GIT Department of Computer Engineering

CSE 222/505 - Spring 2020

Homework 4_Q1

BARAN HASAN BOZDUMAN

171044036

Q1 A + ((B – C * D) / E) + F – G / H

Postfix:

-We get string and start from first char and if the given is operator we push that into stack other wise we append the characters because of operation priorty becouse of paranthesies we push operators until match with the paranthesies if it match with paranthesies we append them to expression

-Also we check the operator priorities if the two operator which has same predence we append on top of the stack and after that we push other operator

TOKEN	STACK	OPERATIONS	**POSTFIX**
A	**top**	Append(A)	**A**
+	**+**	Push(+)	**A**
(	**+(**	Push(()	**A**
(	**+((**	Push(()	**A**
B	**+((**	Append(B)	**AB**
-	**+((-**	Push(-)	**AB**
C	**+((-**	Append(C)	**ABC**
*****	**+((-***	Push(*)	**ABC**
D	**+((-***	Append(D)	**ABCD**
)	**+(**	Pop() Append(*) Pop() Append(-)	**ABCD*-**
/	**+(/**	Push(/)	**ABCD*-**
E	**+(/**	Append(E)	**ABCD*-E**
)	**+**	Pop() Append(/) Pop()	**ABCD*-E/**
+	**+**	Push(+) Pop() Append(+)	**ABCD*-E/+**
F	**+**	Append(F)	**ABCD*-E/+F**
-	**-**	Pop() Append(+) Push(-)	**ABCD*-E/+F+**
G	**-**	Append(G)	**ABCD*-E/+F+G**
/	**-/**	Push(/)	**ABCD*-E/+F+G**
H	**-/**	Append(H)	**ABCD*-E/+F+GH**
End of the token		Pop() Append(/) Pop() Append(-)	**ABCD*-E/+F+GH/-**

Evaluation for postfix expresssion

A=1, B=23, C=4, D=5, E=6, F =7, G=8, H=2 the result must be 9/2

Expression	Action	Stack
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 1	1

1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 23	1 23
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 4	123 4
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 5	1 23 4 5
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Pop 5 and 4 Evaluate 4*5 Push 20	1 23 20
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Pop 23 and 20 Evaluate 23-20 Push 3	1 3
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 6	1 3 6
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Pop 6 and 3 Evaluate 3/6 push 2	1 1/2
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Pop 1/2 and 1 Evaluate 1+1/2 Push 3/2	3/2
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 7	3/2 7
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Pop 7 and 3/2 Evaluate 7+3/2 Push 17/2	17/2
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 8	17/2 8
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 2	17/2 8 2
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Pop 2 and 8 Evaluate 8/2 Push 4	17/2 4
1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Push 4 and 17/2 Evaluate 17/2 - 4 Push 9/2	9/2

1, 23, 4, 5, *, -, 6, / , +, 7, +, 8, 2, /, - |	Pop 9/2 Stack is empty Result is 9/2	

The results match.

Prefix:

-We can use postfix operations on prefix expression but firstly we need to reverse string after that we can place it like postfix operation.

-And end of the process we need to reverse again to get prefix expression also if you append elements to first index each time ,It works too.

TOKEN	STACK	OPERATIONS	**PREFIX**
H	**top**	Append(H)	**H**
/	**/**	Push(/)	**H**
G	**/**	Append(G)	**HG**
-	**-**	Push(-)	**HG/**
F	**-**	Append(F)	**HG/F**
+	**+**	Pop() Append(-) Push(+)	**HG/F-**
)	**+)**	Push())	**HG/F-**
E	**+)**	Append(E)	**HG/F-E**
/	**+)/**	Push(/)	**HG/F-E**
)	**+)/)**	Push())	**HG/F-E**
D	**+)/)**	Append(D)	**HG/F-ED**
*****	**+)/)***	Push(*)	**HG/F-ED**
C	**+)/)***	Append(C)	**HG/F-EDC**
-	**+)/)-**	Pop() Append(*) Push(-)	**HG/F-EDC***
B	**+)/)-**	Append(B)	**HG/F-EDC*B**
(	**+)/**	Pop() Append(-)	**HG/F-EDC*B-**
(	**+**	Pop() Append(/)	**HG/F-EDC*B-/**
+	**+**	Push(+) Pop() Append(+)	**HG/F-EDC*B-/+**
A	**+**	Append(A)	**HG/F-EDC*B-/+A**
End Of the token		Pop() Append(+)	**HG/F-EDC*B-/+A+**

Postfix=>ABCD*-E/+F+GH/- Prefix=>+A+/-B*CDE-F/GH

Evaluation for prefix expresssion

A=1, B=26, C=4, D=5, E=6, F =7, G=8, H=2 the result must be 5

Expression	Action	Stack
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 2	2
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 8	2 8
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Pop 8 and 2 Evaluate 8 / 2 Push 4	4
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 7	4 7
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Pop 7 and 4 Evaluate 7-4 Push 3	3
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 6	3 6
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 5	3 6 5
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 4	3 6 5 4
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Pop 4 and 5 Evaluate 4*5 Push 20	3 6 20
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 26	3 6 20 26
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2	Pop 26 and 20	3 6 6

	Evaluate 26-20 Push 6	
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Pop 6 and 6 Evaluate 6 / 6 Push 1	3 1
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Pop 1 and 3 Evaluate 1+3 Push 4	4
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Push 1	4 1
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Pop 1 and 4 Evluate 1+4 Push 5	5
+, 1, +, /, -, 26, *, 4, 5, 6, -, 7, /, 8, 2 |	Pop 5 Stack is empty Result is 5	

The results matches.

