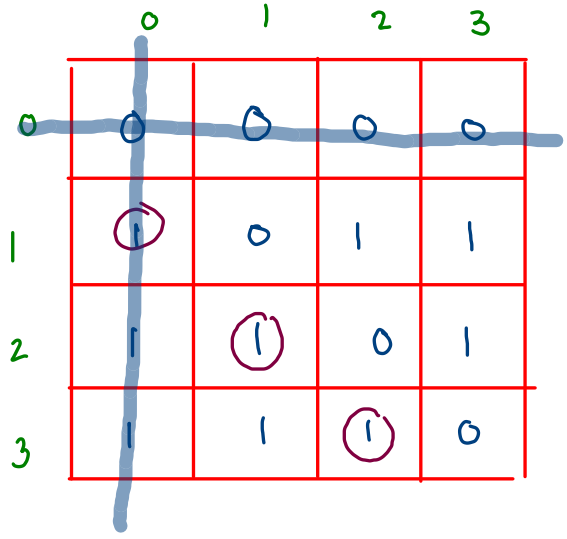
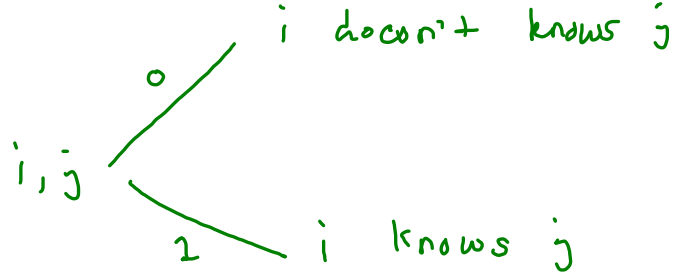
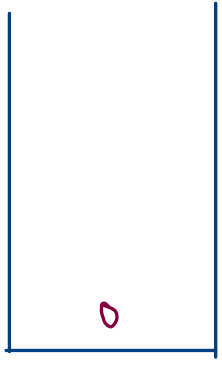
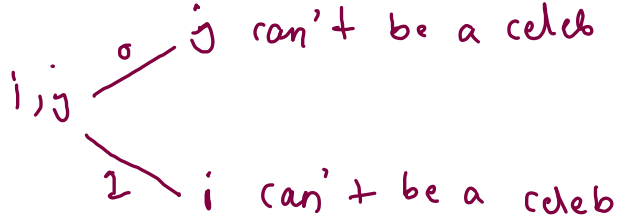
Celebrity Problem

```
4
0000
1011
1101
1110
```

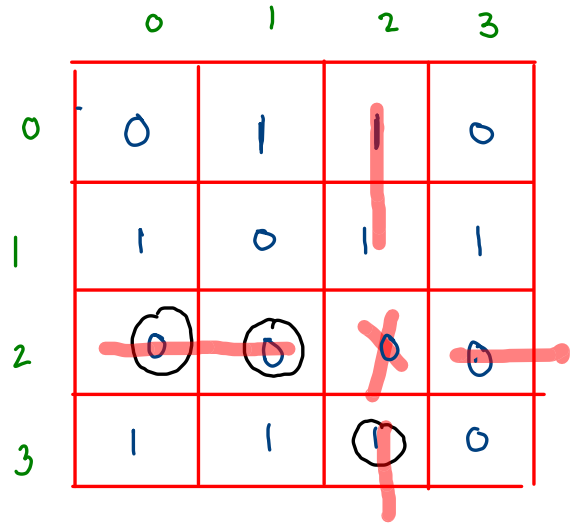


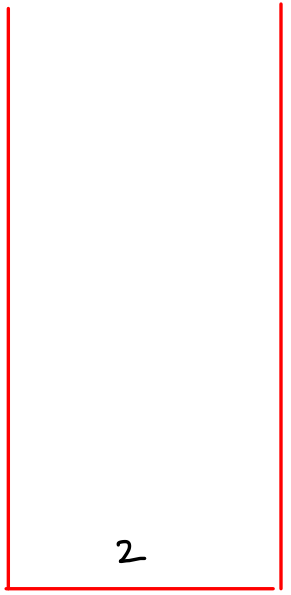


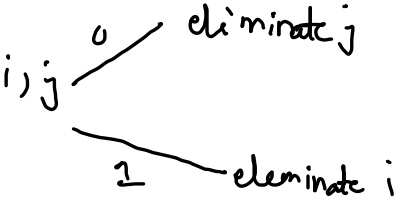




celebrity: knows no-one,
known by everyone.







