Operator Overloading - 1

=> before Starting do Refer to "fraction" CODE, Commented inside Operator_Overloading-1.cpp

Now, int a = 12;
int b = 13;
int C = a + b;
we Can use '+' operator On (int, doubles ..)

but we Cannot use it as
Fraction f1(2,3);
Fraction f2(3,7);
Fraction f3 = f1 + f2;
this will absolute gives Error as of now

So, to Solve Such issue & Perform all our operators on our class we need to use "Operator Overloading".

Lets See ——

In our "fraction" CODE, we have a function that we created as "add".

i.e,

```
Void add(Fraction Const &f2) {
    int lcm = den_ * f2.deno_;
    int x = lcm/denom_;
    int y = lcm/f2.den_;

    int num = x * numerator + (y * f2.num_);

    nume_ = num;
    den_ = lcm;
    Simplify();
}
```

→ what we are basically doing is f1.add(f2) & then we Store our Result in f1 only.

Now Since, we want our function to Return out a fraction we need to modify it (Void with fraction)
& use Return to Return our New fraction –

```
Fraction  add(Fraction Const &F2) {
    int lcm = ________;
    int x = ________;
    int y = ________;

    int num = ________;

    Fraction fNew(num, lcm);
    fNew. Simplify ();
    return fNew;
}
int main() {
    Fraction f1(10, 2);
    Fraction f2(15, 4);
    Fraction f3 = f1.add (f2);
     f3. Print();
}
```

output :- 35/4 ✓

what is happening inside ⟶

$$\underset{\text{this}}{\text{Fraction } f_3 = f_1} \,.\, \text{add}(\underset{\text{argument}}{f_2});$$

but, now we don't want to use this "f_1.add (f_2)";

Lets use our "operator overloading" now

"Fraction $\boxed{\text{operator +}}$(Fraction Const &F2){"

this makes us to use "$f_1 + f_2$"

where,

$$\text{Fraction } f_4 = \underset{\text{this}}{f_1} + \underset{\text{argument}}{f_2};$$

doing these changes,

we now too get

output ⟶ 35/4

Same thing we had done with ~~in~~ multiply & also used "==" in our Code do Refer it.