THEORY QNS

**1. What is an Operator?**

An **operator** is a symbol that tells the compiler or interpreter to perform specific mathematical, logical, or other operations on variables or values in programming. Operators are used to manipulate data and variables within a program.

**Relational Operators:**

**Relational operators** compare two values or expressions and return a Boolean value (either true or false).

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| == | Equal to | x == y (true if x equals y) |
| != | Not equal to | x != y (true if x is not equal to y) |
| > | Greater than | x > y (true if x is greater than y) |
| < | Less than | x < y (true if x is less than y) |
| >= | Greater than or equal to | x >= y (true if x is greater than or equal to y) |
| <= | Less than or equal to | x <= y (true if x is less than or equal to y) |

**2. Difference Between Binary and Unary Operators**

* **Unary Operators**:
  + Unary operators work with a single operand.
  + They usually perform operations like negation, increment, or decrement.
  + Example:
    - +x (Unary plus, gives the value of x)
    - -x (Unary minus, negates the value of x)
    - ++x or x++ (Increment operator, adds 1 to x)
    - --x or x-- (Decrement operator, subtracts 1 from x)
* **Binary Operators**:
  + Binary operators work with two operands.
  + They perform operations like addition, subtraction, multiplication, etc.
  + Example:
    - x + y (Addition of x and y)
    - x - y (Subtraction of x and y)
    - x \* y (Multiplication of x and y)
    - x / y (Division of x by y)

**Summary:**

* Unary: One operand (e.g., ++x, -x)
* Binary: Two operands (e.g., x + y, x - y)

**3. What Do You Mean by Preceding Operators?**

**Operator precedence** defines the order in which different operators are evaluated when there are multiple operators in an expression. Operators with higher precedence are evaluated before operators with lower precedence.

For example, in the expression 3 + 5 \* 2, multiplication (\*) has a higher precedence than addition (+), so the multiplication happens first. The expression is evaluated as 3 + (5 \* 2) = 13.

**Operator Precedence Example:**

1. \*, / (multiplication and division) have higher precedence.
2. +, - (addition and subtraction) have lower precedence.

**4. What Do You Mean by BODMAS Operation?**

**BODMAS** stands for:

* **B**rackets
* **O**rders (Exponents, Roots, etc.)
* **D**ivision
* **M**ultiplication
* **A**ddition
* **S**ubtraction

It is a rule used to determine the order of operations in a mathematical expression to ensure consistency in results.

**Example:**

For the expression 2 + 3 \* (2^2 - 1), according to BODMAS:

1. Evaluate inside the brackets first: 2^2 - 1 = 3
2. Multiply: 3 \* 3 = 9
3. Add: 2 + 9 = 11

**5. Difference Between =, ==, and === Operators**

* **= (Assignment Operator)**:
  + Used to assign a value to a variable.
  + Example: x = 5 assigns the value 5 to the variable x.
* **== (Equality Operator)**:
  + Checks if two values are **equal**, but does not check the data type. It performs type coercion, meaning that it will convert the data types if necessary before comparison.
  + Example: 5 == '5' returns true because JavaScript converts '5' (string) to 5 (number).
* **=== (Strict Equality Operator)**:
  + Checks if two values are **equal** **and** they are of the **same data type**. It does not perform type coercion.
  + Example: 5 === '5' returns false because 5 is a number and '5' is a string, so they are not strictly equal.

**Objective MCQ’s**

**1. What is the value of the expression 4 \* 2 + 3?**

We use operator precedence (BODMAS) to evaluate the expression:

1. Multiplication is done first: 4 \* 2 = 8
2. Then, addition is performed: 8 + 3 = 11

So, the value is 11.

**Answer**: a) 11

**2. The logical AND (&&) operator returns TRUE if \_\_\_\_\_\_\_\_\_**

The logical AND (&&) operator returns true only if **both the left and right operands** are true.

**Answer**: c) The left and right operand both return a value true is true

**3. Which arithmetic operator can be used as both prefix and postfix operators?**

The **increment** (++) and **decrement** (--) operators can be used as both prefix and postfix operators. However, the question asks about an **arithmetic** operator, so the answer is ++.

**Answer**: a) ++

**4. Which of the following values can be assigned to a Boolean variable?**

A Boolean variable can store the values **True** or **False**.

**Answer**: a) True or False

**5. What value is assigned to the $ReturnValue variable in the statement $ReturnValue = 100 != 200;?**

The expression 100 != 200 checks if 100 is **not equal** to 200. Since this is true, the expression evaluates to True.