0 1 2 3

	stack	two pointer
T:	O(n)	O(n)
S:	O(n)	O(1)

Sliding Window Maximum

k = 4,

O(n)

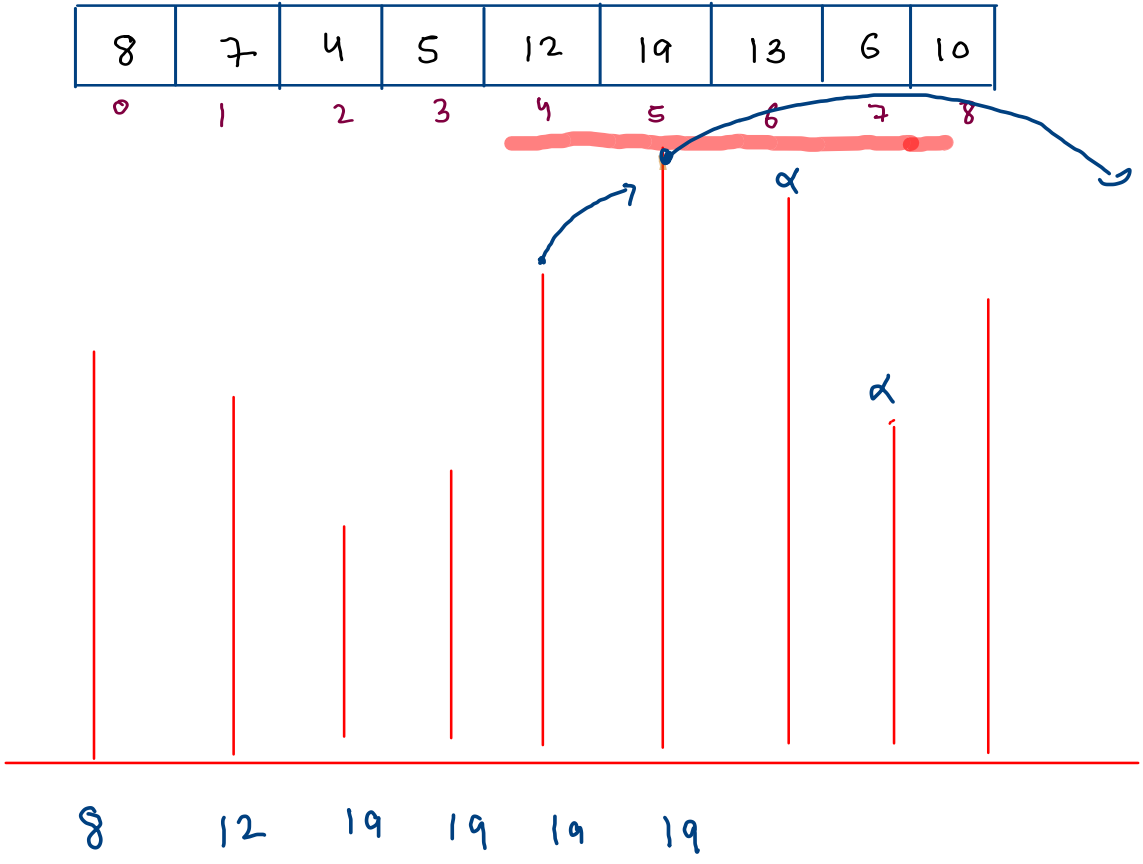
8	7	4	5	12	19	13	6	10
0	1	2	3	4	5	6	7	8

n , k

last window : n-k

start point.

