Q1- I) Convert to Postfix

Input	Postfix Stack	Operator Stack	Info
A+((B-C*D)/E)+F-G/H	A		A is an operand append to output
A+((B-C*D)/E)+F-G/H	A	+	+ is an operator append to operators
A+((B-C*D)/E)+F-G/H	A	+(	(is an operator append to operators
A+((B-C*D)/E)+F-G/H	A	+((	(is an operator append to operators
A+((B-C*D)/E)+F-G/H	AB	+((	B is an operand append to output
A+((B-C*D)/E)+F-G/H	AB	+((-	- is an operator append to operators
A+((B-C*D)/E)+F-G/H	ABC	+((-	C is an operand append to output
A+((B-C*D)/E)+F-G/H	ABC	+((-*	* is an operator append to operators
A+((B-C*D)/E)+F-G/H	ABCD	+((-*	D is an operand append to output
A+((B-C*D)/E)+F-G/H	ABCD*-	+(	Encountered “)”, pop operators until next “(“
A+((B-C*D)/E)+F-G/H	ABCD*-	+(/	/ is an operator append to operators
A+((B-C*D)/E)+F-G/H	ABCD*-E	+(/	E is an operand append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/	+	Encountered “)”, pop operators until next “(“
A+((B-C*D)/E)+F-G/H	ABCD*-E/+	+	+ is an operator with same precedence pop operators and push to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F	+	F is an operand append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+	-	+ is an operator with same precedence pop operators and push to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+G	-	G is an operand append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+G	-/	/ is an operator append to operators
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+GH	-/	H is an operand append to output
A+((B-C*D)/E)+F-G/H	ABCD*-E/+F+GH/-		No more items left Pop the operator stack until empty

Q1- I) Evaluate the converted Postfix

Evaluating Postfix expression “ABCD*-E/+F+GH/-” for:
A=1, B=2, C=3, D=4, E=5, F=6, G=7, H=8

Input	Operand Stack	Current Expression	Info
1 2 3 4 * - 5 / + 6 + 7 8 / -	1		1 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	2 1		2 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	3 2 1		3 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	4 3 2 1		4 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	12 2 1	3 * 4 = 12	* is an operator pop last 2 operands from stack and do expression
1 2 3 4 * - 5 / + 6 + 7 8 / -	-10 1	2 – 12 = -10	- is an operator pop last 2 operands from stack and do expression
1 2 3 4 * - 5 / + 6 + 7 8 / -	5 -10 1		5 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	-2 1	-10 / 5 = -2	/ is an operator pop last 2 operands from stack and do expression
1 2 3 4 * - 5 / + 6 + 7 8 / -	-1	1 + -2 = -1	+ is an operator pop last 2 operands from stack and do expression
1 2 3 4 * - 5 / + 6 + 7 8 / -	6 -1		6 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	5	-1 + 6 = 5	+ is an operator pop last 2 operands from stack and do expression
1 2 3 4 * - 5 / + 6 + 7 8 / -	7 5		7 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	8 7 5		8 is an operand push to operand stack
1 2 3 4 * - 5 / + 6 + 7 8 / -	0.875 5	7 / 8 = 0.875	/ is an operator pop last 2 operands from stack and do

			expression
1 2 3 4 * - 5 / + 6 + 7 8 / -	4.125	5 – 0.875 = 4.125	- is an operator pop last 2 operands from stack and do expression

Q1- I) Convert to Prefix

To find the prefix of expression: A+((B-C*D)/E)+F-G/H

1. Reverse the string: H/G-F+(E/(D*C-B))+A

2. Find postfix of this reversed expression:

Input	Postfix Stack	Operator Stack
H/G-F+(E/(D*C-B))+A	H	
H/G-F+(E/(D*C-B))+A	H	/
H/G-F+(E/(D*C-B))+A	HG	/
H/G-F+(E/(D*C-B))+A	HG/	-
H/G-F+(E/(D*C-B))+A	HG/F	-
H/G-F+(E/(D*C-B))+A	HG/F-	+
H/G-F+(E/(D*C-B))+A	HG/F-	+(
H/G-F+(E/(D*C-B))+A	HG/F-E	+(
H/G-F+(E/(D*C-B))+A	HG/F-E	+(/
H/G-F+(E/(D*C-B))+A	HG/F-E	+(/(
H/G-F+(E/(D*C-B))+A	HG/F-ED	+(/(
H/G-F+(E/(D*C-B))+A	HG/F-ED	+(/(*
H/G-F+(E/(D*C-B))+A	HG/F-EDC	+(/(*
H/G-F+(E/(D*C-B))+A	HG/F-EDC*	+(/(-
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B	+(/(-
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-	+(/
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/	+
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/+	+
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/+A	+
H/G-F+(E/(D*C-B))+A	HG/F-EDC*B-/+A+	

3. Now we reverse the postfix expression of previously reversed expression:

+A+/-B*CDE-F/GH

Q1- I) Evaluate the converted Prefix

Evaluating Prefix expression “+A+/-B*CDE-F/GH” for:
A=1, B=2, C=3, D=4, E=5, F=6, G=7, H=8

Input	Operand Stack	Expression
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	8	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	8 7	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	0,875	7 / 8 = 0,875
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	0,875 6	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125	6 – 0,875 = 5,125
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125 5	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125 5 4	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125 5 4 3	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125 5 12	4 * 3 = 12
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125 5 12 2	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125 5 -10	2 – 12 = -10
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	5,125 -2	-10 / 5 = -2
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	3,125	-2 + 5,125 = 3,125
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	3,125 1	
+ 1 + / - 2 * 3 4 5 - 6 / 7 8	4,125	1 + 3,125 = 4,125

