GIT Department of Computer Engineering

CSE 222/505 – Spring 2020

Homework #04 Part 1 Report

Abdullah ÇELİK

171044002

Note : Right side was considered top of stack.

i) A + ((B – C * D) / E) + F – G / H

a) Convert infix to prefix

Infix to Prefix Input String : A + ((B - C * D) / E) + F - G / H			
Character	**Operations**	**Prefix**	**Operator Stack**
H	insert H to index 0 of prefix	H	
/	push / to stack	H	/
G	insert G to index 0 of prefix	GH	/
-	pop from stack and insert it to index 0 of prefix push - to stack	/GH	-
F	insert F to index 0 of prefix	F/GH	-
+	pop from stack and insert it to index 0 of prefix push + to stack	-F/GH	+
)	push) to stack	-F/GH	+)
E	insert E to index 0 of prefix	E-F/GH	+)
/	push / to stack	E-F/GH	+)/
)	push) to stack	E-F/GH	+)/)
D	insert D to index 0 of prefix	DE-F/GH	+)/)
*	push * to stack	DE-F/GH	+)/)*
C	insert C to index 0 of prefix	CDE-F/GH	+)/)*
-	pop from stack and insert it to index 0 of prefix push - to stack	*CDE-F/GH	+)/)-
B	insert B to index 0 of prefix	B*CDE-F/GH	+)/)-
(	pop from stack and insert it to index 0 of prefix pop from stack	-B*CDE-F/GH	+)/
(	pop from stack and insert it to index 0 of prefix pop from stack	/-B*CDE-F/GH	+
+	pop from stack and insert it to index 0 of prefix push '+' stack	+/-B*CDE-F/GH	+
A	insert A to index 0 of prefix	A+/-B*CDE-F/GH	+
End of string	pop from stack and insert it to index 0 of prefix	+A+/-B*CDE-F/GH	

Conclusion :

Infix : A + ((B – C * D) / E) + F – G / H

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

b) Evaluation Of Prefix

Evaluation of Prefix Prefix Expression : : + A + / - B * C D E - F / G H Example : + 2 + / - 8 * 2 3 2 - 8 / 10 5		
Character	**Operations**	**Stack**
5	push 5 to stack	5
10	push 10 to stack	5 10
/	pop from stack pop from stack calculate 10 / 5 and push result to stack	2
8	push 8 to stack	2 8
-	pop from stack pop from stack calculate 8 - 2 and push result to stack	6
2	push 2 to stack	6 2
3	push 3 to stack	6 2 3
2	push 2 to stack	6 2 3 2
*	pop from stack pop from stack calculate 2 * 3 and push result to stack	6 2 6
8	push 8 to stack	6 2 6 8
-	pop from stack pop from stack calculate 8 - 6 and push result to stack	6 2 2
/	pop from stack pop from stack calculate 2 / 2 and push result to stack	6 1
+	pop from stack pop from stack calculate 1 + 6 and push result to stack	7
2	push 2 to stack	7 2
+	pop from stack pop from stack calculate 2 + 7 and push result to stack	9

Conclusion :

Infix : A + ((B – C * D) / E) + F – G / H	= 2 + ((8 – 2 * 3) / 2) + 8 – 10 / 5	= 9
Prefix : + A + / - B * C D E - F / G H	= + 2 + / - 8 * 2 3 2 – 8 / 10 5	= 9

c) Convert infix to postfix

Infix to Postfix Input String : A + ((B - C * D) / E) + F - G / H			
Character	**Operations**	**Postfix**	**Operator Stack**
A	append A to postfix	A	
+	push + to stack	A	+
(	push (to stack	A	+(
(	push (to stack	A	+((
B	append B to postfix	AB	+((
-	push - to stack	AB	+((-
C	append C to postfix	ABC	+((-
*	push * to stack	ABC	+((-*
D	append D to postfix	ABCD	+((-*
)	pop from stack and append it to postfix pop from stack and append it to postfix pop from stack	ABCD*-	+(
/	push / to stack	ABCD*-	+(/
E	append E to postfix	ABCD*-E	+(/
)	pop from stack and append it to postfix pop from stack	ABCD*-E/	+
+	pop from stack and append it to postfix push + to stack	ABCD*-E/+	+
F	append F to postfix	ABCD*-E/+F	+
-	pop from stack and append it to postfix push - to stack	ABCD*-E/+F+	-
G	append G to postfix	ABCD*-E/+F+G	-
/	push / to stack	ABCD*-E/+F+G	-/
H	append H to postfix	ABCD*-E/+F+GH	-/
End of string	pop from stack and append it to postfix pop from stack and append it to postfix	ABCD*-E/+F+GH/-	

