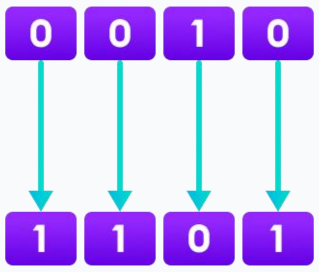
Java Bitwise Complement Operator:

The bitwise complement operator is a unary operator (works with only one operand). It is denoted by `~`
It changes binary digits **1** to **0** and **0** to **1**.



It is important to note that the bitwise complement of any integer **N** is equal to **- (N + 1)**. For example,
Consider an integer **35**. As per the rule, the bitwise complement of **35** should be **-(35 + 1)** = **-36**.
Example:

```
35 = 00100011 (In Binary)

// using bitwise complement operator
~ 00100011
 ________
  11011100
```

In the above example, we get that the bitwise complement of **00100011** (**35**) is **11011100**. Here, if we convert the result into decimal we get **220**.
However, it is important to note that we cannot directly convert the result into decimal and get the desired output. This is because the binary result **11011100** is also equivalent to **-36**.
To understand this we first need to calculate the binary output of **-36**.

2's Complement

In binary arithmetic, we can calculate the binary negative of an integer using 2's complement.

1's complement changes **0** to **1** and **1** to **0**. And, if we add **1** to the result of the 1's complement, we get the 2's complement of the original number.
For example,

```
// compute the 2's complement of 36
36 = 00100100 (In Binary)
```

```
1's complement = 11011011

2's complement:
 11011011
   +   1

__________
 11011100
```

Here, we can see the 2's complement of **36** (i.e. **-36**) is **11011100**. This value is equivalent to the bitwise complement of **35**.
Hence, we can say that the bitwise complement of **35** is **-(35 + 1) = -36**.

Example 3: Bitwise Complement

```
class Main {
  public static void main(String[] args) {

    int number = 35, result;

    // bitwise complement of 35
    result = ~number;
    System.out.println(result);    // prints -36
  }
}
```

Java Logical Complement Operator

A logical complement operator inverts the value of a boolean

```
public class Example {

   public static void main(String[] args) {

      // assignment operator
      boolean b = true;

      // logical complement operator
      b = !b;

      System.out.println("b = " + b);

   }
}
```

Output:

```
$ java Example
b = false
```

Question: what if the number is more than in long datatype

The largest integer number that a long type can represent is 9223372036854775807. If we deal with even larger numbers, we have to use the java.math. BigInteger class. It is used to represent immutable *arbitrary precision* integers. Arbitrary precision integers are only limited by the amount of computer memory available.

With the help of the java.math.BigInteger class, we multiply two very large numbers.

```
System.out.println(Long.MAX_VALUE);
```

We print the largest integer value which can be represented by a long type.

```
BigInteger b = new BigInteger("92233720368547758071");
BigInteger c = new BigInteger("52498235605326345645");
```

We define two BigInteger objects. They both hold larger values that a long type can hold.

```
BigInteger a = b.multiply(c);
```

With the multiply method, we multiply the two numbers. Note that the BigInteger numbers are immutable. The operation returns a new value which we assign to a new variable.

```
System.out.println(a);
```

The computed integer is printed to the console.

```
$ java VeryLargeIntegers.java
9223372036854775807
4842107582663807707870321673775984450795
```

Question: what range can float and double store

In Java, the Float and Double wrapper classes have two properties that return the upper and lower limits of float and double data types, *MIN_VALUE* and *MAX_VALUE*:

```
System.out.println(Double.MAX_VALUE);
System.out.println(Double.MIN_VALUE);

System.out.println(Float.MAX_VALUE);

System.out.println(Float.MIN_VALUE);
```

Some processors and operating systems implement the JVM differently, so different platforms may have some discrepancy between the range of a double or float.

From the output of the program above, we see the size difference between float and double Java types is:

1. The upper range of a double in Java is 1.7976931348623157E308.
2. The lower range of a double in Java is 4.9E-324.
3. The upper range of a float in Java is 3.4028235E38.
4. The lower range of a float in Java is 1.4E-45.