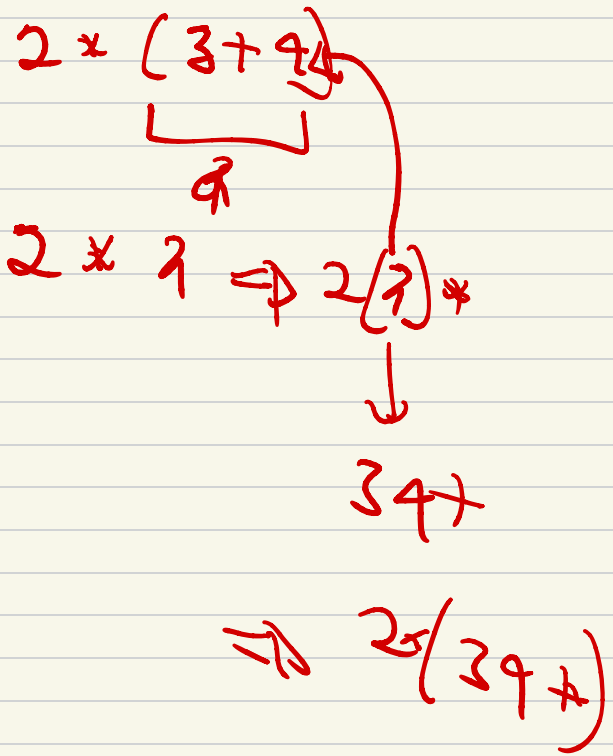
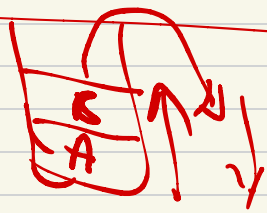
Infix : 1 + 1

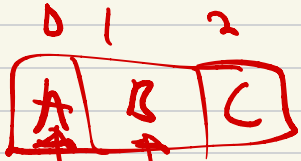
Postfix : 1 1 +



(A + B / C * (D + C) - F)

- Stack
- List / Array





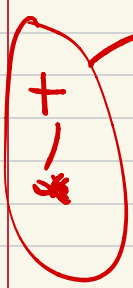
Operators :

Operants :

