COP 3502 2/24/26

3/3 Quiz P0,1
3/10 Quiz P2,3
4/21 Quiz P4,5

} Quiz Dates for ~~for~~ Quizzes on Prog. Asgn.

Today's Topic:

Stacks

- Abstract Data Structure
- operations
- LL implementation
- array implementation
- Application #1: Evaluating Postfix
- Application #2: Converting Infix → Postfix

Abstract Data Structure

No requirements about HOW to store data
Just functions that must be provided.

- push (places item on top)
- pop (removes & returns item on top)
- empty (returns # elems in stack)
- size (true (1) iff 0 elems stack)
- top returns top elem w/o removing

Goal:

Show 2 separate implementations run each function O(1) time. (Amortized time)

LL

push(3)	2	
push(7)	7	7
push(2)	3	3
x = pop()		x 2

→ insert to front

top → x

top → [3|x]

top → [7|→] → [3|x]

top → [2|→] → [7|→] → [3|x]

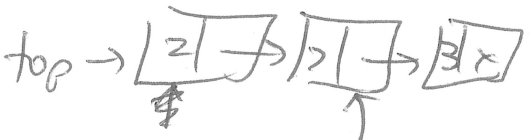
top → [7|→] → [3|x]

x [2]

insertFront → O(1)

pop → deleteFront → O(1)

Sketch Del Front



rest

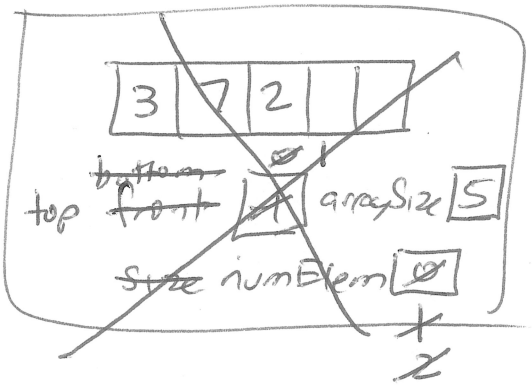
retval 2

2 SOLUTIONS TO MOVING TOP:

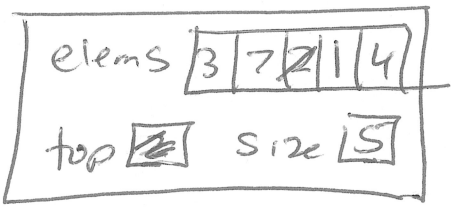
1) ** (2 stks)

2) create separate stack stack

Array Implementation Stack



8



1
2
3
4

push(3)
push(7)
push(2)
pop()
push(8)
push(1)
push(4)

```
if (sPtr->top+1 == sPtr->size){

  sPtr->elems =
  realloc(sPtr->elems, 2*sPtr->size*
                       sizeof(int));

  sPtr->size *= 2;
              ↖ mult
}
sPtr->elems[sPtr->top+1] = val;
sPtr->top++;
```

Evaluate a Postfix Expression

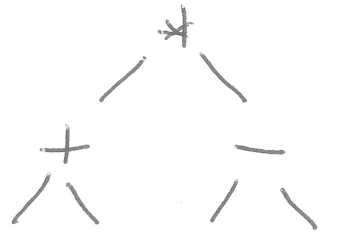
2 7 + 6 4 − *

9 2

18

in infix

(2+7) * (6-4)



Read L → R

1. If you get an operand (num)
   Push onto Stack

2. If you get an operator (+, −, *, /)
   Pop off the top 2 items in stack,
   call these b (top) and a, respectively (2nd from top)
   Push value a op b onto stack.

~~7~~	~~6~~	~~2~~	
~~2~~	9	~~9~~	18

2 + 7 = 9
6 − 4 = 2
9 * 2 = 18

2 7 + 9 +

To detect invalid expr

1. If try to pop empty stack → INVALID
2. If at end stack size > 1.

7 6 3 + * 2 3 − 1 + + 7 /

3			3	1			
6	9		2	-1	0	7	(9)
7	7	63	63	63	63	63	
	+	*		−	+	+	/
	6+3	7*9		2-3	-1+1	63+0	63/7

Infix to Postfix

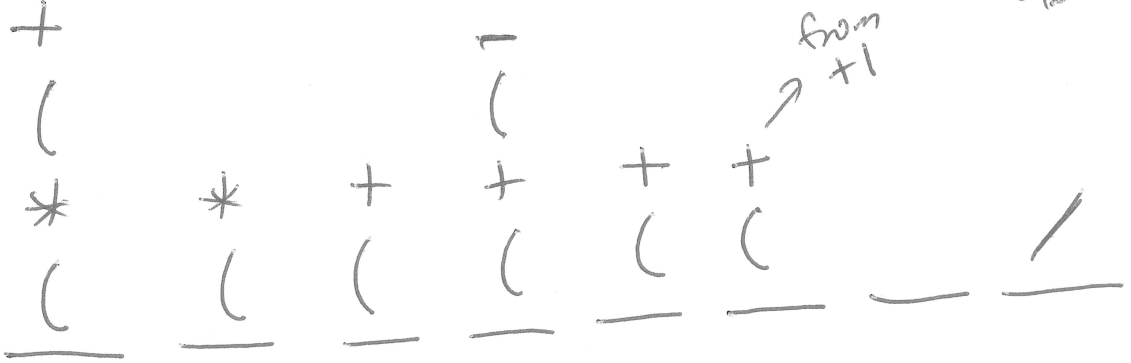
Eval Postfix: Operand Stack (nums)
Returns Number

Infix → Postfix: Operator Stack (+, /, *, -, ( ) also open parens
Returns an expression

(7 * (6 + 3) + (2 - 3) + 1) / 7

Read left to right:

1. If we get open parenthesis, push stack
2. If we get operand, add to end expression.
3. If we get an operator, pop off the stack all operators of equal or greater precedence than the current operator + place in the expression. Stop if you hit: (a) operator of lower precedence (b) empty stack, OR (c) open parenthesis.
   Push operator onto stack
4. If we got CLOSE PAREN, pop items off stack into expr until open.



Expr: 7 6 3 + * 2 3 - + 1 + 7 /

After loop is done, pop off all remaining operators from stack + put at the end of the expression