Vidyalekhan
DATE 12.9.22
PAGE

TUTORIAL - 4

INFIX, PREFIX, POST-FIX

PART I

A.2

a) Infix : A + B * C + D

Scanned	Operator Stack	Expression
	(	
A	(	A
+	(+	A
B	(+	AB
*	(+ *	AB
C	(+ *	ABC
+	(+	ABC*+
D	(+	ABC*+D
	POSTFIX	ABC*+D+

b) Infix : (A+B) * (C+D)

Scanned	Operator Stack	Expression
	(	
(	((	
A	((	A
+	((+	A
B	((+	AB
)	(	AB+
*	(*	AB+
(	(*(	AB+
C	(*(	AB+C
+	(*(+	AB+C
D	(*(+	AB+CD
)	(*	AB+CD+
	POSTFIX	AB+CD+*

(c) Infix : A * B + (C / D) / E - F

Scanned	Operator Stack	Expression
	(	
A	(	A
*	(*	A
B	(*	AB
+	(+	AB*
(	(+ (	AB*
C	(+ (	AB*C
)	(+	AB*C
/	(+ /	AB*C
D	(+ /	AB*CD
)	(+	AB*CD/
/	(+ /	AB*CD/
E	(+ /	AB*CD/E
-	(-	AB*CD/E/+
F	(-	AB*CD/E/+F
	POSTFIX	AB*CD/E/+F-

d) Infix : X ^ Y / (5 * Z) + 2

Scanned	Operator Stack	Expression
	(	
X	(	X
^	(^	X
Y	(^	XY
/	(^ /	XY
(	(^ / (	XY
5	(^ / (	XY5
*	(^ / (*	XY5
Z	(^ / (*	XY5Z
)	(^ /	XY5Z*
+	(+	XY5Z*/^
2	(+	XY5Z*/^2
		XY5Z*/^2+

e) Infix : A + (B*C - (D/E^F) * G) * H

Scanned	Operator Stack	Expression
	(	
A	(	A
+	(+	A
(	(+(	A
B	(+(	AB
*	(+(*	AB
C	(+(*	ABC
-	(+(*-	ABC
(	(+(*-(	ABC
D	(+(*-(	ABCD
/	(+(*-(/	ABCD
E	(+(*-(/	ABCDE
^	(+(*-(^	ABCDE/
F	(+(*-(^	ABCDE/F
)	(+(*-	ABCDE/F^
*	(+(*	ABCDE/F^-*
G	(+(*	ABCDE/F^-*G
)	(+	ABCDE/F^-*G*
*	(+*	ABCDE/F^-*G*
H	(+*	ABCDE/F^-*G*
		POSTFIX ABCDE/F^-*G**+

g) Infix : (A + B* (C-D)) / E

Scanned	Operator Stack	Expression
	(	
(	((	
A	((	A
+	((+	A
B	((+	AB
*	((+*	AB
(	((+*(	AB

C	((+*(	ABC
-	((+*(-	ABC
D	((+*(-	ABCD
)	((+*E	ABCD-
)	(	ABCD-*+
/	(/	ABCD-*+
E	(/	ABCD-*+E
		ABCD-*+E/

f) Infix : K+L-M*N+(O^P)*W/U/V*T+Q.

Scanned	Operator Stack	Expression
	(	
K	(	K
+	(+	K
L	(+	KL
-	(-	KL+
M	(-	KL+M
*	(-*	KL+M
N	(-*	KL+MN
+	(+	KL+MN*-
(	(+(	KL+MN*-
O	(+(	KL+MN*-O
^	(+(^	KL+MN*-O
P	(+(^	KL+MN*-OP
)	(+	KL+MN*-OP^
*	(+*	KL+MN*-OP^
W	(+*	KL+MN*-OP^W
/	(/	KL+MN*-OP^W*+
U	(/	KL+MN*-OP^W*+U
/	(//	KL+MN*-OP^W*+U
V	(//	KL+MN*-OP^W*+UV
*	(*	KL+MN*-OP^W*+UV//
T	(*	KL+MN*-OP^W*+UV//T
+	(*+	KL+MN*-OP^W*+UV//T*

Vidyalekhan

DATE
PAGE

Q (+ KL+MN*/OP^W*+UV//T*Q

POSTFIX KL+MN*/OP^W*+UV//T*Q+

A.1 Infix to Postfix -

1. Initialize a stack for operators, output list
2. Split the input into a list of tokens.
3. Repeat step 4 to 9 for each token (left to right).
4. if it is operand : append to output
5. if it is '(' : push into operators' stack
6. if it is in '+ - * /':
7. While stack[top] has precedence >= it
8. pop & append to the output list
9. Push onto stack
10. Pop and append the rest of the stack.