stack

list

order



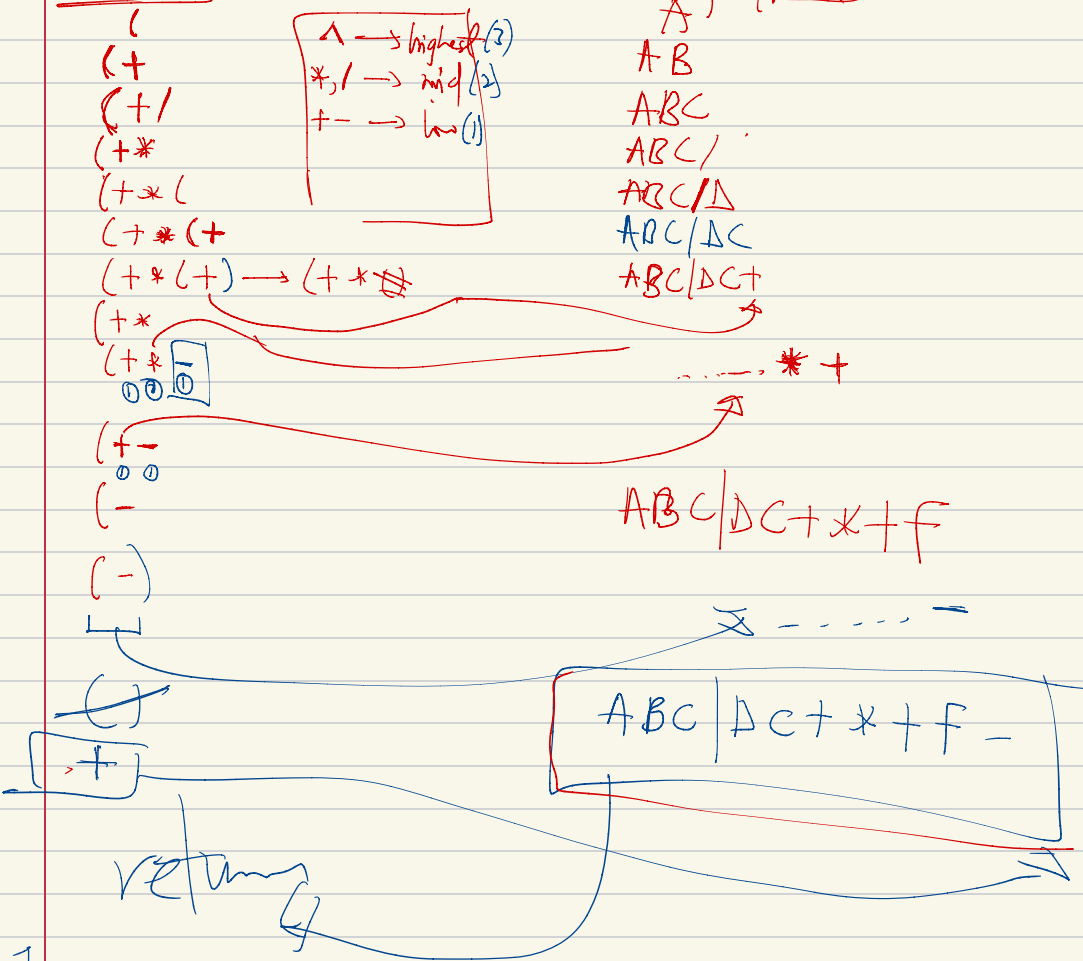
stack

2 + (5 * 2

(A + B / C * (D + C) - F)

Stack (operator)

List Postfix / Result / Number



Rules 1

① Incoming element w/ higher prec should only be added to the stack

↳ ① = ① ⇒ preserve only current incoming elem.

↳ ② = ① ↳ pop/remove existing

② if ")", get rid of everything b4 until you reach "("

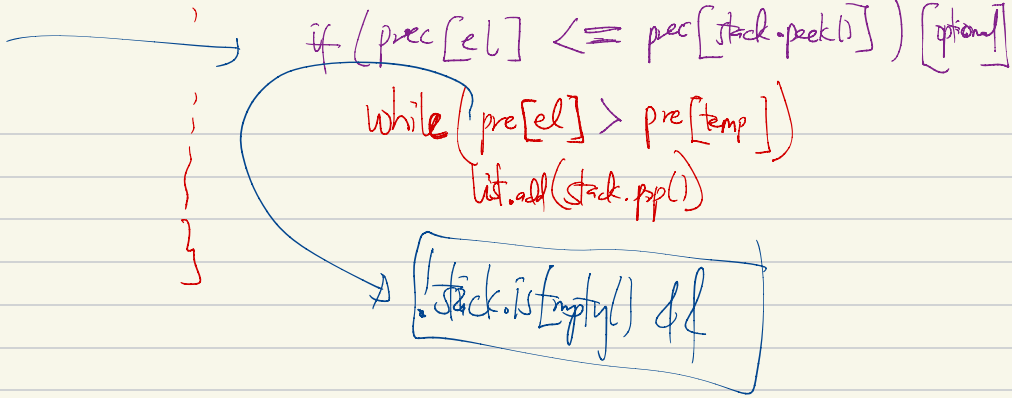
③ Add whatever elements left in stack to list as part of the final result

```
function infixToPostfix (inp){
    Stack stack;
    ListResult list; [ ]
    Prec pre = { "+" : 1,
                 "-" : 1,
                 "*" : 2,
                 "/" : 2
                 "^" : 3
    }
    for (el in inp) {
        if (el is Number)
            list.add(el)
        if (el == "(")
            stack.add(el)
        else if (el == ")") {
            pel = stack.peek()
            while (pel != "(")
                popel = stack.pop()
                list.add(popel)
            if (pel == ")") (optional)
                stack.pop()

        if (el is in the prec keys) {
            if (prec[el] > prec[stack.peek()])
                stack.add(el)
        }
    }
```

!stack.isEmpty() &&

pr.hasKey(el)


if (prec[el] <= prec[stack.peek()]) [optional]
while (pre[el] > pre[temp])
list.add(stack.pop())
!stack.isEmpty() &&