Conclusion :

Infix : A + ((B – C * D) / E) + F – G / H

Postfix : A B C D * - E / + F + G H / -

d) Evaluation of Postfix

Evaluation of Postfix **Postfix Expression : A B C D * - E / + F + G H / -** **Example : 2 8 2 3 * - 2 / + 8 + 10 5 / -**		
Character	**Operations**	**Stack**
2	push 2 to stack	2
8	push 8 to stack	2 8
2	push 2 to stack	2 8 2
3	push 3 to stack	2 8 2 3
*	pop from stack pop from stack calculate 2 * 3 and push result to stack	2 8 6
-	pop from stack pop from stack calculate 8 - 6 and push result to stack	2 2
2	push 2 to stack	2 2 2
/	pop from stack pop from stack calculate 2 / 2 and push result to stack	2 1
+	pop from stack pop from stack calculate 2 + 1 and push result to stack	3
8	push 8 to stack	3 8
+	pop from stack pop from stack calculate 3 + 8 and push result to stack	11
10	push 10 to stack	11 10
5	push 5 to stack	11 10 5
/	pop from stack pop from stack calculate 10 / 5 and push result to stack	11 2
-	pop from stack pop from stack calculate 11 - 2 and push result to stack	9

Conclusion :

Infix : A + ((B – C * D) / E) + F – G / H   = 2 + ((8 – 2 * 3) / 2) + 8 – 10 / 5   = 9

Postfix : A B C D * - E / + F + G H / -   = 2 8 2 3 * - 2 / + 8 + 10 5 / -   = 9

ii) ! (A && ! ((B < C) || (C > D))) || (C < E)

a) Convert infix to prefix

Infix to Prefix Input String : ! (A && ! ((B < C) || (C > D))) || (C < E)			
Character	**Operations**	**Prefix**	**Operator Stack**
)	push) to stack		)
E	insert E to index 0 of prefix	E	)
<	push < to stack	E	)<
C	insert C to index 0 of prefix	CE	)<
(	pop from stack and insert it to index 0 of prefix pop from stack	<CE	
||	push || to stack	<CE	||
)	push) to stack	<CE	||)
)	push) to stack	<CE	||))
)	push) to stack	<CE	||)))
D	insert D to index 0 of prefix	D<CE	||)))
>	push > to stack	D<CE	||)))>
C	insert C to index 0 of prefix	CD<CE	||)))>
(	pop from stack and insert it to index 0 of prefix pop from stack	>CD<CE	||))
||	push || to stack	>CD<CE	||))||
)	push) to stack	>CD<CE	||))||)
C	insert C to index 0 of prefix	C>CD<CE	||))||)
<	push < to stack	C>CD<CE	||))||)<
B	insert B to index 0 of prefix	BC>CD<CE	||))||)<
(	pop from stack and insert it to index 0 of prefix pop from stack	<BC>CD<CE	||))||
(	pop from stack and insert it to index 0 of prefix pop from stack	||<BC>CD<CE	||)
!	push ! to stack	||<BC>CD<CE	||)!
&&	pop from stack and insert it to index 0 of prefix push && to stack	!||<BC>CD<CE	||)&&
A	insert B to index 0 of prefix	A!||<BC>CD<CE	||)&&
(	pop from stack and insert it to index 0 of prefix pop from stack	&&A!||<BC>CD<CE	||
!	push ! to stack	&&A!||<BC>CD<CE	||!
End of string	pop from stack and insert it to index 0 of prefix pop from stack and insert it to index 0 of prefix	||!&&A!||<BC>CD<CE	

Conclusion :

Infix : ! (A && ! ((B < C) || (C > D))) || (C < E)

