COP 3502 9/22/2025

2 Stack Applications

1. Evaluating Postfix Expressions
2. Converting Infix to Postfix

3 5 + (3+5 = 8)

3 8 2 / + 8 5 2 * - +

Read left to right, have a operand stack (numbers)

① If you get a # (operand) push it onto stack

② If you get an operator (op), pop off 2 items, call them y and x, respectively. Calculate x op y and push onto stack.

When you finish, if there's 1 # left on stack AND you never tried to pop an empty stack, that # is what the expression evaluates to

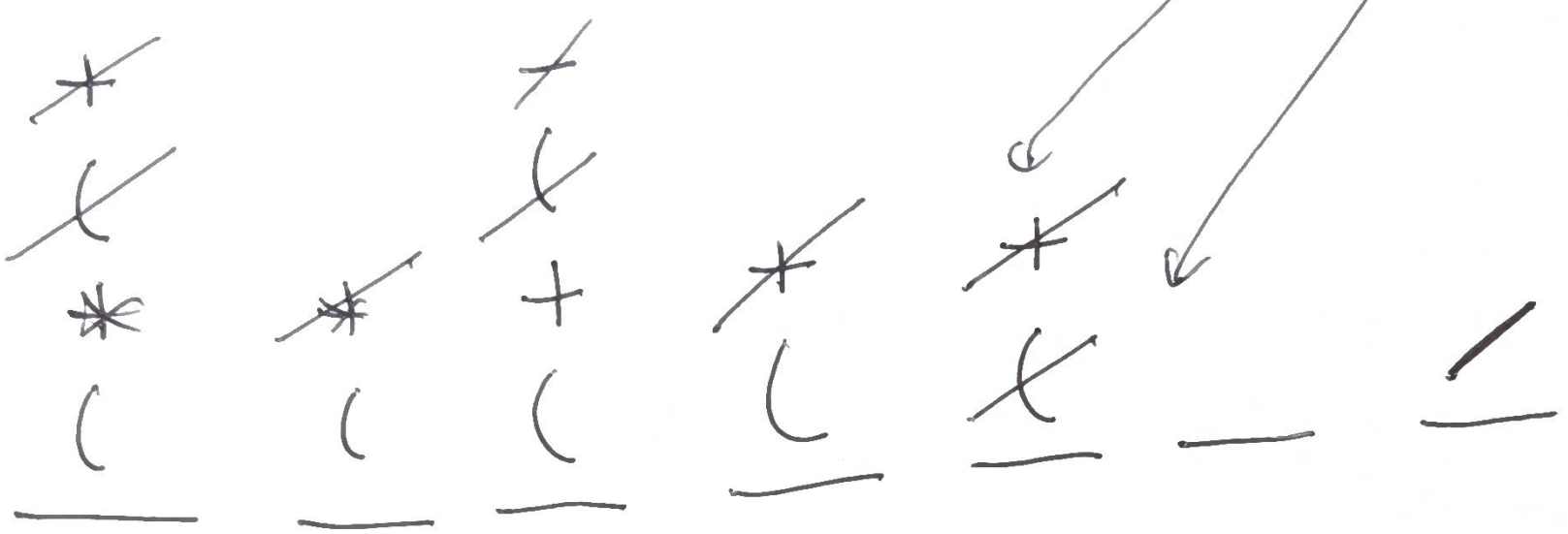
3	8	2	/	+	8	5	2	*	-	+
	x	y			x	y	5	10		
8	8	/ 2	4		3	+ 4	8	8	-2	
3	→ 4		3		= 7		7	7	7	5
stack			stack				stack	stack	stack	stack

(5*2), (8-10), (7+-2)

Infix to Postfix

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

Read left to right, OPERATOR STACK



Output: 7 6 3 + * 2 3 - + 1 + 7

Rules ① If open paren, push onto stack

② if a operand/number the output into expression.

③ if get operator, pop off each operator of equal or greater precedence, stopping only when you reach an operator of lower precedence, OR an open paren or empty stack.

④ if get ')' (close paren), pop off each item from stack + place it into the expression stopping when you hit matching open paren. first

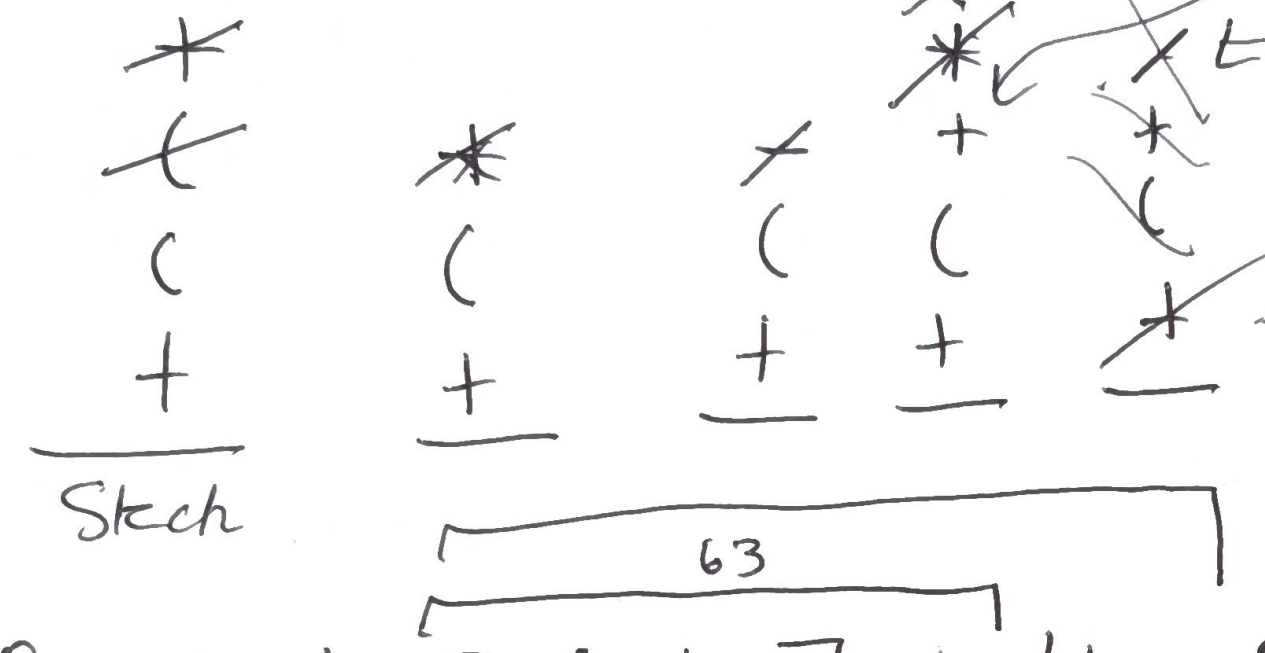
⑤ If stack is non-empty @ end, pop off each item and place into expression

Invalid Expressions

+ 2 3 4 -

2 3 + * 6

4 + ((3 + 6) * 7 − 4 + 2 * (6 − 3) / 2)



Output: 4 3 6 + 7 * 4 − 2 6 3 − * 2 / + +

63

3

4 59 3 + +

66