10**

* First, we evaluate the innermost parentheses:
  + 3 + 2 = 5
* Now, the expression becomes:
  + (2 + 5) \* 10
* Next, we evaluate 2 + 5 = 7, then:
  + 7 \* 10 = 70  
    The result is 70.

**d. 34 + 12/4 – 45**

* First, we evaluate the division:
  + 12 / 4 = 3
* Now, substitute the result into the expression:
  + 34 + 3 - 45
* Perform addition and subtraction from left to right:
  + 34 + 3 = 37
  + 37 - 45 = -8  
    The result is -8.

2. Rewrite the following expressions with improved readability

**a. age < 18 && height < 48 || age > 60 && height > 72**

Ans)To clarify the precedence and improve readability, use parentheses to group the logical conditions:

(age < 18 && height < 48) || (age > 60 && height > 72)

This explicitly shows the logical grouping and avoids ambiguity.

**b. char name value**

Ans) This seems like an incomplete or incorrect declaration. It should be written as:

char name; // Declare a character variable named 'name'

**c. char $name**

Ans) The variable name $name is invalid in C because variable names cannot start with a $ symbol.

3. Predict the value of a after each statement.

int main(void)

{

int i = 10;

char a = 'd';

a += 10;

a \*= 5;

a /= 4;

a %= 2;

a \*= a + i;

return 0;

}

Ans: **Step-by-step calculation:**

* a = 'd' = 100 (since ASCII value of 'd' is 100)
* a += 10;  
  a = 100 + 10 = 110
* a \*= 5;  
  a = 110 \* 5 = 550
* a /= 4;  
  a = 550 / 4 = 137 (integer division)
* a %= 2;  
  a = 137 % 2 = 1 (remainder of 137 divided by 2)
* a \*= a + i;  
  a = 1 \* (1 + 10) = 1 \* 11 = 11

**Final value of a:**  
a = 11

4. Consider a = 12, b = 3, predict the output of the following .

**a. (a>100) && (b<10)**

* (12 > 100) is false (or 0).
* (3 < 10) is true (or 1).
* false && true is false (or 0).

**Result:**  
0

**b. (a==4) && (b==2)**

* (12 == 4) is false (or 0).
* (3 == 2) is false (or 0).
* false && false is false (or 0).

**Result:**  
0

**c. (a==11) && (a++)**

* (12 == 11) is false (or 0).
* Since the first part is false, the second part (a++) is not evaluated due to short-circuiting in the && operator.
* The value of a is unchanged.

**Result:**  
0 (and a remains 12).

5. Consider a = 10, b = 11, predict the output of the following .

**a. (**a>10) || (b<10)

* (10 > 10) is false (or 0).
* (11 < 10) is false (or 0).
* false || false is false (or 0).

**Result:**0

**b. a || 12.12**

* a = 10, which is true (non-zero).
* Since the first part is true, the second part (12.12) is not evaluated due to short-circuiting in the || operator.

**Result:**  
true (or 1 in C).

**c. a || b**

* a = 10, which is true (non-zero).
* Since the first part is true, the second part (b = 11) is not evaluated due to short-circuiting.

**Result:**  
true (or 1 in C).

**d. !(a >** **5)**

* (10 > 5) is true (or 1).
* The negation !true gives false (or 0).

**Result:**

0  
6. Consider int age = 10, height = 45, year = 2000; Predict the output of the following.

**a.(age < 12 && height < 48) || (age > 65 && height > 72)**

* (10 < 12 && 45 < 48) is true (both conditions are true).
* (10 > 65 && 45 > 72) is false (both conditions are false).
* true || false is true.

**Result:**  
true (or 1 in C).

**b. (year % 4 == 0 && year % 100 != 0 ) || (year % 400 == 0);**

* (2000 % 4 == 0 && 2000 % 100 != 0) is true (2000 is divisible by 4 and not by 100).
* (2000 % 400 == 0) is true (2000 is divisible by 400).
* true || true is true.

**Result:**true (or 1 in C).