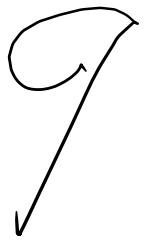
Let’s solve it



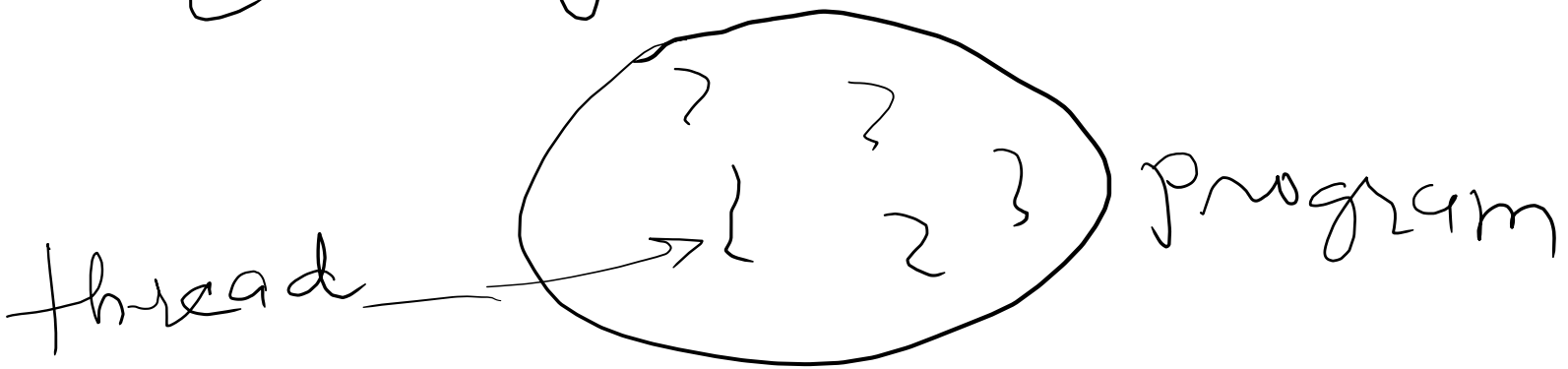
Volatile - something which changes rapidly or externally and hence compiler is required to check the latest value everytime the variable used / fetch fresh.

This instructs compiler not to optimize even though this variable is not on the left hand side of assignment. Meaning not to assume that the variable will not be changed, it may.

If you want to enforce that this program is not going to change /update then

const volatile int x = 10;

mostly possible when multithreading is ongoing.



categories of operator

- Arithmetic operators
  - +, -, *, /, %
- Relational operators
  - >
  - <
  - ==
  - >=
  - <=
  - != Not equal to

comparison

<=
>=
<
>

! Tested OK just for fun

10 > 20 false

!0 < 20 true

(5++) < (13 + 7)

12 < (21)

true

Precedence
priority

Associativity

Be careful when you apply negation.

!(10 > 20) true

15/10

$\Rightarrow 1$

Integer arithmetic

Real Arithmetic

15.0/10.0

$\Rightarrow 1.5$

```
   1.5
 10)15
    10
    --
    50
   -50
    --
     0
```