**Answer**: a) True

**6. Which of the following is an example of initializing a variable?**

Initialization refers to assigning a value to a variable at the time of declaration, such as $num = 2;.

**Answer**: a) $num = 2;

**7. A relational operator is used to \_\_\_\_\_\_\_\_\_**

A relational operator is used to **compare values** and return a Boolean result (true or false).

**Answer**: d) Compare the values

**8. Operators are used to perform some operation on the \_\_\_\_\_\_\_\_.**

Operators perform operations on **operands**, which are the values or variables involved in the operation.

**Answer**: a) Operands

PRACTICAL QN 4-8

**4. Exercise with $A = 34 and $B = 55:**

Given:

* $A = 34
* $B = 55

**a) $A == 34 && $B == 55**

* **Explanation**: The && operator checks if both conditions are true. Here, $A == 34 is true, and $B == 55 is also true.
* **Result**: true

**b) $A >= 30 || $B <= 50**

* **Explanation**: The || operator checks if **at least one** of the conditions is true. $A >= 30 is true, but $B <= 50 is false. Since one condition is true, the whole expression evaluates to true.
* **Result**: true

**c) $B == 55 || $A == 35**

* **Explanation**: The || operator checks if at least one condition is true. $B == 55 is true, but $A == 35 is false. Since one condition is true, the result is true.
* **Result**: true

**d) $A != 34**

* **Explanation**: The != operator checks if $A is **not equal** to 34. Since $A == 34, this expression is false.
* **Result**: false

**e) $A >= 30 && $A < 35**

* **Explanation**: Both conditions are checked with &&. $A >= 30 is true, and $A < 35 is also true because $A is exactly 34.
* **Result**: true

**f) $B > 50 || $B < 56**

* **Explanation**: The || operator checks if **either** condition is true. $B > 50 is true, and $B < 56 is also true. Since both conditions are true, the result is true.
* **Result**: true

5. <?php

$a = 33;

$b = 55;

$a += $b; // $a = 33 + 55 = 88

$c = $a; // $c = 88

$c -= $b; // $c = 88 - 55 = 33

$c \*= $a; // $c = 33 \* 88 = 2904

$a++; // $a = 88 + 1 = 89

echo "a=$a ,b=$b , c=$c";

?>

6. <?php

$a = 8;

$b = ++$a + 5; // $a becomes 9, $b = 9 + 5 = 14

$c = $b-- + 10; // $c = 14 + 10 = 24, then $b becomes 13

echo "a= $a , b= $b , c = $c ";

?>

a= 9 , b= 13 , c= 24

**7. Output and Justification:**

**a) echo 4 + 2 - 12 \* 3;**

* **Explanation**: Apply operator precedence (BODMAS). Multiplication (\*) happens first:
  + 12 \* 3 = 36
  + Then, addition and subtraction: 4 + 2 - 36 = 6 - 36 = -30
* **Output**: -30

**b) echo 4 + (2 - 12) \* 3;**

* **Explanation**: Solve inside parentheses first:
  + 2 - 12 = -10
  + Then, multiply: -10 \* 3 = -30
  + Finally, add: 4 + (-30) = -26
* **Output**: -26

**c) echo ((4 + 2) - 12) \* 3;**

* **Explanation**: Solve inside the first parentheses:
  + 4 + 2 = 6
  + Then subtract: 6 - 12 = -6
  + Finally, multiply: -6 \* 3 = -18
* **Output**: -18

**8. Values Assigned to $ReturnValue:**

**a) $ReturnValue = 2 == 3;**

* **Explanation**: 2 == 3 evaluates to false.
* **Output**: false

**b) $ReturnValue = "2" + "3";**

* **Explanation**: PHP performs implicit type conversion and adds the two strings as numbers: "2" + "3" = 5.
* **Output**: 5

**c) $ReturnValue = 2 > 3;**

* **Explanation**: 2 > 3 evaluates to false.
* **Output**: false

**d) $ReturnValue = 2 < 3;**

* **Explanation**: 2 < 3 evaluates to true.
* **Output**: true

**e) $ReturnValue = (2 > 3) && (2 < 3);**

* **Explanation**: The first condition (2 > 3) is false. The && operator returns false if either condition is false.
* **Output**: false

**f) $ReturnValue = (2 > 3) || (2 < 3);**

* **Explanation**: The first condition (2 > 3) is false, but the second condition (2 < 3) is true. The || operator returns true if at least one condition is true.
* **Output**: true