Example3: conversion of an infix expression to a postfix expression

October 28, 2018

Source:https://faculty.utrgv.edu/john.abraham/6314/assignments/postfix%20tutorial.pdf

Extra practice: after you finish reading Example3 (next page) try to convert the following infix expression to postfix.

Remember to check the algorithm used to convert an infix expression to a postfix expression from the slides of ADT Stack $->$ Operations.

Infix: 4 + (9 -(3 * 2)) % 3 + 5 * (2 +(6 / 3)) -1

Operator Precedence (from highest to lowest priority):

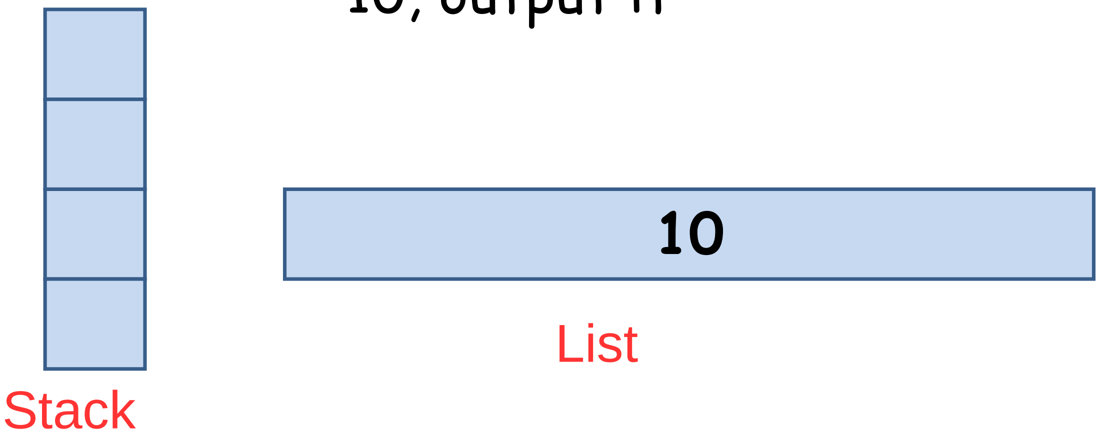
- $*/\%$
- $+-$

Solution: 4 9 3 2 * - 3 % + 5 2 6 3 / + * + 1 -

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- We see the first number 10, output it



10

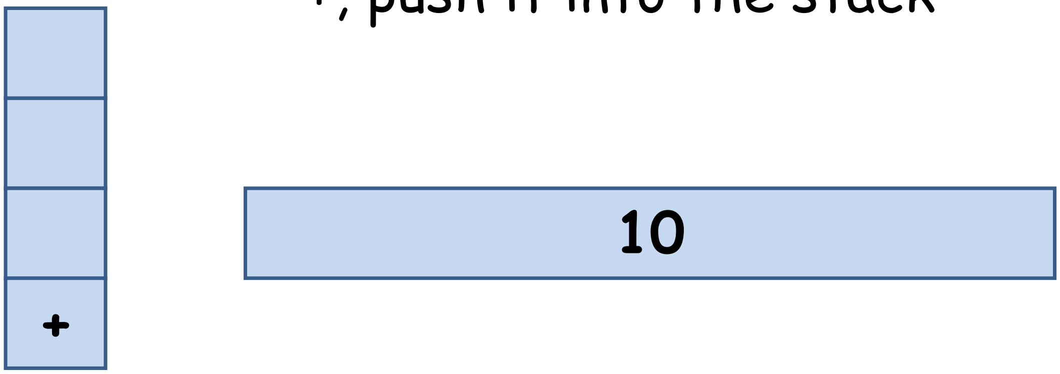
List

Stack

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- We see the first operator +, push it into the stack



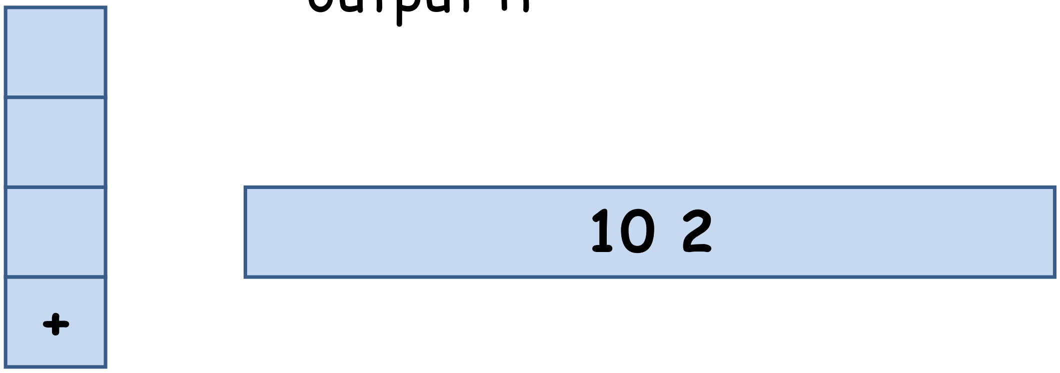
10

+

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- We see the number 2, output it



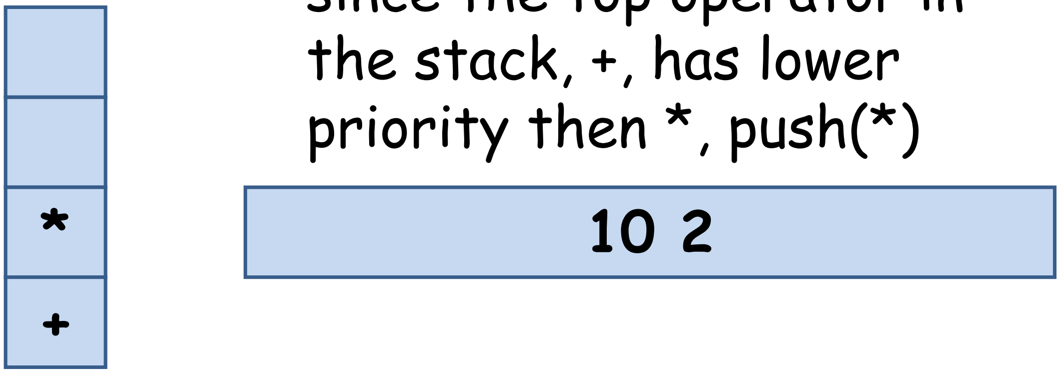
+

10 2

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- We see the operator *, since the top operator in the stack, +, has lower priority then *, push(*)



*

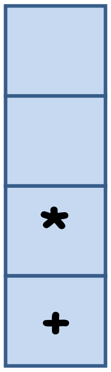
+

10 2

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- We see the number 8, output it

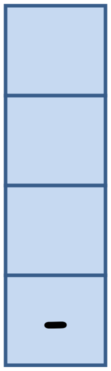


10 2 8

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- We see the operator -, because its priority is lower then *, we pop. Also, because + is on the left of it, we pop +, too. Then we push(-)

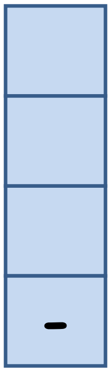


10 2 8 * +

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- We see the number 3, output it

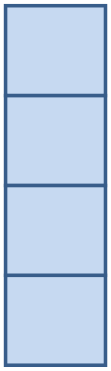


10 2 8 * + 3

Transform Infix to Postfix

Ex: 10 + 2 * 8 - 3

- Because the expression is ended, we pop all the operators in the stack



10 2 8 * + 3 -