**Operators Associativity and Precedence Assignment**

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

**a. x = 34 + 12/4 – 56**

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

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

**d. 34 + 12/4 – 45**

**a.) x = 34 + 12/4 – 56**

* a.) Operator Precedence and Associativity:
  + First, division (/) is evaluated due to its higher precedence than addition (+) and subtraction (-).
  + Then, addition and subtraction are evaluated from left to right (left-to-right associativity for + and -).
* Step-by-step Evaluation:

12 / 4 = 3 (division happens first)

* + 1. + 3 = 37 (addition)

1. - 56 = -19 (subtraction)

Result: x = -19

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

* Operator Precedence and Associativity:
  + Division (/) has the highest precedence, so 4 / 2 = 2.
  + Then, addition and subtraction are evaluated from left to right, followed by comparison (<).
* Step-by-step Evaluation:

4 / 2 = 2

12 + 3 = 15

15 - 2 = 13

3 + 1 = 4

13 < 4 = FALSE (comparison)

* Result: FALSE

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

* Operator Precedence and Associativity:
  + Parentheses are evaluated first.
  + Then, multiplication (\*) is evaluated.
* Step-by-step Evaluation:

3 + 2 = 5 (innermost parentheses)

2 + 5 = 7 (outer parentheses)

7 \* 10 = 70 (multiplication)

* Result: 70

**d. 34 + 12 / 4 - 45**

* Operator Precedence and Associativity:
  + Division (/) is evaluated first.
  + Then, addition and subtraction are evaluated from left to right.
* Step-by-step Evaluation:

12 / 4 = 3 (division)

34 + 3 = 37 (addition)

37 - 45 = -8 (subtraction)

* Result: -8

**A. a . -19**

**b . FALSE**

**c. 70**

**d. -8**

**2. Rewrite the following expressions with improved readability**

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

Group the logical conditions to make the expression clearer.

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

**b. char name value**

The initialization is missing.

char name = value;

**c. char $name**

Valid expression: Variable names in C can't start with $. If $ is a special character, you may want to choose a valid variable name.

char name$;

**3. Predict the value of a after each statement.**

A white rectangular object with a black border

Description automatically generated with medium confidence

1. a += 10: 'd' (100) becomes 'n' (110).
2. a \*= 5: 'n' (110) becomes 550.
3. a /= 4: 550 / 4 = 137.
4. a %= 2: 137 % 2 = 1.
5. a \*= a + i: a = 1 \* (1 + 10) = 1 \* 11 = 11.

**Result**:

* Final value of a = 11.

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

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

**Evaluation**:

* (a > 100) is FALSE because 12 > 100 is false.
* (b < 10) is TRUE because 3 < 10 is true.
* Using && (logical AND), if one operand is FALSE, the result is FALSE.

**Result**: FALSE

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

**Evaluation**:

* (a == 4) is FALSE because 12 == 4 is false.
* (b == 2) is FALSE because 3 == 2 is false.
* && between two FALSE values results in FALSE.

**Result**: FALSE

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

**Evaluation**:

* (a == 11) is FALSE because 12 == 11 is false.
* The second part a++ is a post-increment, but since the first condition is FALSE, this part is not evaluated (short-circuiting).
* Using &&, if one operand is FALSE, the whole expression is FALSE.

**Result**: FALSE

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

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

**Evaluation**:

* (a > 10) is FALSE because 10 > 10 is false.
* (b < 10) is FALSE because 11 < 10 is false.
* Using || (logical OR), if both operands are FALSE, the result is FALSE.

**Result**: FALSE

b. a || 12.12

**Evaluation**:

* a is 10, which is non-zero and evaluates to TRUE.
* Since || (logical OR) only needs one TRUE operand to return TRUE, this expression evaluates to TRUE.

**Result**: TRUE

**c. a || b**

**Evaluation**:

* a is 10 (non-zero), so the result is TRUE.
* Since a is TRUE, the second part (b) is not even evaluated due to short-circuiting.

**Result**: TRUE

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

**Evaluation**:

* (a > 5) is TRUE because 10 > 5.
* !TRUE is FALSE.

**Result**: FALSE

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

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

**Evaluation**:

* (age < 12 && height < 48) is TRUE because both conditions are true (10 < 12 and 45 < 48).
* (age > 65 && height > 72) is FALSE because both conditions are false (10 > 65 is false and 45 > 72 is false).
* Using ||, if one operand is TRUE, the result is TRUE.

**Result**: TRUE

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

**Evaluation:**

* year = 2000. Check if it's a leap year:
  + 2000 % 4 == 0 is TRUE (2000 is divisible by 4).
  + 2000 % 100 != 0 is FALSE because 2000 is divisible by 100.
  + year % 400 == 0 is TRUE because 2000 is divisible by 400.
* Using ||, since the second condition (year % 400 == 0) is true, the result is TRUE.

**Result:** TRUE