total window : n-k+1

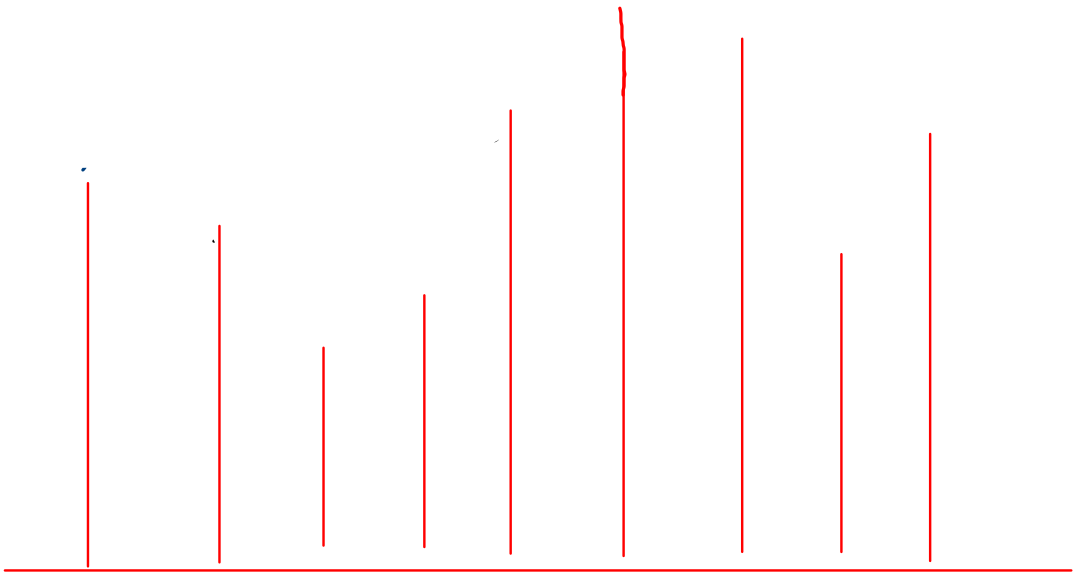


ngr

k = 4

ngr - idx based

```java
for(int i = 0; i <= n-k;i++) {
    int j = i;

    while(ngr[j] < i + k) {
        j = ngr[j];
    }

    System.out.println(arr[j]);
}
```



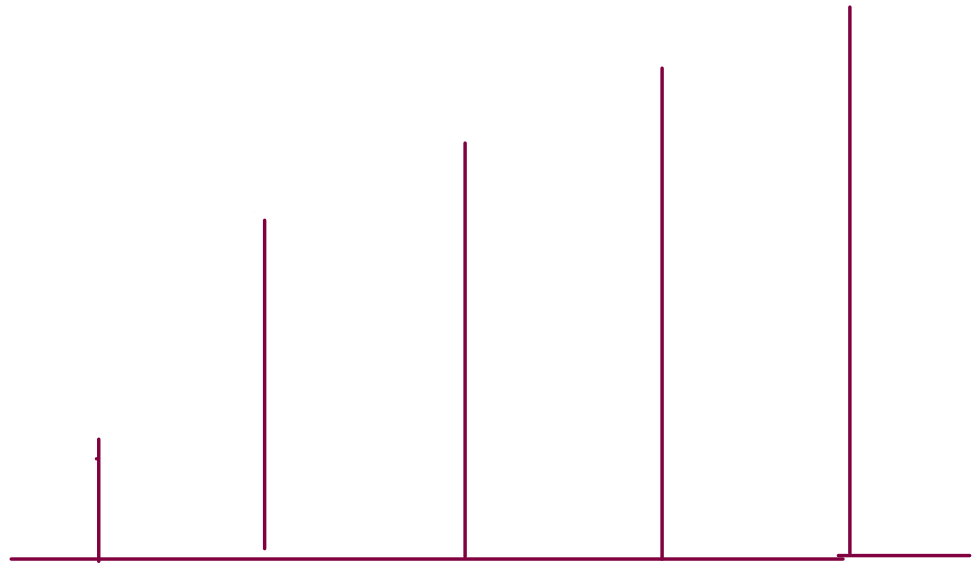
$k = 4$

i → window start

j → jump

val	8_0	7_1	4_2	5_3	12_4	19_5	13_6	6_7	10_8
ngri	4	4	3	4	5	9	9	8	9
wm	8	12	19	19	19	19			

```
for(int i = 0; i <= n-k;i++) {
    int j = i;

    while(ngr[j] < i + k) {
        j = ngr[j];
    }

    System.out.println(arr[j]);
}
```

