COP3502 2/9/22

① Andres - meet with him!

② Stacks!

Stack - Abstract Data Type

Specifies behaviors but now how it's stored. (Reg Data type specifies storage) array, linked list, etc.

Operations / Behaviors

Goal in constant time

1) push(item) - place item on top
2) item pop() - removes + returns item from top

Trace

push(3)
push(7)
x = pop()
push(9)
push(6)
y = pop()

y [6]
x [7]

~~6~~
~~7~~ 9
3

Two Sample Algs use Stack

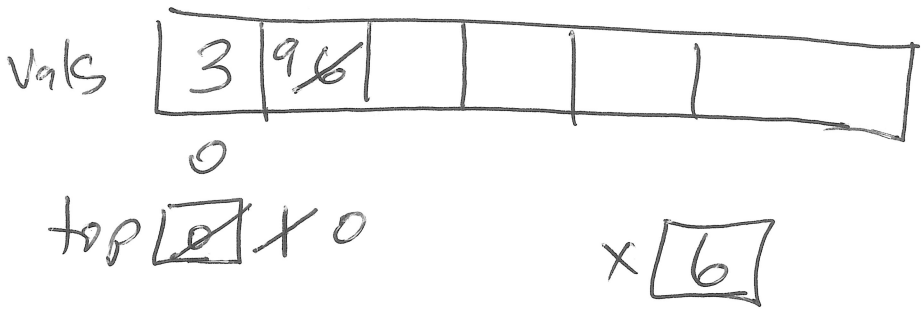
#2 (1) Evaluating Postfix Expression

#4 (2) Converting Infix Expr ⇒ Postfix

Two Implementations of Stack

#1 (1) Array based

#3 (2) Linked List based



x 6

push(3)
push(6)
x = pop()
always added to index top+1
return what's in index top
subtract 1 from top
push(9)

Auxiliary Functions

top() - returns item at top w/o popping

empty() - returns 1 iff stack is empty now

full() - returns 1 iff stack is full

Better Array Stack - when array fills up realloc to twice the size.

Linked List Implementation Stack

push(6)
push(7)
X = pop()
push(4)

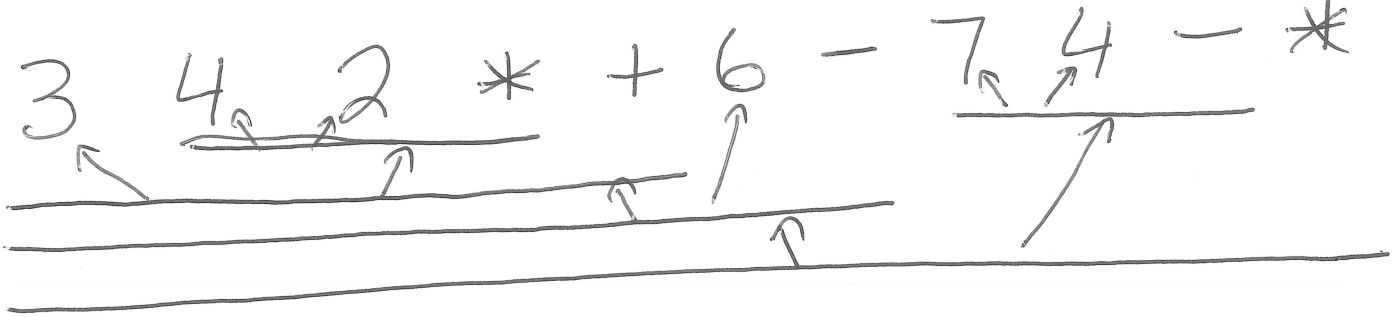
top → [6] , push 7

top → [7] → [6 x]

PUSH - insert front (const time)

POP - delete front

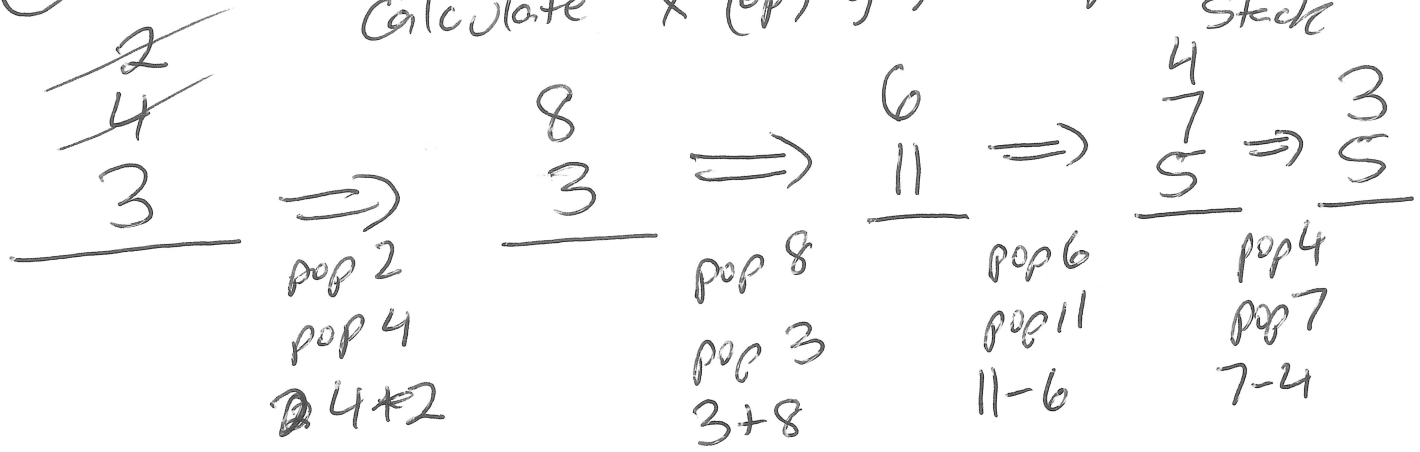
Evaluating Postfix Expressions



Read symbols L → R.

(1) If you get operand, push onto stack

(2) If you get an operator pop off y, then x
Calculate x (op) y, and push onto stack



⇒ pop 3, pop 5, 5*3 ⇒ **15** → value of expression

If stack empty in step 2 ⇒ Invalid Expr.

If at end size(stack) > 1 ⇒ Invalid Expr.