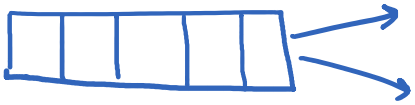
Stack

Friday, October 1, 2021 2:49 PM

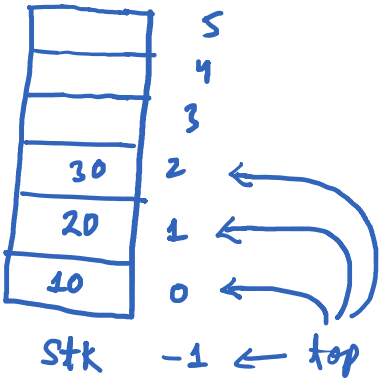
Stack :-

Linear Data Structure where both insertion & deletion are performed at the same end.



Used to eliminate recursion

top



Push (10)
Push (20)
Push (30)
Push (40)
Push (50)
Push (60)

Pop ()
Pop ()
Pop ()

```
Push (x)
if (top == n-1)
    print " Stack is full. Overflow"
    exit
Stk[++top] = x ;
```

```
Pop()
if(top == -1)
    print " Stack is empty. Underflow"
    exit
x = Stk[top];
top--;
return x;
```

return Stk[top--];

Applications:-

Conversion of Notation Standards :—

Infix expression — opnd 1 opr opnd 2 A + B

Infix expression — opnd 1 opr opnd 2 — A + B

Postfix expression — opnd 1 opnd 2 opr — A B +

Prefix expression — opr opnd 1 opnd 2 — + A B

$10 \times 6 - 4$

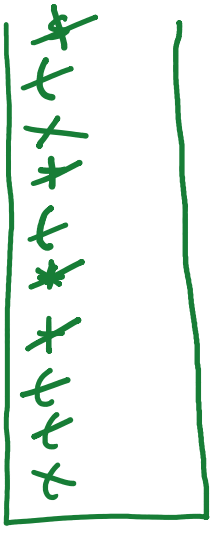
10, 6, x, 4, −

1) Infix to Postfix

Fully parenthesized expression

$(((A+B) * (C+D)) / (E \$ F))$

Exponent - ^, ↑, **, $



Postfix Expression

AB+CD+*EF$/

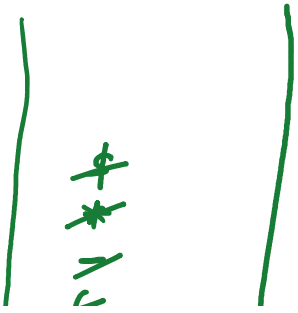
(,), opnd, opr

↑ push ↑ ↑ display ↑ push

Pop & display all the operators until we encounter an opening bracket

Partially Parenthesized Expression

$A + B \$ (C - D * E) \$ F$



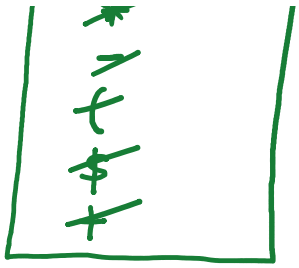
If current ptr is having higher precedence than stk ptr then push else pop till current ptr is not having

Precedence List

A $ B $ C — $ exponent

*, /

A + D + C — +, −



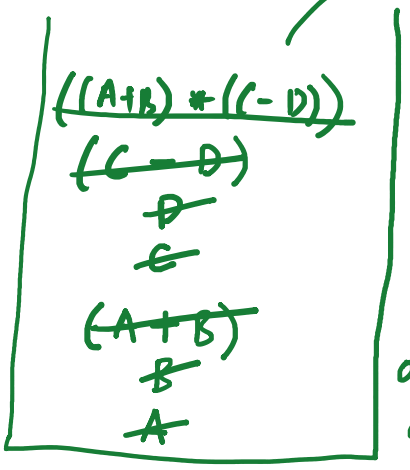
else pop till current ptr is not having higher precedence

A+B+C +, −

A B C D E * − F $ $ +

2) Postfix to Infix

A B + C D − * ⟶ ((A+B) * (C−D))



opnd → Push

opr → Pop 2 then Push (opnd1 opr opnd2)

3) Infix to prefix // Prefix to Infix

((A+B) * (C−D)) —Postfix→ AB+CD− *

Prefix ↓

* +AB − CD

((A+B) * (C−D)) —Reverse it→))D−C(*)B+A((—Replace '(' with ')' & vice-versa→ ((D−C) * (B+A))