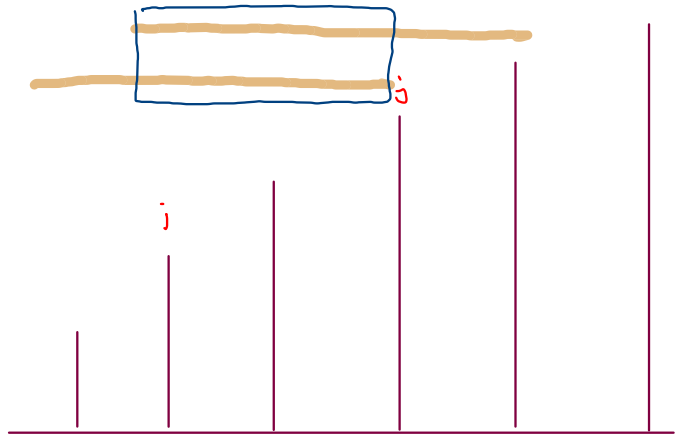


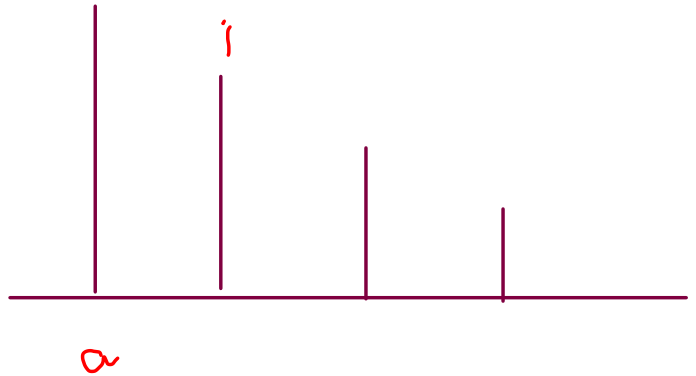
ele a_0 b_1 c_2 d_3 e_4 n = 5

ngr 1 2 3 4 5 k = 3

wm

```java
int j = 0;
for(int i = 0; i <= n-k;i++) {

    if(j < i) {
        j = i;
    }

    while(ngr[j] < i + k) {
        j = ngr[j];
    }

    System.out.println(arr[j]);
}
```

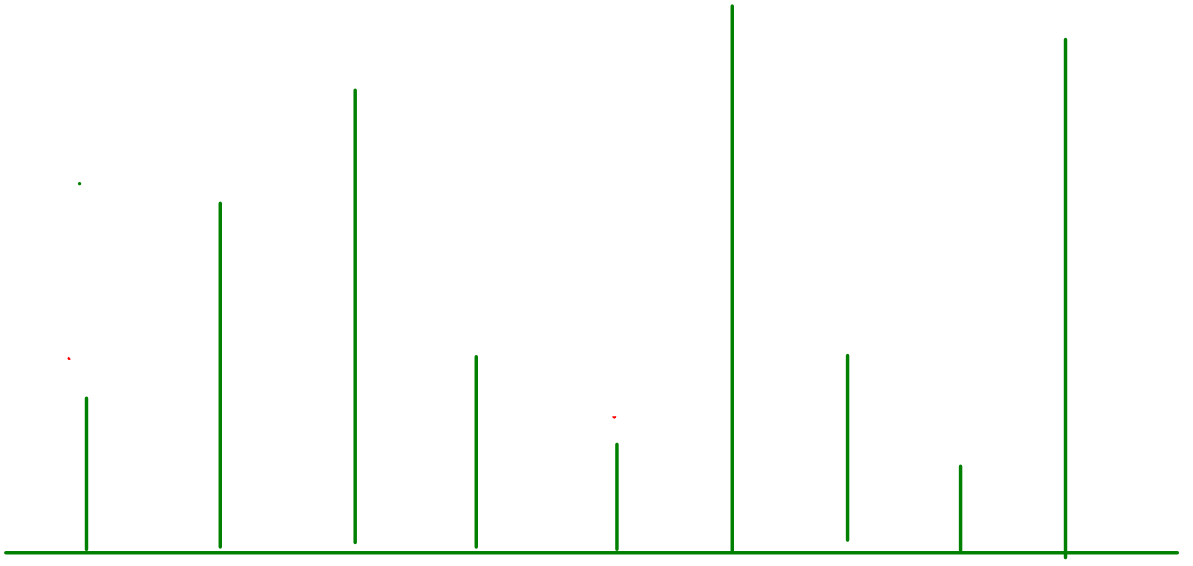




```java
int j = 0;
for(int i = 0; i <= n-k;i++) {

    if(j < i) {
        j = i;
    }

    while(ngr[j] < i + k) {
        j = ngr[j];
    }

    System.out.println(arr[j]);
}
```

