expression conversion Application of Stack

- Infix
- Prefix
- postfix

Assume only binary operators.

Infix <operand> <operator> <operand>

- Evaluation
- conversion.

$2 + 3 * 5$

$2 + 5 * 3 - 2 * 10 / 5$

Preference rule

1. ( ) , { }, [ ]
2. exponent a^K ← right associative
3. multiplication and division
4. Addition and subtraction.

} ← left associative.

$a * b / c$

Associativity rule

2^{2^2}

a+b*c convert it to postfix.

How?? manually.

a+b*c

= (a+(b*c))

= (a+(bc*))

= (a (bc*) +)

= a b c * +

Prefix

a + b * c

= (a + (b * c))

= (a + (* bc))

= (+ a (* bc))

= + a * bc

Ex^n

$2 + 5 * 4 - 3 + 6 / 2$ convert into postfix.

Ans

$2\ \ 5\ \ 4 * 3 - + 6 / 2 +$

$2\ \ 5\ \ 4 * + 3\ \ 6\ \ 2 / + -$

$(((2 + \underbrace{(5 * 4)}_{a}) - 3) + \underbrace{(6 / 2)}_{b})$

$2 + a - 3 + b$

$(((2 + a) - 3) + b)$

Evaluation of prefix or postfix expression.

a b * c d * + e −

let a = 3, b = 5, c = 2, d = 4, e = 6

3 5 * 2 4 * + 6 −

Scan from left to right.

operator

op1 op2

3 5 * 2 4 * + 6 −

= 15 2 4 * + 6 −

= 15 8 + 6 −

= 23 6 −

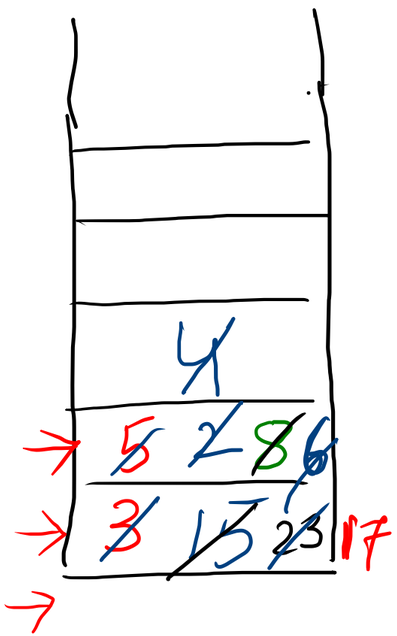
= 17

token — either a number or an operator

op1 operator op2

op2 = pop

op1 = pop



Pseudocode

```
Postfix_evaluation(E)        → expression.
create a stack S
for i = 0 to size(E) - 1
    if E[i] is a operand
        Push(S, E[i])
    else if E[i] is an operator
        op2 = Pop(S)
        op1 = Pop(S)
        result = evaluation(op1, E[i], op2)
        Push(S, result)
return top(S)
```

in postfix.

Ex^n $(((5 * 3) - (2 / 1)) + 6)$

$(5\ 3\ *) - (2\ 1\ /) + 6$

$5\ 3\ *\ 2\ 1\ /\ -\ 6\ +$

Prefix evaluation

$$(((3 * 5) + (2 * 4)) - 6)$$

$$= ((* 35) + (* 24)) - 6$$

$$= + * 35 * 24 - 6$$

$$= - + * 35 * 24 6$$

H.W.