What are increment and decrement operators?

Increment operator is used to increment the value of a variable by one. Similarly, decrement operator is used to decrement the value of a variable by one.

Increment

```
int a = 5;
a++;

a = 6
```

Decrement

```
int a = 5;
a--;

a = 4
```

```
a++; is same as a = a + 1;
a--; is same as a = a - 1;
```

Increment and Decrement operators

- Both are unary operators.
  - because they are applied on single operand.

```
a++;
```

```
a ++ a;
```

Increment and Decrement operators

Pre-increment operator

```
++a;
```

Post-increment operator

```
a++;
```

Pre-decrement operator

```
--a;
```

Post-decrement operator

```
a--;
```

Increment and Decrement operators

- You cannot use rvalue before or after increment/decrement operator.

Example:

```
(a + b)++;    error!

++(a + b);    error!
```

```
error: lvalue required as increment operand
```

lvalue (left value) : simply means an object that has an identifiable location in memory (i.e. having an address).

- In any assignment statement "lvalue" must have the capability to hold the data
- lvalue must be a variable because they have the capability to store the data.
- Lvalue cannot be a function, expression (like a+b) or a constant (like 3, 4 etc).

rvalue (right value) : simply means an object that has no identifiable location in memory.

- Anything which is capable of returning a constant expression or value.
- Expressions like a + b will return some constant value.

```
(a + b)++;    error!

++(a + b);    error!
```

```
error: lvalue required as increment operand
```

Compiler is expecting a variable as an increment operand but we are providing an expression `(a + b)` which does not have the capability to store data.

Question: What is the difference between pre-increment and post-increment operator OR pre-decrement and post-decrement operator?

Pre - means first increment/decrement then assign it to another variable .

Post - means first assign it to another variable then increment/decrement.

```
x = ++a;
```

x = 6

```
x = a++;
```

x = 5

Q1: What is the output of the following C program fragment:

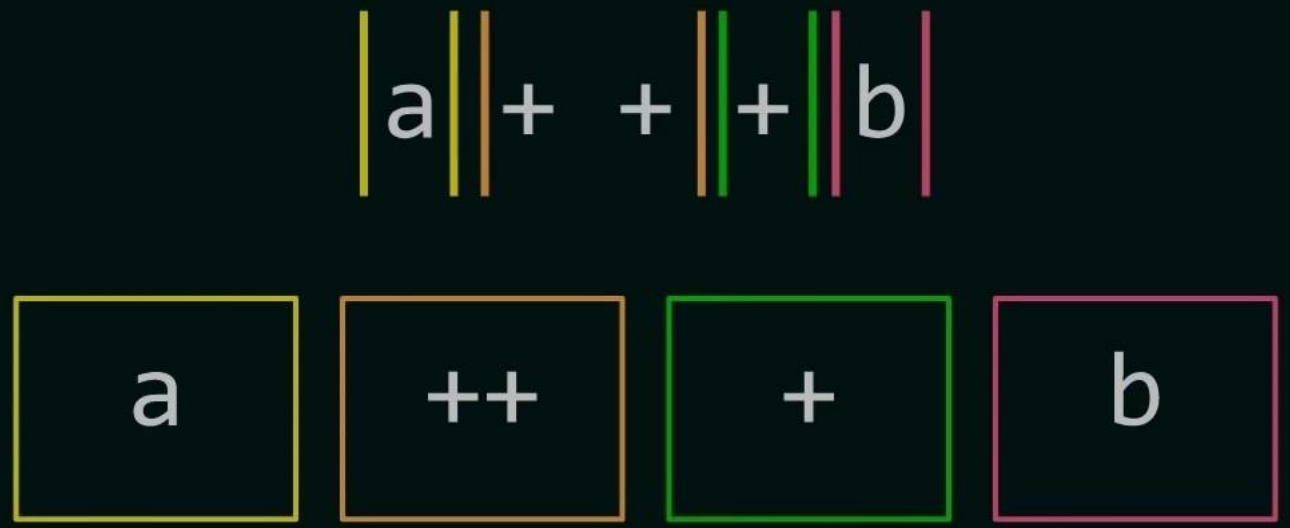
```c
#include <stdio.h>

int main() {
    int a = 4, b = 3;
    printf("%d", a+++b);
    return 0;
}
```

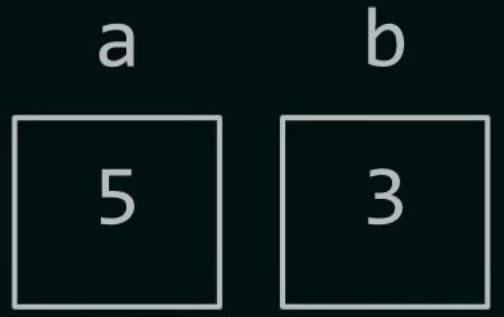
a+++b

Token Generation

- ★ Lexical analysis is the first phase in the compilation process.
- ★ Lexical analyzer (scanner) scans the whole source program and when it finds the meaningful sequence of characters (lexemes) then it converts it into a token
- ★ Token: lexemes mapped into token-name and attribute-value.
  Example: int -> <keyword, int>
- ★ It always matches the longest character sequence.



a++ + b



4 + 3

Post increment/decrement in context of equation:

First use the value in the equation and then increment the value

Pre increment/decrement in context of equation:

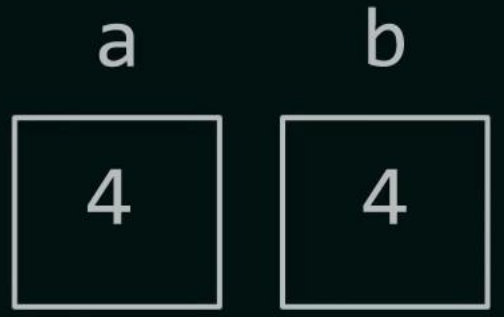
First increment the value and then use in the equation after completion of the equation.

Q2: What is the output of the following C program fragment:

```c
#include <stdio.h>

int main() {
    int a = 4, b = 3;
    printf("%d", a + ++b);
    return 0;
}
```

a+ ++b



4 + 4

Post increment/decrement in context of equation:

First use the value in the equation and then increment the value

Pre increment/decrement in context of equation:

First increment the value and then use in the equation after completion of the equation.

Homework

Q3: What is the output of the following C program fragment:

```c
#include <stdio.h>

int main() {
    int a = 4, b = 3;
    printf("%d", a+++++b);
    return 0;
}
```

a) 7
b) 8
c) 9
d) Error

You can post your answer in the comment section below