i -> n

j -> n

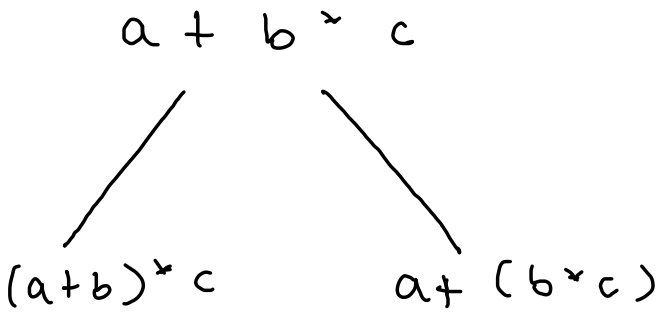


k = 4

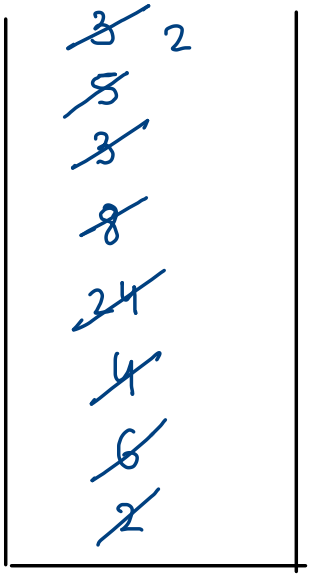
Infix Evaluation

1. Expression is balanced
2. The only operators used are +, -, *, /
3. Opening and closing brackets - () - are used to impact precedence of operations
4. + and - have equal precedence which is less than * and /. * and / also have equal precedence.
5. In two operators of equal precedence give preference to the one on left.
6. All operands are single digit numbers.