Postfix : || ! && A ! || < B C > C D < C E

b) Evaluation Of Prefix

Evaluation of Prefix **Prefix Expression : || ! && A ! || < B C > C D < C E** **Example : || ! && false ! || < 4 5 > 5 2 < 5 4**		
Character	**Operations**	**Stack**
4	push 4 to stack	4
5	push 5 to stack	4 5
<	pop from stack pop from stack calculate 5 < 4 and push result to stack	false
2	push 2 to stack	false 2
5	push 5 to stack	false 2 5
>	pop from stack pop from stack calculate 5 < 2 and push result to stack	false false
5	push 5 to stack	false false 5
4	push 4 to stack	false false 5 4
<	pop from stack pop from stack calculate 4 < 5 and push result to stack	false false true
||	pop from stack pop from stack calculate true || false and push result to stack	false true
!	pop from stack calculate ! true and push result to stack	false false
false	push false to stack	false false false
&&	pop from stack pop from stack calculate false && false and push result to stack	false false
!	pop from stack calculate ! false and push result to stack	false true
||	pop from stack pop from stack calculate true || false and push result to stack	true

Conclusion :

Infix : ! (A && ! ((B < C) || (C > D))) || (C < E) = ! (false && ! ((4 < 5) || (5 > 2))) || (5 < 4) = true

Postfix : || ! && A ! || < B C > C D < C E = || ! && false ! || < 4 5 > 5 2 < 5 4 = true

c) Convert infix to postfix

Infix to Postfix Input String : ! (A && ! ((B < C) || (C > D))) || (C < E)			
Character	**Operations**	**Postfix**	**Operator Stack**
!	push ! to stack		!
(	push (to stack		!(
A	append A to postfix	A	!(
&&	push && to stack	A	!(&&
!	push ! to stack	A	!(&&!
(	push (to stack	A	!(&&!(
(	push (to stack	A	!(&&!((
B	append B to postfix	AB	!(&&!((
<	push < to stack	AB	!(&&!((<
C	append C to postfix	ABC	!(&&!((<
)	pop from stack and append it to postfix pop from stack	ABC<	!(&&!(
||	push || to stack	ABC<	!(&&!(||
(	push (to stack	ABC<	!(&&!(||(
C	append C to postfix	ABC<C	!(&&!(||(
>	push > to stack	ABC<C	!(&&!(||(>
D	append D to postfix	ABC<CD	!(&&!(||(>
)	pop from stack and append it to postfix pop from stack	ABC<CD>	!(&&!(||
)	pop from stack and append it to postfix pop from stack	ABC<CD>||	!(&&!
)	pop from stack and append it to postfix pop from stack and append it to postfix pop from stack	ABC<CD>||!&&	!
||	pop from stack and append it to postfix push || to stack	ABC<CD>||!&&!	||
(	push (to stack	ABC<CD>||!&&!	||(
C	append C to postfix	ABC<CD>||!&&!C	||(
<	push < to stack	ABC<CD>||!&&!C	||(<
E	append E to postfix	ABC<CD>||!&&!CE	||(<
)	pop from stack and append it to postfix pop from stack	ABC<CD>||!&&!CE<	||
End of string	pop from stack and append it to postfix	ABC<CD>||!&&!CE<||	

Conclusion :

Infix : ! (A && ! ((B < C) || (C > D))) || (C < E)

Postfix : A B C < C D > || ! && ! C E < ||

d) Evaluation of Postfix

Evaluation of Postfix
Postfix Expression : A B C < C D > | | ! && ! C E < | |
Example : false 4 5 < 5 2 > | | ! && ! 5 4 < | |

Character	**Operations**	**Stack**
false	push false to stack	false
4	push 4 to stack	false 4
5	push 5 to stack	false 4 5
<	pop from stack pop from stack calculate 4 < 5 and push result to stack	false true
5	push 5 to stack	false true 5
2	push 2 to stack	false true 5 2
>	pop from stack pop from stack calculate 5 > 2 and push result to stack	false true true
| |	pop from stack pop from stack calculate true | | true and push result to stack	false true
!	pop from stack calculate ! true and push result to stack	false false
&&	pop from stack pop from stack calculate false && false and push result to stack	false
!	pop from stack calculate ! false and push result to stack	true
5	push 5 to stack	true 5
4	push 4 to stack	true 5 4
<	pop from stack pop from stack calculate 5 < 4 and push result to stack	true false
| |	pop from stack pop from stack calculate true | | false and push result to stack	true

Conclusion :

Infix : ! (A && ! ((B < C) | | (C > D))) | | (C < E) = ! (false && ! ((4 < 5) | | (5 > 2))) | | (5 < 4) = true

Postfix : A B C < C D > | | ! && ! C E < | | = false 4 5 < 5 2 > | | ! && ! 5 4 < | | = true