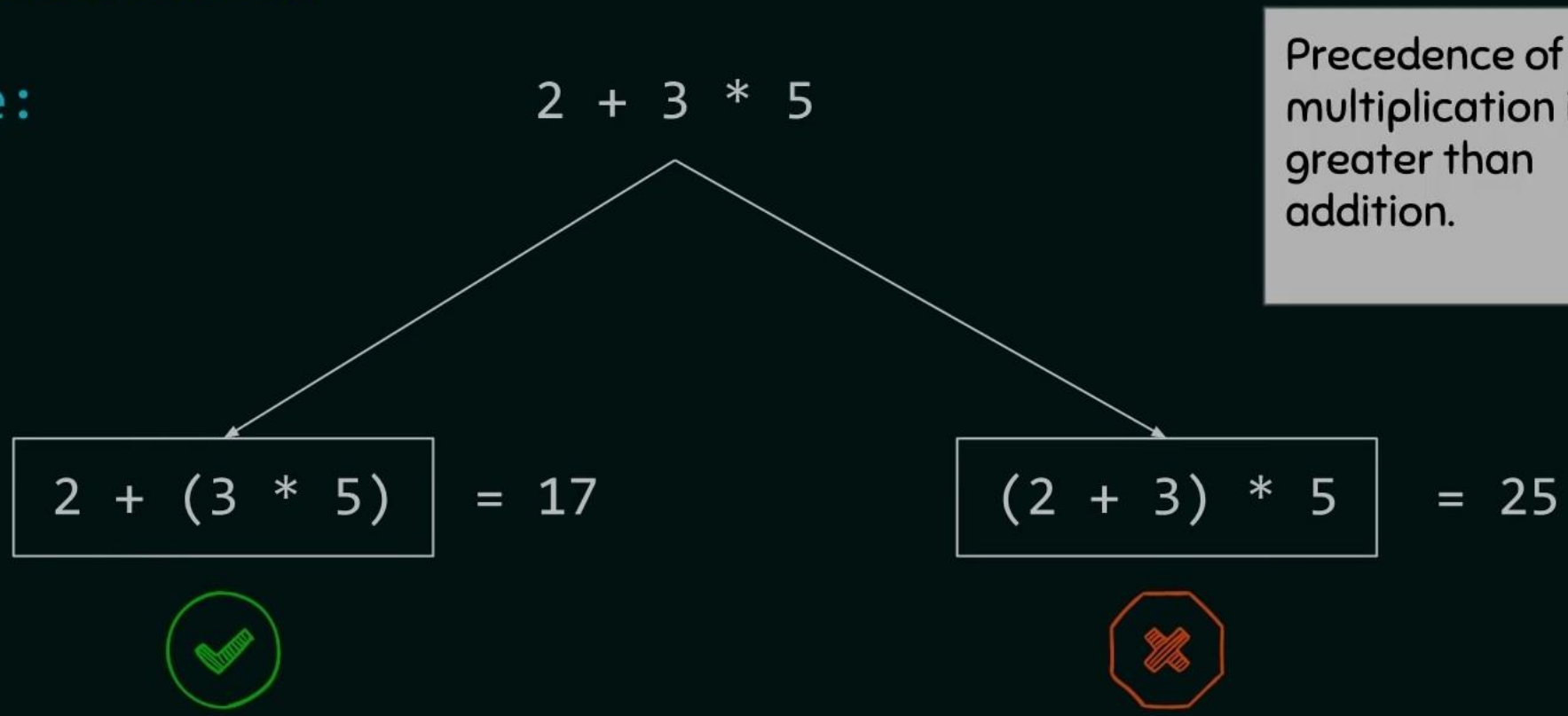
Precedence of Operators

Precedence of operators came into picture when in an expression we need to decide which operator will be evaluated first. Operator with higher precedence will be evaluated first.

Example:



Associativity of Operators

Associativity of operators came into picture when precedence of operators are same and we need to decide which operator will be evaluated first.

Example: `10 / 2 * 5`

Left to right: `(10 / 2) * 5` = 25 

Right to left: `10 / (2 * 5)` = 1 

Associativity can be either:

1. Left to right
   OR
2. Right to left

Precedence and associativity table

Category	Operators	Associativity
Parenthesis/brackets	() [] -> . ++ --	Left to right
Unary	! ~ ++ -- + - * & (type) sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Bitwise Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right

Precedence and associativity table

Category	Operators	Associativity
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	? :	Right to left
Assignment	= += -= *= /= %= &= ^= |= <<= >>=	Right to left
Comma	,	Left to right

() - parenthesis in function calls.

Example:

```
int var =fun();
```

operand

= operator is having less precedence as compared to ()
therefore, () belongs to fun and fun will be treated as a function.

```
int var =(fun());
```

If suppose = operator is having greater precedence then, fun will belong to = operator and therefore it will be treated as a variable.

```
int (var =fun)();
```

Member access operators (-> .)

They are used to access members of structures.

We will talk about structures later in this course.

Postfix Increment/Decrement (++, --)

Precedence of Postfix Increment/Decrement operator is greater than Prefix Increment/Decrement.

Associativity of Postfix is also different from Prefix. Associativity of postfix operators is from left to right and that of prefix operators is from right to left.

Example:

```
int main()
{
    int a;
    a = fun1() + fun2();
    printf("%d", a);
    return 0;
}
```

```
int fun1()
{   printf("Neso");
    return 1;
}

int fun2()
{   printf("Academy");
    return 1;
}
```

Which function is called first ?
fun1() or fun2()?

It is not defined whether fun1() will be called first or whether fun2() will be called. Behaviour is undefined and output is compiler dependent.

NOTE: Here associativity will not come into picture as we have just one operator and which function will be called first is undefined. Associativity will only work when we have more than one operators of same precedence

Important Facts

Associativity can only help if there are two or more operators of same precedence and not when there is just one operator.

Operators with same precedence have same associativity as well.

Homework Problem

What is the output of the following C program fragment?

```c
#include <stdio.h>

int main() {
    //code
    int a=10, b=20, c=30, d=40;
    if(a <= b == d > c)
    {
        printf("TRUE");
    }
    else
    {
        printf("FALSE");
    }
    return 0;
}
```

a) TRUE
b) FALSE