BITWISE COMPLEMENT

The term "bitwise complement" refers to a unary operation that operates on individual bits within a binary representation of a number. It is also commonly known as "bitwise negation" or "bitwise inversion."

The bitwise complement operation flips the bits of a binary number, changing 0s to 1s and 1s to 0s. This operation is often used in computer programming and digital systems to manipulate and perform operations on binary data at the bit level.

In many programming languages, the tilde (~) symbol denotes the bitwise complement operator. For example, if x is a variable holding a binary value, then ~x represents the bitwise complement of x.

Two's complement representation: In most programming languages, integers are represented using the two's complement notation. The bitwise complement operation obtains the two's complement of a number and then adds 1 to the result. This is often used for arithmetic operations like subtraction.

Example: Let's consider the binary number 10101010. Applying the bitwise complement operation to this number will result in 01010101. Each bit has been flipped.

LOGICAL COMPLEMENT

The term "logical complement" refers to the logical negation of a proposition or a boolean value. It is also known as "logical not."

Logical complements are used in logic and programming to reverse a boolean expression's truth value or test the opposite condition. The result of applying the logical complement operation to a boolean value is the opposite of its original truth value.

The logical complement operator is denoted by the exclamation mark (!) symbol. For example, if x is a boolean variable, then !x represents the logical complement of x.

In boolean logic, there are two possible truth values: true and false. Applying the logical complement operation negates the original truth value. If the original value is true, the complement is false, and vice versa.

Logical complements are used in various programming contexts, such as:

- Conditional statements: Logical complements can be used to test the opposite condition in if statements and loops.
- Boolean algebra: Logical complements are essential for performing logical operations like AND, OR, and XOR.
- De Morgan's laws: De Morgan's laws state that the complement of a logical expression involving AND or OR can be obtained by complementing the individual terms and changing the logical operator.

Truth tables are often used to illustrate the behavior of logical complements and other logical operations. A truth table lists all possible input combinations and their corresponding outputs based on the logical operation being applied.

WHAT WILL HAPPEN IF ANY NUMBER GREATER THAN THE LONG DATA TYPE

If the number exceeds the long datatype range, then it will circle back or it will cycle to the minimum range of the long datatype i.e., from Long.MAX_VALUE+1 to Long.MIN_VALUE.

FLOAT AND DOUBLE RANGE

1. Float:

- Size: 32 bits (4 bytes)
- Range: Approximately ±1.4e-45 to ±3.4e+38
- Precision: About 7 decimal digits

The float data type is used to represent single-precision floating-point numbers. It provides a good balance between range and precision and is suitable for most general-purpose floating-point calculations.

2. Double:

- Size: 64 bits (8 bytes)
- Range: Approximately ±4.9e-324 to ±1.8e+308
- Precision: About 15 decimal digits

The double data type is used to represent double-precision floating-point numbers. It offers a larger range and higher precision compared to float and is commonly used when more significant digits and a wider range are required.