Transform Infix to Postfix

Transform Infix to Postfix

$$\boxed{a \;/\; b \;-\; c \;+\; d \;*\; e \;-\; a \;*\; c}$$

$$\Downarrow$$

$$\boxed{a \; b \;/\; c \;-\; d \; e \;*\; + \; a \; c \;*\; -}$$

$$\boxed{a * (b + c) * d}$$

$$\Downarrow$$

$$\boxed{\qquad\qquad\qquad\qquad}$$

Translation from Infix to Postfix(1)

- The order of operands is the same in infix and postfix
- During scanning the infix expression left-to-right,
  - operands are passed to the output expression as they are encountered
  - stack operators as long as the precedence of the operator at the top is less than the precedence of the incoming operator
  - If the operator with higher or equal precedence is on the top of the stack, it is removed first
- Unstack when reaching eos(end of string)

Exercise

6 / 2 – 3 + 4 * 2 $\Rightarrow$ 6 2 / 3 – 4 2 * +

Token	Stack			Top	
	[0]	[1]	[2]		
6				– 1	6
/	/			0	6
2	/			0	6 2
–	–			0	6 2 /
3	–			0	6 2 / 3
+	+			0	6 2 / 3 –
4	+			0	6 2 / 3 – 4
*	+	*		1	6 2 / 3 – 4
2	+	*		1	6 2 / 3 – 4 2
eos				– 1	6 2 / 3 – 4 2 * +

Translation from Infix to Postfix(2)

- When meeting the left parenthesis, put it on the stack always
- When meeting the right parenthesis, unstack until reaching the corresponding left parenthesis

Exercise – program 3.11: postfix

- **a * (b + c) * d**

Token	Stack			Top	Output
	[0]	[1]	[2]		
a				– 1	a
*	*			0	a
(	*	(		1	a
b	*	(		1	a b
+	*	(	+	2	a b c
c	*	(	+	2	a b c
)	*			0	a b c +
*	*			0	a b c + *
d	*			0	a b c + * d
eos				– 1	a b c + * d *

precedence of operators

- an example of precedence values

	(	)	+	-	*	/	%	eos
in-stack	?	9	2	2	4	4	4	0
incoming	?	9	2	2	4	4	4	0