Q2 !(A &&! ((B < C) || (C > D))) || (C < E)

TOKEN	STACK -->top	OPERATIONS	**POSTFIX**
!	**!**	Push(!)	
(	**!(**	Push(()	
A	**!(**	Append(A)	**A**
&&	**!(&&**	Push(&&)	**A**
!	**!(&&!**	Push(!)	**A**
(	**!(&&!(**	Push(()	**A**
(	**!(&&!((**	Push(()	**A**
B	**!(&&!((**	Append(B)	**AB**
<	**!(&&!((<**	Push(<)	**AB**
C	**!(&&!((<**	Append(C)	**ABC**
)	**!(&&!(**	Pop() Append(<)	**ABC<**
||	**!(&&!(||**	Push(||)	**ABC<**
(	**!(&&!(||(**	Push(()	**ABC<**
C	**!(&&!(||(**	Append(C)	**ABC<C**
>	**!(&&!(||(>**	Push(>)	**ABC<C**
D	**!(&&!(||(>**	Append(D)	**ABC<CD**
)	**!(&&!(||**	Pop() Append(>)	**ABC<CD>**
)	**!(&&!**	Pop() Append(||)	**ABC<CD>||**
)	**!**	Pop() Append(!) Pop() Append(&&)	**ABC<CD>||!&&**
||	**||**	Push(||)	**ABC<CD>||!&&!**
(	**||(**	Push(()	**ABC<CD>||!&&!**
C	**||(**	Append(C)	**ABC<CD>||!&&!C**
<	**||(<**	Push(<)	**ABC<CD>||!&&!C**
E	**||(<**	Append(E)	**ABC<CD>||!&&!CE**
)	**||**	Pop() Append(<)	**ABC<CD>||!&&!CE<**
End of token		Pop() Append(||)	**ABC<CD>||!&&!CE<||**

Evaluation for postfix expression

A=1, B=2, C=3, D=4, E=5 the result must be 1

Expression	Action	Stack
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Push 1	1
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Push 2	1 2
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Push 3	1 2 3
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 3 and 2 Evaluate 2 < 3 Push 1	1 1
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Push 3	1 1 3
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Push 4	1 1 3 4
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 4 and 3 Evaluate 3>4 Push 0	1 1 0
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 0 and 1 Evaluate 0 ||1 Push 1	1 1
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 1 Evaluate !1 Push 0	1 0
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 0 and 1 Evaluate 1 &&0 Push 0	0
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 0 Evaluate ! 0	1

	Push 1	
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Push 3	1 3
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Push 5	1 3 5
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 5 and 3 Evaluate 3<5 Push 1	1 1
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 1 and 1 Evaluate 1||1 Push 1	1
1, 2, 3, <, 3, 4, >, ||, !, &&, !, 3, 5, <, || |	Pop 1 Stack is empty Result is 1	

The results match.

TOKEN	STACK -->|top	OPERATIONS	**PREFIX**
)	**)**	**Push())**	
E	**)**	**Append(E)**	**E**
<	**)<**	**Push(<)**	**E**
C	**)<**	**Append(C)**	**EC**
(		Pop() **Append(<)**	**EC<**
||	**||**	**Push(||)**	**EC<**
)	**||)**	**Push())**	**EC<**
)	**||))**	**Push())**	**EC<**
)	**||)))**	**Push())**	**EC<**
D	**||)))**	**Append(D)**	**EC<D**
>	**||)))>**	**Push(>)**	**EC<D**
C	**||)))>**	**Append(C)**	**EC<DC**
(	**||))**	Pop() **Append(>)**	**EC<DC>**
||	**||))||**	**Push(||)**	**EC<DC>**
)	**||))||)**	**Push())**	**EC<DC>**
C	**||))||)**	**Append(C)**	**EC<DC>C**
<	**||))||)<**	**Push(<)**	**EC<DC>C**
B	**||))||)<**	**Append(B)**	**EC<DC>CB**
(	**||))||**	Pop() **Append(<)**	**EC<DC>CB<**
(	**||)**	Pop() **Append(||)**	**EC<DC>CB<||**
!	**||)!**	**Push(!)**	**EC<DC>CB<||**
&&	**||)&&**	Pop() **Append(!)** **Push(&&)**	**EC<DC>CB<||!**
A	**||)&&**	**Append(A)**	**EC<DC>CB<||!A**
(	**||**	Pop() **Append(&&)**	**EC<DC>CB<||!A&&**
!	**||!**	**Push(!)**	**EC<DC>CB<||!A&&**
End of token	**||**	Pop() **Append(!)**	**EC<DC>CB<||!A&&!**
		Pop() **Append(||)**	**EC<DC>CB<||!A&&!||**

Postfix=>ABC<CD>||!&&!CE<|| **Prefix=>||!&&A!||<BC>CD<CE**

Evaluation for prefix expression

A=1, B=2, C=3, D=4, E=5 the result must be 1

Expression	Action	Stack
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Push 5	5
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Push 4	5 4
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 4 and 5 Evaluate 4 <5 Push 1	1
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Push 4	1 4
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Push 3	1 4 3
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 3 and 4 Evaluate 3 >4 Push 0	1 0
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Push 3	1 0 3
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Push 2	1 0 3 2
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 2 and 3 Evaluate 2<3 Push 1	1 0 1
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5	Pop 1 and 0	1 1

| |	Evaluate 1 || 0 Push 1	
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 1 Evaluate !1 Push 0	1 0
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Push 1	1 0 1
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 1 and 0 Evaluate 0 && 1 Push 0	1 0
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 0 Evaluate!0 Push 1	1 1
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 1 and 1 Evaluate 1 || 1 Push 1	1
||, !, &&, 1, !, ||, <, 2, 3, >, 3, 4, <, 4, 5 |	Pop 1 Stack is empty Result is 1	

The results match.