Evaluation of Postfix Expression

I/p: 10 2 * 3 +

O/p: 23

I/p: 10 2 + 3 *

O/p: 36

I/p: 10 2 3 ^ ^

O/p: 100000000

Evaluation of Postfix Expression

I/p: $\underbrace{10 \quad 2 \quad *}_{}\overset{20}{} \quad 3 \quad +$

O/p: 23

$\Rightarrow \boxed{((10 * 2) + 3)}$

$= (20 + 3) = 23$

I/p: $\overset{12}{10 \quad 2 \quad +} \quad 3 \quad *$

O/p: 36

$\Rightarrow ((10+2) * 3)$

$\Rightarrow 12 \times 3 = 36$

I/p: $10 \quad \overset{2^3 \;\; 8}{2 \quad 3 \quad \wedge} \quad \wedge$ $\quad 10^8$

O/p: 100000000

$\Rightarrow 10 \wedge 2 \wedge 3$

Algorithm
to
Evaluate Postfix
Expression

Algorithm
to
Evaluate Postfix
Expression

→ Hint :

① Postfix Expression

not required

→ Precedence ✗
Associativity ✗
Brackets ✗

infix → postfix

Operator

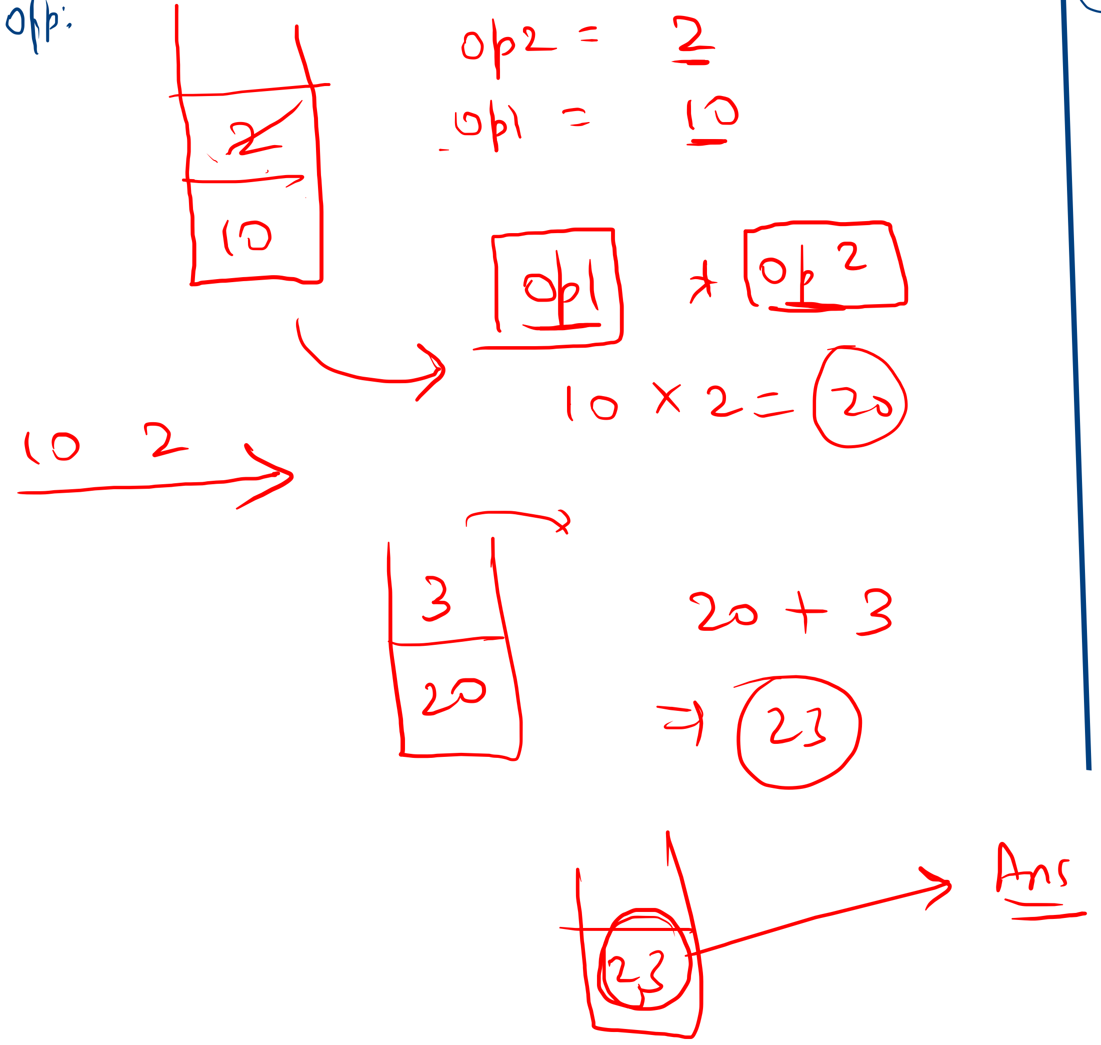
Algorithm to Evaluate Postfix Expression

Operand → Print

```
① create an Empty stack, st
② Traverse through Every symbol x of
   given postfix
   a) if x is a operand, push to st
   b) else ( x is operator)
        → op2 = x.top() ; x.pop();
        → op1 = x.top() ; x.pop();
        → compute $\boxed{op1 \; x \; op2}$ and
          push the result to st.
③ Return st.top();
```