mixed arithmetic

15/10.0 or 15.0/10

15.0 / 10

⇒ 15.0 / 10.0

⇒ 1.5

15 / 10.0

15.0 / 10.0

⇒ 1.5

10 | 15

1 0

5

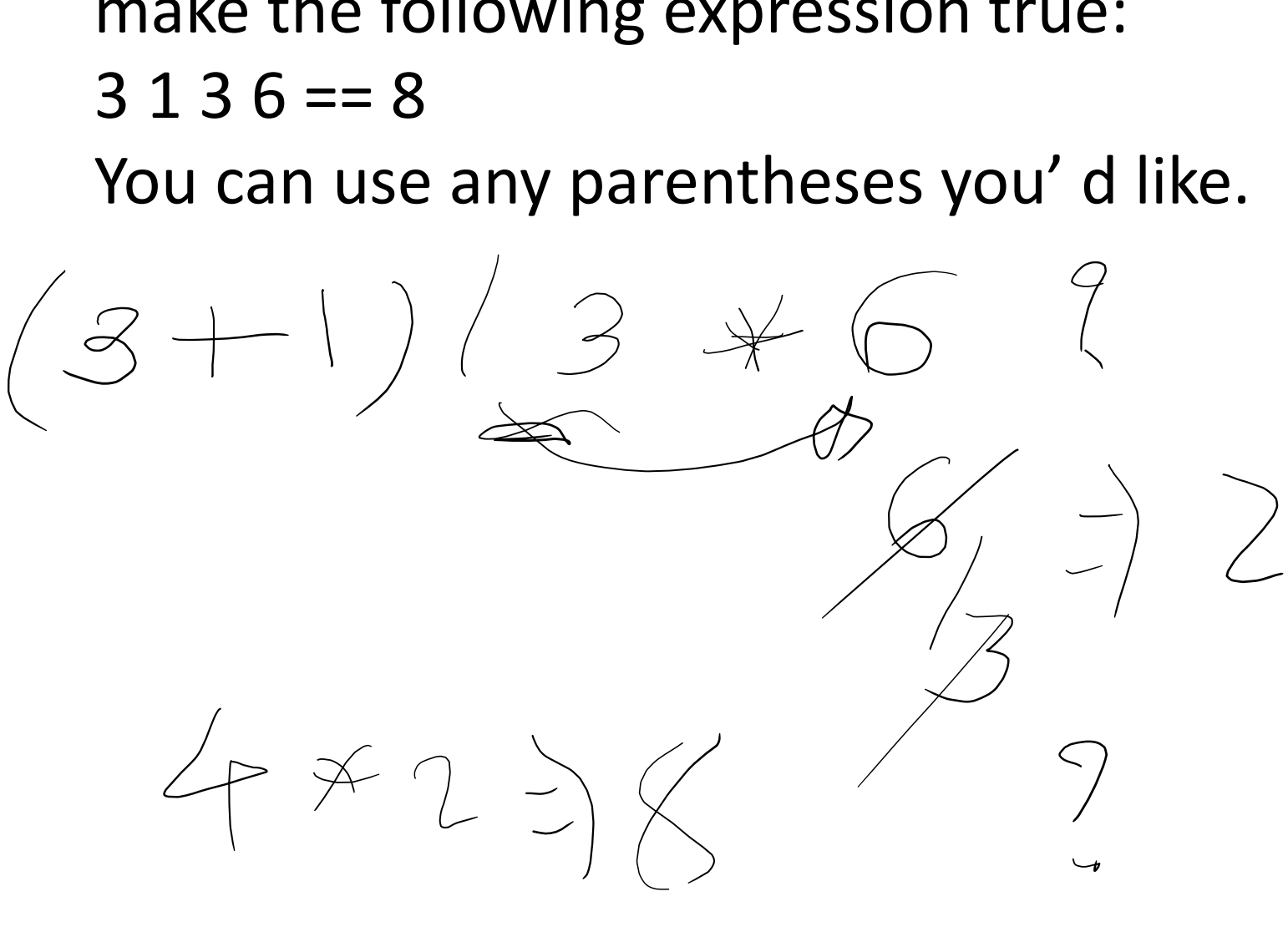
remainder

% modulo

Add arithmetic operators (plus, minus, times, divide) to make the following expression true:

3 1 3 6 == 8

You can use any parentheses you' d like.



If done integer arithmetic only

(3+1) ⇒ 4

4 / 3 ⇒ 1

1 * 6 ⇒ 6 and not 8 !

we need either float variables
or 4/3.0 or 4.0/3 to get 1.333

Increment & decrement operators

CR6 C++

int i = 10;

i++; // post increment

i = i + 1;

Increase the value of i by 1.

x = i++;

int x;

int j = 20;

++i; // pre-increment

int y;

y = ++j;

21

j 21

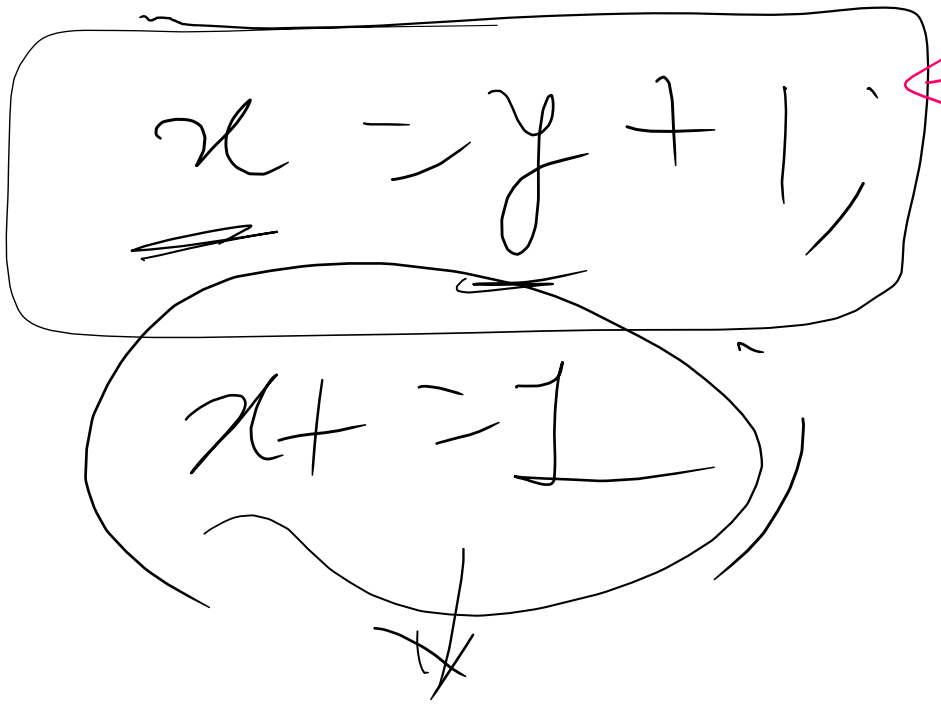
i = i + 1;

i++; or ++i;

i += 1; // i = i + 1;

shorthand / short form operat

i = i - 1; i -= 1; i = i - 1;



Can not use +=

y += 1;

x = x + 1;

y = y + 1

shorthand operators +=, -= like can be used only when same operand on both sides of expression regular.

comma operator

```
int x, y, z;
```

```
x = 10;
y = 20;
z = 30;
```

```
x = 10, y = 20, z = 30;
```

i.e for(i=0, j=n ; ——— ; i++, j--)

puzzle: know

I want to ~~count~~ number of students present in the class without me counting it & counted by any one single person. How do I get this answer?