Q1- II) Convert to Postfix

Input	Postfix Stack	Operator Stack	Info
!(A&&!((B<C)||(C>D)))||(C<E)		!	! is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)		!(	(is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	A	!(	A is an operand append to output
!(A&&!((B<C)||(C>D)))||(C<E)	A	!(&&	&& is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	A	!(&&!	! is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	A	!(&&!(	(is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	A	!(&&!((	(is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	AB	!(&&!((	B is an operand append to output

!(A&&!((B<C)||(C>D)))||(C<E)	AB	!(&&!((<	< is an operators append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	ABC	!(&&!((<	C is an operand append to output
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<	!(&&!(	Encountered “)”pop the operator stack until next “(“.
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<	!(&&!(||	|| is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<	!(&&!(||(	(is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<C	!(&&!(||(	C is an operand append to output
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<C	!(&&!(||(>	> is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD	!(&&!(||(>	D is an operand append to output
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>	!(&&!(||	Encountered “)”pop the operator stack until next “(“.
!(A&&!((B<C)||(C>D))||(C<E)	ABC<CD>||	!(&&!	Encountered “)”pop the operator stack until next “(“.
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&	!	Encountered “)”pop the operator stack until next “(“.
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&!	||	|| is an operator with less precedence pop operators and push to output
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&!	||(	(is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&!C	||(	C is an operand append to output
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&!C	||(<	< is an operator append to operators
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&!CE	||(<	E is an operand append to output
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&!CE<	||	Encountered “)”pop the operator stack until next “(“.
!(A&&!((B<C)||(C>D)))||(C<E)	ABC<CD>||!&&!CE<||		Empty the operator stack

Q1- II) Evaluate the converted Postfix

Evaluating Postfix expression “ABC<CD>||!&&!CE<||” for:

A=1, B=0, C=1, D=0, E=1

Input	Operand Stack	Expression
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1	
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 0	
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 0 1	
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 1	0 < 1 = 1
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 1 1	
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 1 1 0	
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 1 1	1 > 0 = 1
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 1	1 || 1 = 1
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 0	!1 = 0
1 0 1 < 1 0 > || ! && ! 1 1 < ||	0	1 && 0 = 0
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1	!0
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 1	
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 1 1	
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1 0	1 < 1 = 0
1 0 1 < 1 0 > || ! && ! 1 1 < ||	1	1 || 0 = 1

Infix version of expression was: !(A&&!((B<C)||(C>D)))||(C<E)

With values above: !(1&&!((0<1)|| (1>0)))||(1<1)

!(1&&!(1|| 1))||0

!(1&&!1)||0

!(1&&0)||0

!(0)||0

1||0

1

Evaluations holds up.

Q1- II) Convert to Prefix

To find the prefix of expression: !(A&&!((B<C)||(C>D)))||(C<E)

1. Reverse the string: (E<C)||(((D>C)||(C<B))!&&A)!
2. Find postfix of this reversed expression:

Input	Postfix Stack	Operator Stack
(E<C)||(((D>C)|| (C<B))!&&A)!		(
(E<C)||(((D>C)|| (C<B))!&&A)!	E	(
(E<C)||(((D>C)|| (C<B))!&&A)!	E	(<
(E<C)||(((D>C)|| (C<B))!&&A)!	EC	(<
(E<C)||(((D>C)||	EC<	

(C<B))!&&A)!		
(E<C)||(((D>C)||(C<B))!&&A)!	EC<	||
(E<C)||(((D>C)||(C<B))!&&A)!	EC<	||(
(E<C)||((D>C)||(C<B))!&&A)!	EC<	||((
(E<C)||(((D>C)||(C<B))!&&A)!	EC<	||(((
(E<C)||(((D>C)||(C<B))!&&A)!	EC<D	||(((
(E<C)||(((D>C)||(C<B))!&&A)!	EC<D	||(((>
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC	||(((>
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>	||((
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>	||((||
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>	||((||(
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>C	||((||(
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>C	||((||(<
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB	||((||(<
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<	||((||
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<||	||(
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<||	||(!
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<||!	||(&&
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<||!A	||(&&
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<||!A&&	||
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<||!A&&	||!
(E<C)||(((D>C)||(C<B))!&&A)!	EC<DC>CB<||!A&&!||	

3. Now we reverse the postfix expression of previously reversed expression:

||!&&A!||<BC>CD<CE

Q1- II) Evaluate the converted Prefix

Evaluating Prefix expression “||!&&A!||<BC>CD<CE” for:

A=1, B=0, C=1, D=0, E=1

Input	Operand Stack	Expression
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	1	
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	1 1	
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0	1 < 1 = 0
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 0	
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 0 1	
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 1	1 > 0 = 1
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 1 1	
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 1 1 0	
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 1 1	0 < 1 = 1
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 1	1 || 1 = 1
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 0	!1 = 0
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 0 1	
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 0	0 && 1
|| ! && 1 ! || < 0 1 > 1 0 < 1 1	0 1	!0 = 1
|| ! && 1 ! || < 0 1 > 1 0 <1 1	1	0 || 1

Which again, holds with the evaluation of infix version.