Infix to Prefix -

1. Reverse the infix expression.
2. Initialize a stack for operators, output list.
3. Split the input into a list of tokens.
4. Repeat step 5 to 10 for each token (left to right)
5. if it is operand : append into operators stack
6. if it is '(' : push into operators' stack
7. if it is in '+ - * /' :
8. while stack[top] has precendence >= it
9. pop & append to the output list.
10. Push into stack.
11. Pop and append the rest of the stack.

PART II

A.2 a) Infix : A + B * C + D Reverse : D + C * B + A

Scanned	Operator Stack	Expression
	(	
A	C	A D
+	(+	A D
B	(+	D C
*	(+*	D C
B	(+*	D C B
+	(+	D C B * +
A	(+	D C B * + A
		D C B * + A +

∴ Prefix : + A + * B C D

b) Infix : (A+B) * (C+D) Reverse :) D + C (*) B + A (

-> (D + C) * (B + A)

Scanned	Operator Stack	Expression
(	((	
D	((	D
+	((+	D
C	((+	D C
)	(	D C +
*	(*	D C +
(	(*(	D C +
B	(*(	D C + B
+	(*(+	D C + B
A	(*(+	D C + B A
)	(*	D C + B A +
		D C + B A + *

Prefix : * + A B + C D

c) Infix: A*B + (C/D)/E - F

Reverse: F - E / (D / C) + B * A

Scanned	Operator Stack	Expression
	(	
F	(	F
-	(-	F
E	(-	FE
/	(-/	FE
(	(-/(	FE
D	(-/(	FED
/	(-/(/	FED
C	(-/(/	FEDC
)	(-/	FEDC/
+	(+	FEDC//-
B	(+	FEDC//-B
*	(+*	FEDC//-B
A	(+*	FEDC//-BA
		FEDC//-BA*+

∴ Prefix = +*AB-//CDEF

d) Infix: X^Y/(5*Z)+2 Reverse: 2+(Z*5)/Y^X

Scanned	Operator Stack	Expression
	(	
2	(	2
+	(+	2
(	(+(	2
Z	(+(	2Z
*	(+(*	2Z
5	(+(*	2Z5
)	(+	2Z5*
/	(+/	2Z5*
Y	(+/	2Z5*Y
^	(+/^	2Z5*Y
X	(+/^	2Z5*Y/+X
		2Z5*Y/+X^

∴ Prefix: ^X+/Y*5Z2

(c) Infix : A + (B * C - (D / E ^ F) * G) * H

Reverse : H * (G * (F ^ E / D) - C * B) + A

Scanned	Operator Stack	Expression
	(	
H	(	H
*	(*	H
(	(*(	H
G	(*(	HG
*	(*(*	HG
(	(*(*(	HG
F	(*(*(	HGF
^	(*(*(^	HGF
E	(*(*(^	HGFE
/	(*(*(^/	HGFE
D	(*(*(^/	HGFED
)	(*(*	HGFED/^
-	(*(*-	HGFED/^*
C	(*(-	HGFED/^*C
*	(*(-*	HGFED/^*C
B	(*(-*	HGFED/^*CB
)	(*	HGFED/^*CB*-
+	(+	HGFED/^*CB*-*
A	(+	HGFED/^*CB*-*A
		HGFED/^*CB*-*A+

Prefix = + A * - * B C * ^ / D E F G H

f. Infix : K+L-M*N+(O^P)*W/U/V*T+Q
Reverse : Q+T*V/U/W*(P^O)+N*M-L+K

Scanned	Operator Stack	Expression
	(	
Q	(	Q
+	(+	Q
T	(+	QT
*	(+*	QT
V	(+*	QTV
/	(+*/	QTV*+
U	(+/	QTV*+U
/	(+//	QTV*+U
W	(+//	QTV*+UW
*	(+*	QTV*+UW//
(	(+*(	QTV*+UW//
P	(+*(	QTV*+UW//P
^	(+*(^	QTV*+UW//P^
O	(+*(^	QTV*+UW//PO
)	(+*	QTV*+UW//PO^

5.3 What is the result of evaluating the following expression? (Infix to Postfix and Infix to prefix)

a) 10 + 3 * 5 / (16 - 4)

→ 10, 3, 5, *, 16, 4, -, /, +
10, 15, 12, /, +
10, 1.25 +
= 11.25

→ (4 - 16) / 5 * 3 + 10
4, 16, -, 5, /, 3, *, 10, +
=> +, 10, *, 3, /, 5, -, 16, 4
=> +, 10, *, 3, /, 5, 12 => +, 10, *, 1.25
=> 11.25

b) (5+3) * (8-2)

→ 5,3,+, 8, 2,-, * → Postfix

⇒ 8, 6, *

⇒ 48

→ (2-8) * (3+5)

⇒ 2,8,-, 3, 5,+, *

⇒ *, +, 5, 3, -, 8, 2 → Prefix

⇒ *, 8, 6

⇒ 48