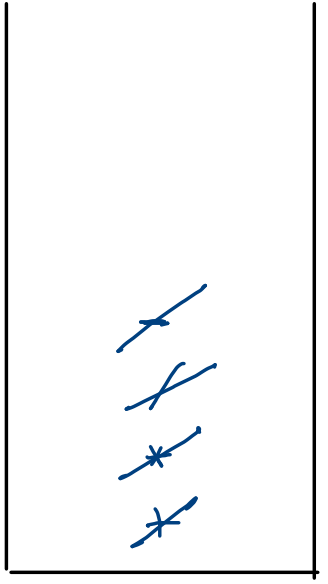
2 + 6 * 4 / 8 - 3



2 + 6 * 4 / 8 - 3

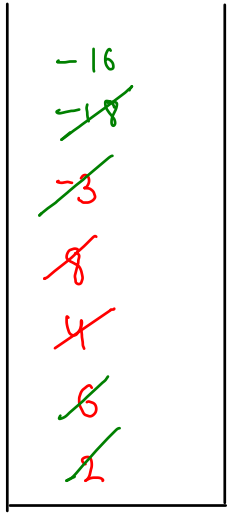


value

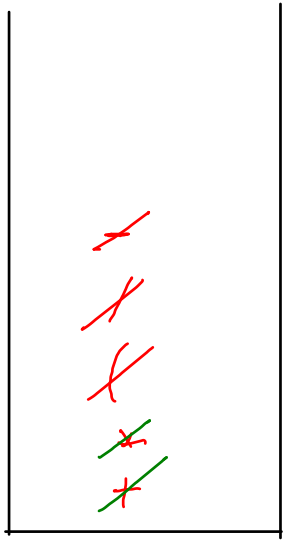


operator

$$2 + 6 * (4 / 8 - 3)$$



VS



OS

operand → push into value stack

(→ push into operator stack

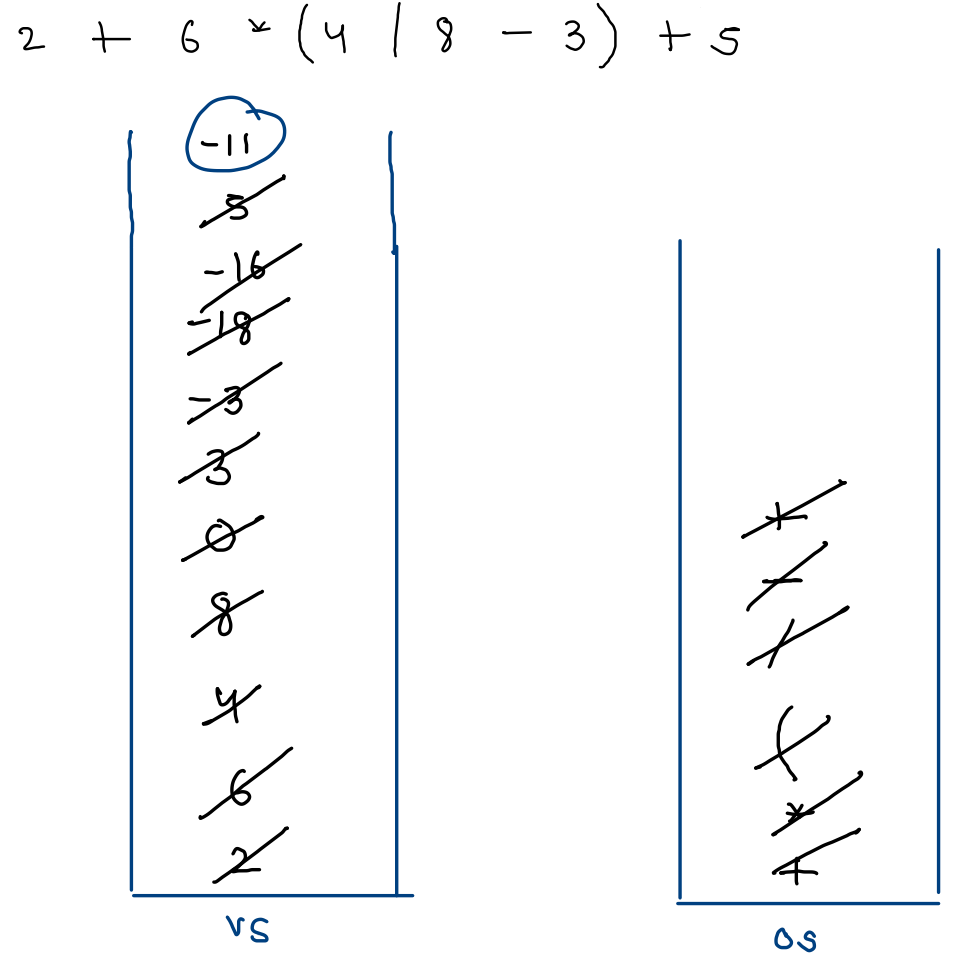
) → evaluate till an '('

operator → first evaluate those operators which have equal or higher priority, then push yourself to stack.

[st.peek() != '(']

```java
for (int i = 0; i < exp.length(); i++) {
    char ch = exp.charAt(i);

    if (ch >= '0' && ch <= '9') {
        //operand, push in value st
        valst.push(ch - '0');
    } else if (ch == '(') {
        //push in operator st
        oprst.push(ch);
    } else if (ch == ')') {
        //evaluate till and opening bracket occurs
        while (oprst.peek() != '(') {
            //evaluate
            int rv = valst.pop();
            int lv = valst.pop();

            char opr = oprst.pop(); //operation
            int val = calculate(lv, rv, opr);

            valst.push(val);

        }

        oprst.pop(); //opening bracket
    } else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
        //operator
        while (oprst.size() > 0 && oprst.peek() != '(' && priority(oprst.peek()) >= priority(ch)) {
            //evaluate
            int rv = valst.pop();
            int lv = valst.pop();

            char opr = oprst.pop(); //operation
            int val = calculate(lv, rv, opr);

            valst.push(val);
        }

        oprst.push(ch);
    }
}

while (oprst.size() > 0) {
    //evaluate
    int rv = valst.pop();
    int lv = valst.pop();

    char opr = oprst.pop(); //operation
    int val = calculate(lv, rv, opr);

    valst.push(val);
}
```

```java
public static int priority(char opr) {
    if (opr == '+' || opr == '-') {
        return 1;
    } else if (opr == '*' || opr == '/') {
        return 2;
    } else {
        return -1;
    }
}

public static int calculate(int v1, int v2, char opr) {
    if (opr == '+') {
        return v1 + v2;
    } else if (opr == '-') {
        return v1 - v2;
    } else if (opr == '*') {
        return v1 * v2;
    } else if (opr == '/') {
        return v1 / v2;
    } else {
        return -1;
    }
}
```



5 + 3 * 4

```
while (oprst.size() > 0) {
    //evaluate
    int rv = valst.pop();
    int lv = valst.pop();

    char opr = oprst.pop(); //operation
    int val = calculate(lv, rv, opr);

    valst.push(val);
}
```