Q. I/p: 10 2 * 3 +

O/p:



1. create an Empty stack, st
2. Traverse through Every symbol x of given postfix
   a) if x is a operand, push to st
   b) **else** (x is operator)
      → op2 = x.top() ; x.pop();
      → op1 = x.top(); x.pop();
      → compute op1 x op2 and push the result to st.
3. Return st.top();

I/P: 10 2 * 3 5 * + 9 - →

O/P:

Input symbol	Stack
10	10
2	2, 10
*	$10 \times 2 = 20$; 20
3	3, 20
5	5, 3, 20
*	$3 \times 5 = 15$; 15, 20
+	
9	9, 35
→	

~~9−35~~ ✗

$35 - 9 \Rightarrow \boxed{26} \Rightarrow$

(3×5)

① create an Empty stack, st

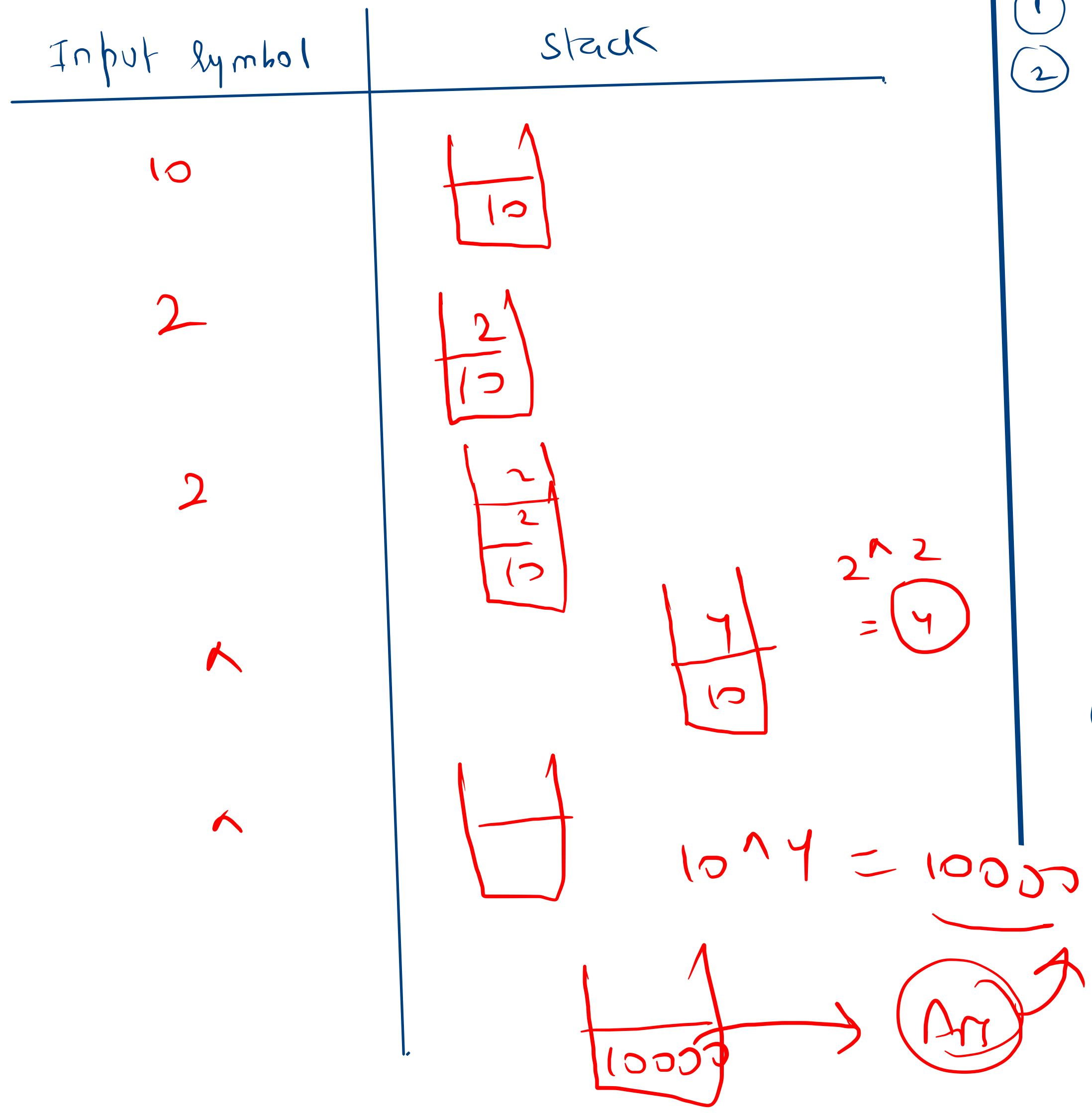
② Traverse through Every symbol x of given postfix

```
a) if x is a operand, push to st
b) else ( x is operator)
    → op2 = st.top(); st.pop();
    → op1 = st.top(); st.pop();
    → compute op1 x op2 and
      push the result to st.
```

③ Return st.top();

I/P: 10 2 2 ^ ^

O/P:



```
(1) create an Empty stack, st
(2) Traverse through Every symbol x of
    given postfix
      a) if x is a operand, push to st
      b) else ( x is operator)
          → op2 = x.top(); x.pop();
          → op1 = x.top(); x.pop();
          → compute op1 x op2 and
            push the result to st.
(3) Return st.top();
```