↑ Convert to

Convert to Postfix

Reverse it

+AB-CD ⟷ DC-BA+

5) Postfix to Prefix

AB+CD-* ⟶ *+AB-CD

AB+ CD- *

(+AB) CD- *

(+AB) (-CD) *

*+AB-CD

*+AB-CD

~~-CD~~

~~D~~

~~C~~

~~+AB~~

~~B~~ opnd2

~~A~~ opnd1

opr opnd1 opnd2

6) Prefix to Postfix

+ AB-CD ⟶ (Reverse it) DC-BA+ ⟶ (Convert like in 5) *-DC+BA

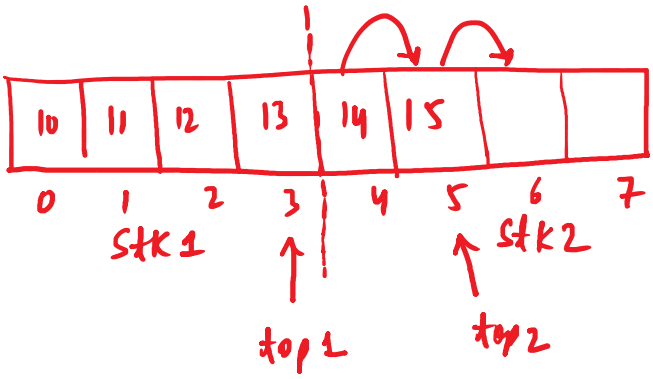
Reverse it

AB+CD-*

Representation of 2 stacks in same array:-

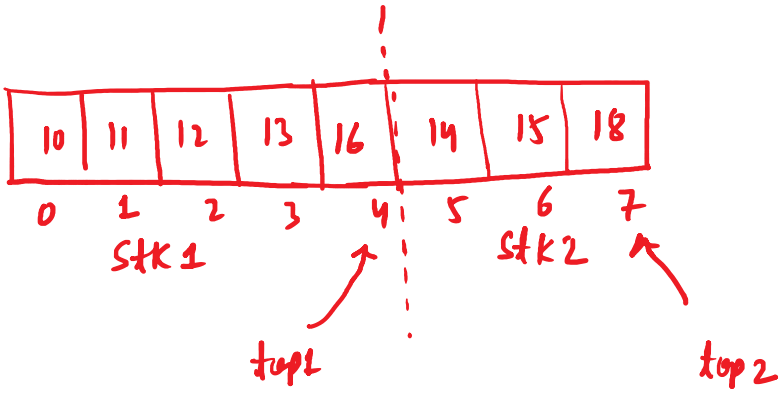
Initial Situation

Push (stk 1, 12)



Push (stk 1, 16)

Local Overflow

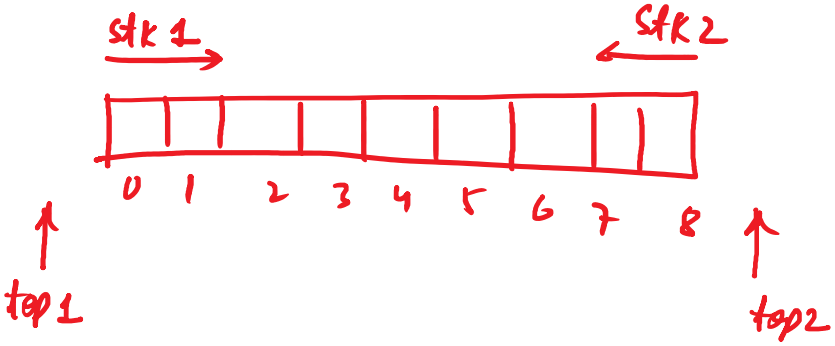


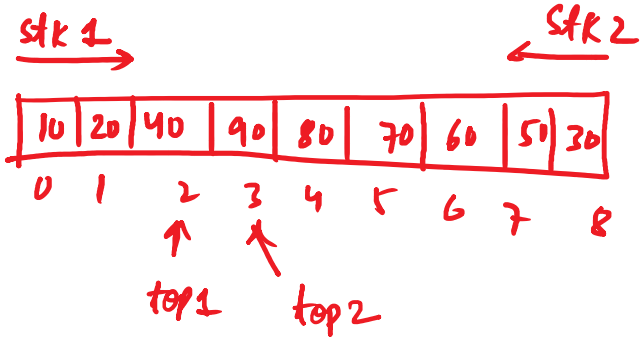
Push (stk 2, 18)

Push (stk 1, 19)

Global Overflow

Better Approach :-





Push (stk 1, 10)

Push (stk 1, 20)

Push (stk 2, 30)

Push (stk 1, 40)

Push (stk 2, 50)

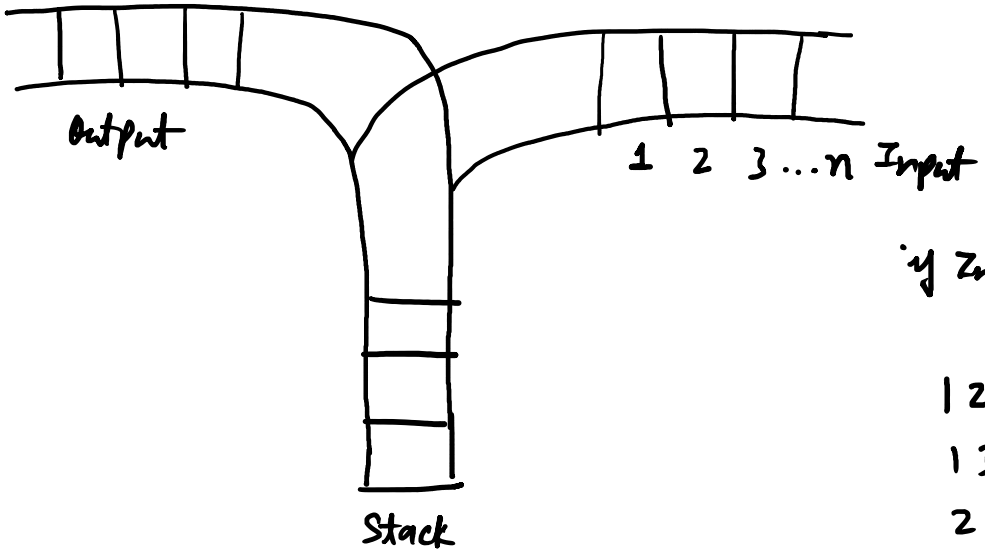
if (top1 == top2 - 1)
"Global Overflow"

Push(stk2, 80)
Push(stk2, 90)

Push(stk2, 50)
Push(stk2, 60)
Push(stk2, 70)

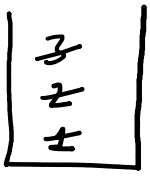
n-stacks are to be stored in same array

Re-packaging algorithm top & base pointers



if Input = 1 2 3

123 ✓
132 ✓
213 ✓
231 ✓
312 ×
321 ✓



1 3 2

3

Try with n=4.